Semiparametric Thresholding Least Squares Inference for Causal Effects with R

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

Abstract

The vignette explains how to use the causalTLSE package to estimate causal effects using the semiparametric thresholding least squares methods developped by Giurcanu et al. (2023).

1 Introduction

This document presents the `causalTLSE` package explaining in details all functions implemented in the package. It is intended for users interested in all the details about the methods presented in Giurcanu et al. (2023) and how they are implemented.

The main 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 non-treated and $X \in \mathbb{R}^q$ is a q-vector of confounders. We can approximate this model by the following regression:

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

where $U_{l,k} = u_{l,k}(X_l) = (u_{j,l,k}(X_l)\ :\ 1 \leq j \leq p_{l,k}) \in \mathbb{R}^{p_{l,k}}$ is a vector of basis functions corresponding to the l$^{\text{th}}$ nonparametric component $f_{l,k}(X_l)$, $\psi_{l,k} \in \mathbb{R}^{p_{l,k}}$ is an unknown vector of regression coefficients, $U_k = u_k(X) = (u_{l,k}(X_l)\ :\ 1 \leq l \leq q) \in \mathbb{R}^{p_k}$ and $\psi_k = (\psi_{l,k}\ :\ 1 \leq l \leq q) \in \mathbb{R}^{p_k}$, with $p_k = \sum_{l=1}^{q} p_{l,k}$. The paper proposes a data-driven method for selecting the vectors $U_0(X)$ and $U_1(X)$. Note that we allow the number of basis functions ($p_{l,k}$) to differ across confounders and groups.

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 $U = u(x) = (u_j(x)\ :\ 1 \leq j \leq p) \in \mathbb{R}^p$. We just need to keep in mind that it is different for the treated and non-treated groups and also for different confounders. Let $\{\kappa_1, ..., \kappa_{p-1}\}$ be a set of $p-1$ knots strictly inside the support of X satisfying $\kappa_1 < \kappa_2 <, ..., < \kappa_{p-1}$. In the case of local linear basis functions, we have:

[*] 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

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

Therefore, if the number of knots is equal to 1, we only have the two basis functions. Since the knots must be strictly inside the support of X, for any categorical variable with two levels, which includes as a special case binary variables, 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.

Note that for the sample regression

$$Y_i = \beta_0(1 - Z_i) + \beta_1 Z_i + \psi_0^T u_0(X_i)(1 - Z_i) + \psi_1^T u_1(X_i) Z_i + \zeta_i\,,$$

for $i = 1, ..., n$, the knots of X_l, $l = 1, ..., k$, must be strictly inside the sample range of $\{X_{li} \ : \ 1 \leq i \leq n\} \in \mathbb{R}^n$ instead of inside the support of X_l.

2 The causalTLSE package

2.1 Setting up the Model

The first step in using the package is to define the causal model. The model contains the information about the outcome (Y), the treatment indicator (Z), the covariates (X) and their knots ($\kappa_{l,k}$). This is the starting point before applying any basis selection method. 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 `causalTLSE` package.

```
library(causalTLSE)
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 the outcome. The dataset includes the continuous covariates age (`age`), education (`ed`), the 1975 real income (`re75`), and some binary variables (`black`, `hisp`, `married` and `nodeg`). We start by considering the covariates `age`, `re75`, `ed`, and ‘`married`. To setup the model, we simply by run the following command:

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

The left of | is designated for the formula linking the outcome (re78) and the treatment indicator (treat). The covariates are entered after | as a formula without a dependent variable. This formula works similarly to formulas in the `lm` function. For example, we can add interactions, transformations of the variables, etc. The following is an example:

```
model0 <-  setModel(re78 ~ treat | ~ age + I(age^2) + re75 + ed * married,
                    data=nsw)
```

This will create the vector of covariates {age, age^2, re75, ed, married, ed×married}. The function `setModel` creates an object of class `tlseModel` with its own print method, which will be presented later.

The following sub-sections explain all arguments of the function.

2.1.1 The starting knots

By default, the function automatically generates knots for each variable based on the following procedure. This procedure is applied separately for the treated and control groups. The term `sample size` means the number of observations in the treated or control 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 controls, the starting number of knots in the treated and control groups are respectively equal to 3=floor(200^0.3)-1 and 4=floor(300^0.3)-1. 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.
2. Let $(p-1)$ be the number of knots determined in the previous step. The 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 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, which includes as a special case binary variables, is equal to 0. For nominal variables with a small number of levels, the number of knots 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.

We can inspect the knots of the current model as follows. Note that each object in the package is S3-class, so the elements can be accessed using the operator $. The elements knots0 and knots1 are the list of knots for the control and treated groups. For example, in our case the initial knots for the treated are:

```
model1$knots1
```

```
## $age
## 20% 40% 60% 80%
##  19  22  25  28
##
## $re75
##       40%       60%       80%
##  357.9499 1961.8640 5588.6640
##
## $ed
## 20% 40% 60% 80%
##   9  10  11  12
##
## $married
## NULL
```

We see that it is set to NULL for married, because it is a binary variable. 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, but not for re75. The reason is that re75 contains a large fraction of zeros. Since the 20th percentile is equal to 0 and 0 is also the minimum value of ed75, it is dropped (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
```

By printing the object, we see a summary of the model. It includes the list of variables with a positive number of knots and the ones with no knots.

```
model1
```

```
## Semiparametric TLSE Model
## **************************
##
## Number of treated:  297
## Number of control:  425
## Number of missing values:  0
## Selection method: SLSE
## Covariates approximated by semiparametric TLSE:
##  age, re75, ed
## Covariates not approximated by semiparametric TLSE:
##  married
```

Note that the selection method is set to SLSE, which stands for Semiparametric Least Squares Estimator. We refer to this when the knots are automatically selected by the method described above. Later in the document, we will present methods for selecting a subset of the SLSE using TLSE.

As another example, the simulated dataset `simDat4` 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 nominal variables `X2` and `X3`, with 2 and 3 levels respectively, and `X4` is a binary variable. The levels for `X2` are {3,4} and for `X3` the levels are {1,2,3}.

```
data(simDat4)
model2 <- setModel(Y~Z |~X1+X2+X3+X4, data=simDat4)
model2$knots0
```

```
## $X1
##        40%        60%        80%
##  0.2531388  2.9118507 12.1110772
##
## $X2
## NULL
##
## $X3
## 40%
##   2
##
## $X4
## NULL
```

We see that the number of knots for the variables with 2 levels (`X2` and `X4`) are set to 0 and it is equal to 1 for the variable with 3 levels (`X3`).

2.1.2 Setting the knots manually

We have the control over the knots through the arguments `knots0` and `knots1`. When the arguments are missing (the default), all knots are set automatically. One way to set the number of knots to 0 for all variables in a given group is to set the argument to `NULL`. For example, the number of knots is equal to 0 for all variables of the treated group in the following:

```
setModel(re78~treat | ~age+re75+ed+married, data=nsw, knots1=NULL)
```

```
## Semiparametric TLSE Model
## **************************
##
## Number of treated:  297
## Number of control:  425
## Number of missing values:  0
## Selection method: User Based
## Covariates approximated by semiparametric TLSE:
##  Treated: None
##  Control: age, re75, ed
## Covariates not approximated by semiparametric TLSE:
##  Treated: age, re75, ed, married
##  Control: married
```

Notice that the selection method is defined as "User Based" whenever knots are provided manually by the user. Also, the print method shows the lists of covariates by group only when they differ, which is the case here. The other option is to provide a list of knots. For each element, 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.

The following explains all possible formats for the list of knots.

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

  Suppose you want to set for the control group an automatic selection for `age`, no knots for `ed`, the knots $\{1000, 5000, 10000\}$ for `re75`, and the knots 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`. In the following, the argument `knots=TRUE` is added to the `print` method to only print the knots.

```
model <- setModel(re78~treat | ~age+re75+ed+married, data=nsw,
                  knots0=list(NA, c(1000,5000,10000), NULL, NA))
print(model, knots=TRUE)
```

```
## Semiparametric TLSE Model
## **************************
##
## Selection method: User Based
## Lists of knots for the treated group
## ************************************
## age:
## 20%  40%  60%  80%
##  19   22   25   28
## re75:
##       40%        60%        80%
##  357.9499  1961.8640  5588.6640
## ed:
## 20%  40%  60%  80%
##   9   10   11   12
## married:
## None
##
## Lists of knots for the Control group
## ************************************
## age:
## 16.66667%  33.33333%        50%  66.66667%  83.33333%
##        18         20         23         26         30
## re75:
##    k1      k2      k3
##  1000   5000  10000
## ed:
## None
## married:
## None
```

- A named list of length equal to the number of covariates. In that case, the order does not matter. It will automatically be reorder to match the order implied by the formula. The names must match perfectly the covariate names generated by R. This is particularly useful when we have interaction terms and we are not sure how R orders them.

  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 we want no knots for the interaction term. The name of the interaction depends on how we enter it in the formula. 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 knots to show that the order does not matter.

```
knots <- list(married=NA, ed=NULL, 'age:ed'=NULL,
              re75=c(1000,5000,10000), age=NA)
model <- setModel(re78~treat | ~age*ed+re75+married, data=nsw,
                  knots0=knots)
model$knots0
```

```
## $age
## 16.66667% 33.33333%       50% 66.66667% 83.33333%
##        18         20        23        26        30
##
## $ed
## NULL
##
## $re75
##    k1    k2    k3
##  1000  5000 10000
##
## $married
## NULL
##
## $`age:ed`
## NULL
```

- A named list of length strictly less than the number of covariates. The names of the selected covariates must match perfectly the names generated by R and the order does not matter.

  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:

```
knots <- list(ed=NULL, 'age:ed'=NULL, re75=c(1000,5000,10000))
model <- setModel(re78~treat | ~age*ed+re75+married, data=nsw,
                  knots0=knots)
model$knots0
```

```
## $age
## 16.66667% 33.33333%       50% 66.66667% 83.33333%
##        18         20        23        26        30
```

```
##
## $ed
## NULL
##
## $re75
##    k1    k2    k3
##  1000  5000 10000
##
## $married
## NULL
##
## $`age:ed`
## NULL
```

Note that the previous case offers an easy way of setting the number of knots to 0 for a subset of 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.

```
knots <- list('age:ed'=NULL, 'ed:re75'=NULL, 'ed:married'=NULL)
model <- setModel(re78~treat | ~age*ed+re75*ed+married*ed, data=nsw,
                  knots0=knots, knots1=knots)
model
```

```
## Semiparametric TLSE Model
## **************************
##
## Number of treated:  297
## Number of control:  425
## Number of missing values:  0
## Selection method: User Based
## Covariates approximated by semiparametric TLSE:
##  age, ed, re75
## Covariates not approximated by semiparametric TLSE:
##  married, age:ed, ed:re75, ed:married
```

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 extended set of covariates. We want to estimate the parameters of model

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

where U_0 and U_1 are defined above (which depends on the model knots). The function that estimates the model is `estModel`. The function has three arguments, but two of them are mostly used internally by other functions. We present it in case it is needed. The arguments are:

- `model`: A model created by the function `setModel`.
- `w0` and `w1`: lists of integers to select knots for the control and treated groups respectively. For example, suppose we have 2 covariates with 5 knots each. If we want to estimate the model with only the first knot for the first covariate and knots 3 and 5 for the second, we set `w0` to `list(1L,c(3L, 5L))`. By default they are set to `NULL` and all the knots from the model are used.

We illustrate the usage of `estModel` with a simple model containing 2 covariates and one knot per variable.

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

```
## Semiparametric TLSE Model
## *************************
##
## Selection method: SLSE
## Lists of knots for the treated group
## ************************************
## age:
## 50%
##  23
## married:
## None
##
## Lists of knots for the Control group
## ************************************
## age:
## 50%
##  23
## married:
## None
```

```
fit <- estModel(model)
fit
```

```
## Semiparametric TLSE Estimate
## ****************************
## Selection method: SLSE
##
## factor(treat)0  factor(treat)1        Xf0age_1        Xf0age_2      Xf0married
##     4558.28061      3754.98326        27.79868       -12.51415      -115.81593
##       Xf1age_1        Xf1age_2      Xf1married
##       89.25358        22.22331      1435.28205
```

The object has its own print method that returns the coefficient estimates. A more detrailed presentation of the results can be obtained using the `summary` method. The following is an example with just one knot per eligible variable.

```
summary(fit)
```

```
## Semiparametric TLSE Estimate
## ****************************
## Selection method: SLSE
##
##                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 contain an intercept and the R^2 is computed differently for models without an intercept. The definition of the R^2 used by R is the following (RSS means residual sum of squares):

$$R^2 = 1 - \frac{\text{RSS for the model with the regressors}}{\text{RSS for the model without the regressors}}\,.$$

In a model with an intercept, the residual of the model without the regressors is $Y_i - \bar{Y}$, but it is equal to Y_i when the model does not have an intercept. As a result, the R^2 with and without an intercept are respectively

$$R^2_{with} = 1 - \frac{\sum_{i=1}^n \hat{e}_i^2}{\sum_{i=1}^n (Y_i - \bar{Y})^2}$$

and

$$R^2_{without} = 1 - \frac{\sum_{i=1}^n \hat{e}_i^2}{\sum_{i=1}^n Y_i^2}\,.$$

However, our model does contain an intercept since we include a binary variable for both the control and treated groups.

2.3 The `predict` and `plot` method

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 control groups 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 of Zeileis (2006). The function returns a list of 2 elements, `treated` and `control`. 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
##
## $control
## [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
##
## $control
##        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 control, 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 treated:

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

```
## $fit
## [1] 6975.337 7064.591
##
## $se.fit
##        1        2
## 1188.116 1185.194
```

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

- **x**: An object of class `tlseFit`.
- **y**: An alias for `which` for compatibility with the generic `plot` function.
- **which**: covariate to plot against the outcome variable. It could be an integer (the position of the covariate) or a character (the name of the covariate)
- **interval**: The type of confidence interval to include. The default is "none". The other alternative is "confidence".
- **level**: The confidence level when `interval="confidence"`. The default is 0.95.
- **newdata**: An optional named vector of fixed values for some or all other covariates. The values of the covariates not specified are determined by the argument `FUN`.
- **legendPos**: The position of the legend. The default is "topright".
- **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"`.
- **col0, col1, lty0, lty1**: The line colors and shapes for the control and treated. The defaults are `col0=1` (black), `col1=2` (red), `lty0=1` (solid) and `lty1=2` (dashed).
- **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 "control".
- **cex**: The font size for the legend. The default is 1.
- **ylim, xlim**: optional ranges for the y-axis and x-axis.

- **addPoints**: Should we include the scatterplot of the outcome and covariate to the graph? The default is `FALSE`.
- **FUN**: A function to determine how the other covariates are fixed. The default is `mean`. Note that the function is applied to each group separately.
- **main**: An optional title to replace the default one.
- `...`: 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.

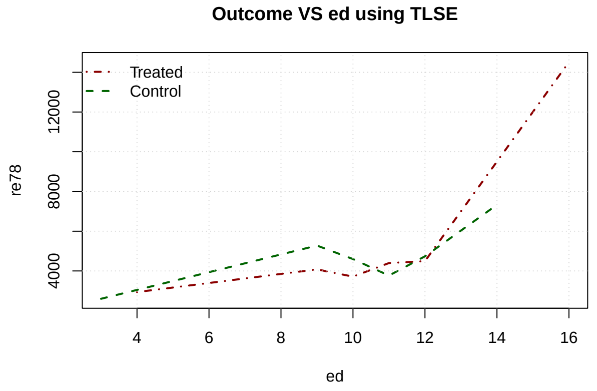
In the following, we illustrate some examples. Consider the model:

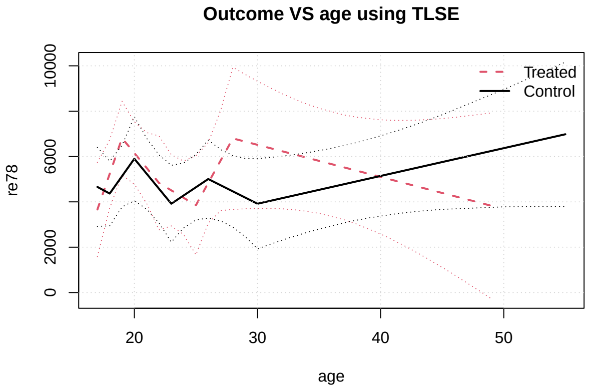
```
model1 <- setModel(re78~treat | ~age+re75+ed+married, data=nsw)
fit1 <- estModel(model1)
```

Suppose we want to compare the predicted income with respect to age or education, holding the other covariates 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.

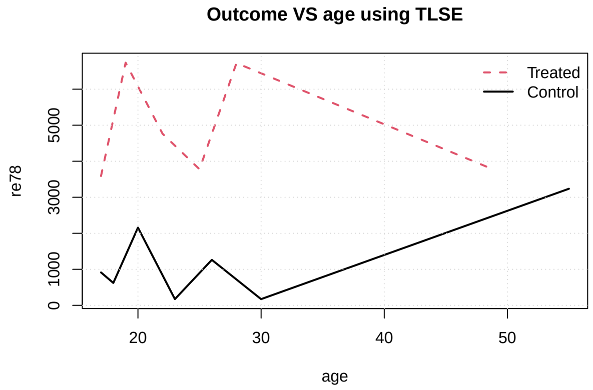
```
library(sandwich)
plot(fit1, "ed", col0="darkgreen", col1="darkred", lty0=2, lty1=4,
     legendPos="topleft", vcov.=vcov)
plot(fit1, "age", interval='confidence', level=0.9, type="HC1")
```

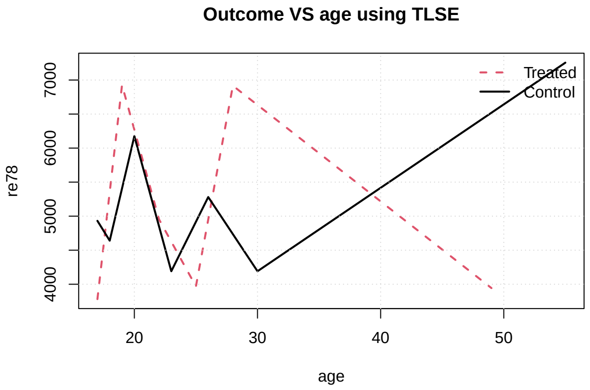




If we want to fix the other covariates using another function, we can change the argument `FUN`. The new function must be a function of one argument. For example, if we want to fix the other covariates to their group medians, we set `FUN` to `median` (no quotes). We proceed the same way for any function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

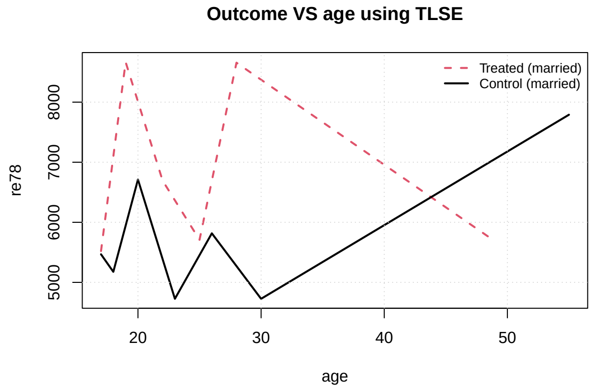
```
plot(fit1, "age", FUN=mode)
plot(fit1, "age", FUN=function(x) quantile(x, .20))
```

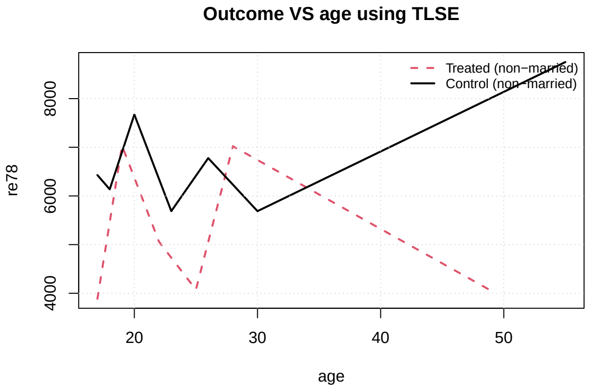




It is also possible to set some of the other covariates to a specific value by changing the argument `newdata`. This argument must be a named vector with the names corresponding to the variables you want to fix. You can also add a description to the legend with the argument `addToLegend`.

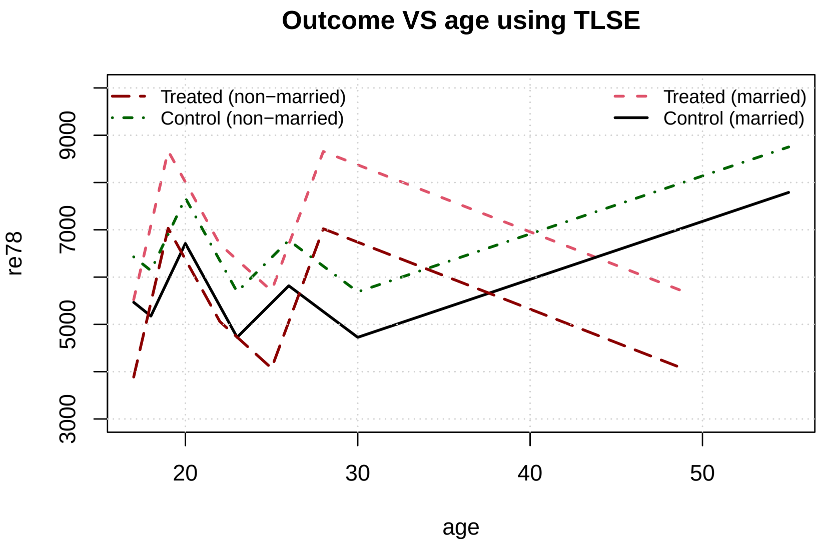
```
plot(fit1, "age", newdata=c(married=1, re75=10000), addToLegend="married", cex=0.8)
plot(fit1, "age", newdata=c(married=0, re75=10000), addToLegend="non-married", cex=0.8)
```





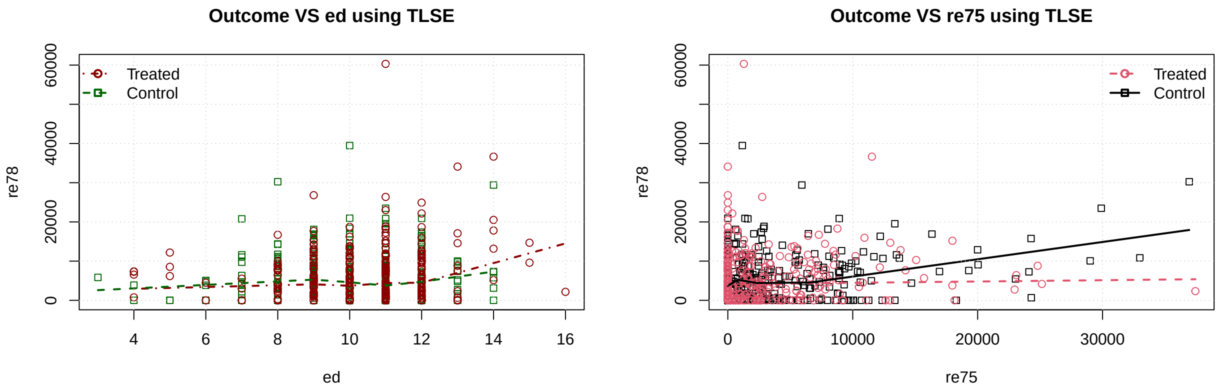
To better compare the two groups, it is also possible to have them plotted on the same graph by setting the argument `add.` to `TRUE`. We just need to adjust some of the arguments to better distinguish the different curves. In the following example, we set the colors and line shapes to different values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

```
plot(fit1, "age", newdata=c(married=1, re75=10000), addToLegend="married", cex=0.8,
     ylim.=c(3000,10000))
plot(fit1, "age", newdata=c(married=0, re75=10000), addToLegend="non-married", cex=0.8,
     legendPos='topleft', col0="darkgreen", col1="darkred", lty0=4, lty1=5,
     add.=TRUE)
```



Finally, it is also possible to add the observed points to the graph.

```
plot(fit1, "ed", col0="darkgreen", col1="darkred", lty0=2, lty1=4,
     legendPos="topleft", addPoints=TRUE)
plot(fit1, "re75", addPoints=TRUE)
```



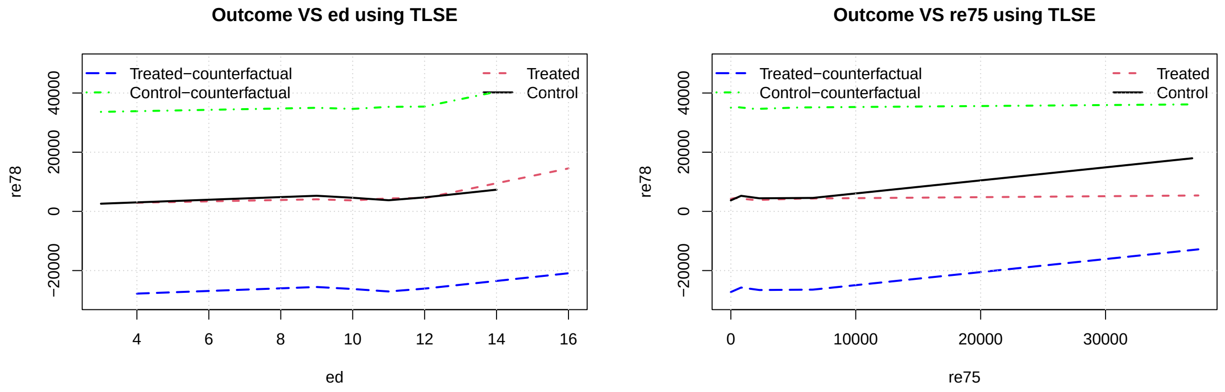
2.3.1 Counterfactual

We define the counterfactual predictions at $X = x$ for the treated and non-treated respectively as:

$$\begin{aligned}\hat{Y}_1 &= \hat{\beta}_1 + \hat{\psi}_0^T u_0(x)\\ \hat{Y}_0 &= \hat{\beta}_0 + \hat{\psi}_1^T u_1(x)\end{aligned}$$

It is computed by the `predict` method when the argument `counterfactual` is set to `TRUE`. We can also visualize the counterfactual prediction using `plot` with the same argument. In the following example, we compare the prediction with and the counterfactual prediction.

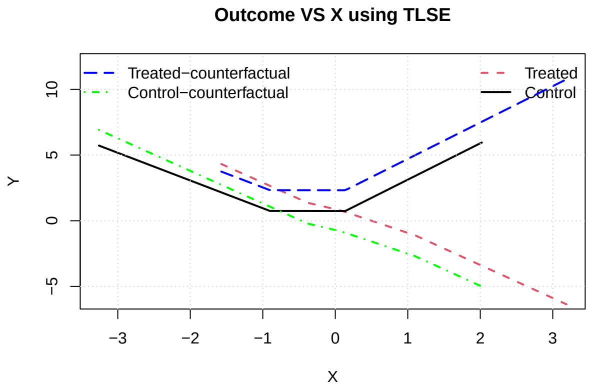
```
plot(fit1, "ed", ylim=c(-30000, 50000))
plot(fit1, "ed", counterfactual=TRUE, col0="green", col1="blue", lty0=4,
     lty1=5, add.=TRUE, legendPos="topleft")
plot(fit1, "re75", ylim=c(-30000, 50000))
plot(fit1, "re75", counterfactual=TRUE, col0="green", col1="blue", lty0=4,
     lty1=5, add.=TRUE, legendPos="topleft")
```



This is an example with the model 1. How this counterfactual will enter the vignette will be modified once we have tested it.

```
data(simDat1)
mod <- setModel(Y~Z | ~X, data=simDat1)
c1 <- causalTLSE(mod, selType="SLSE")
```

```
plot(c1, "X", ylim=c(-6,12))
plot(c1, "X", counterfactual=TRUE, col0="green", col1="blue", lty0=4,
     lty1=5, add.=TRUE, legendPos="topleft")
```



2.4 Optimal selection of the knots

We propose two methods for selecting the knots: the backward (BTLSE) and the forward (FTLSE) methods. For each method, we propose 3 criteria: the asymptotic (ASY), the Akaike Information (AIC), and the Bayesian Information (BIC). The two selection methods can be summarized as follows:

BTLSE:

1. We estimate the model with all knots included in the model.
2. For each knot, we test if the slopes of the basis function are the same before and after, and return the p-value.

3. The knots are selected using one of the following criteria
   - **ASY**: 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.

FTLSE:

1. We estimate the model by including a subset of the knots, one variable at the time. When we test a knot for one covariate, the number of knots is set to 0 for all the other covariates.
2. For each knot, we test if the slope of the piecewise linear polynomial is the same before and after, 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 before and after the knot.
   - Variables with 2 knots: we include the two knots and return the p-values of the test of equality before and after for each knot.
   - Variables with p knots ($p > 2$): We test the equality before and after the 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.
3. The knots are selected using one of the following criteria
   - **ASY**: 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 function `selTLSE`. The arguments are:

- **model**: An object of class `tlseModel`.
- **selType**: This is the selection method. We have the choice between “FTLSE” (the default) and “BTLSE”.
- **selCrit**: This is the criterion used by the selection method. We have the choice between “AIC” (the default), “BIC” or “ASY”.
- **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 covariate. 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 `method` is set to “ASY”.
- **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"`.
- `...`: This is used to pass arguments to the `vcov.` function.

The function returns a model of class `tlseModel` 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, knots=TRUE)
```

```
## Semiparametric TLSE Model
## **************************
##
## Selection method: SLSE
## Lists of knots for the treated group
## ************************************
## age:
## 20%  40%  60%  80%
##  19   22   25   28
## re75:
##       40%        60%        80%
##  357.9499  1961.8640  5588.6640
## ed:
## 20%  40%  60%  80%
##   9   10   11   12
## married:
## None
##
## Lists of knots for the Control group
## ************************************
## age:
## 16.66667%  33.33333%        50%  66.66667%  83.33333%
##        18         20         23         26         30
## re75:
##       50%  66.66667%  83.33333%
##  823.2544  2292.1710  6567.3290
## ed:
## 16.66667%  33.33333%  66.66667%  83.33333%
##         9         10         11         12
## married:
## None
```

```
model2 <- selTLSE(model1)
print(model2, knots=TRUE)
```

```
## Semiparametric TLSE Model
## **************************
##
## Selection method: FTLSE
## Criterion: AIC
##
## Lists of knots for the treated group
## ************************************
## age:
## 20%  60%  80%
##  19   25   28
## re75:
## None
## ed:
## 80%
##  12
## married:
## None
##
## Lists of knots for the Control group
## ************************************
## age:
## 33.33333%        50%
##        20         23
## re75:
##       50%  83.33333%
##  823.2544  6567.3290
## ed:
## 16.66667%  66.66667%
##         9         11
## married:
## None
```

For example, the FTLSE-AIC method has removed all knots from `re75` for the treated group and kept two knots for the control group. The print method indicates which method was used to select the knots. In the following example, we see BTLSE as selection method and BIC as criterion. Note that the BIC selects 0 knots for all covariates.

```
model3 <- selTLSE(model1, selType="BTLSE", selCrit="BIC")
model3
```

```
## Semiparametric TLSE Model
## **************************
##
## Number of treated:  297
## Number of control:  425
## Number of missing values:  0
## Selection method: BTLSE
## Criterion: BIC
##
## Covariates approximated by semiparametric TLSE:
##  None
## Covariates not approximated by semiparametric TLSE:
##  age, re75, ed, married
```

Since the function selTLSE function returns a new model, we can apply the estModel to it:

```
estModel(selTLSE(model1, selType="FTLSE", selCrit="BIC"))
```

```
## Semiparametric TLSE Estimate
## ****************************
## Selection method: FTLSE
## Criterion: BIC
##
## factor(treat)0  factor(treat)1          Xf0age         Xf0re75           Xf0ed
##   4.825878e+03   -3.889679e+02   -2.010566e+01    2.982477e-01    2.500219e+00
##     Xf0married          Xf1age         Xf1re75           Xf1ed      Xf1married
##  -1.094084e+03    4.105403e+01    2.676162e-02    4.849161e+02    1.417291e+03
```

2.5 The causalTLSE method for tlseFit objects

The regression model estimated by estModel, as described in the introduction, can be written as

$$Y_i = \beta_0(1-Z_i) + \beta_1 Z_i + \psi_0' u_0(X_i)(1-Z_i) + \psi_1' u_1(X_i) Z_i + \zeta_i \text{ for } i = 1, ..., n\,.$$

Let $\hat\beta_0$, $\hat\beta_1$, $\hat\psi_0$ and $\hat\psi_1$ be the least squares estimates of the above parameters. Then, the TLSE average causal effect (ACE), causal effect on the treated (ACT) and causal effect on the non-treated (ACN) are defined respectively as follows:

$$\begin{array}{rcl} \text{ACE} & = & \hat\beta_1 - \hat\beta_0 + \hat\phi_1'\overline{U_1} - \hat\phi_0'\overline{U_0} \\ \text{ACT} & = & \hat\beta_1 - \hat\beta_0 + \hat\phi_1'\overline{U_1 Z} - \hat\phi_0'\overline{U_0 Z} \\ \text{ACN} & = & \hat\beta_1 - \hat\beta_0 + \hat\phi_1'\overline{U_1(1-Z)} - \hat\phi_0'\overline{U_0(1-Z)}\,, \end{array}$$

where

$$\begin{aligned}\overline{U_j} &= \frac{1}{n}\sum_{i=1}^{n} u_j(X_i), \text{ for j=0,1} \\ \overline{U_jZ} &= \frac{1}{n_1}\sum_{i=1}^{n} u_j(X_i)Z_i, \text{ for j=0,1} \\ \overline{U_j(1-Z)} &= \frac{1}{n_0}\sum_{i=1}^{n} u_j(X_i)(1-Z_i), \text{ for j=0,1}\end{aligned}$$

and n_0 and n_1 are the sample size in the control and treated groups. The method `causalTLSE` estimates the causal effects from `tlseFit` objects using the knots included in the estimated model. The arguments of the method are:

- **object**: An object of class `tlseFit`.
- **seType**: The method to compute the standard errors of the causality measures. By default, they are computed using an analytic expression derived in the paper. Alternatively, we can set the argument to "lm" and use the least squares standard errors based on the asymptotic properties.
- **causal**: What causality measure should the function compute? We have the choice between "All" (the default), "ACT", "ACE" or "ACT".
- **vcov.**: An alternative function used to compute the covariance matrix of the least squares estimates. By default, `vcovHC` is used with `type="HC3"`.
- `...`: 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 <- setModel(re78 ~ treat | ~ age + re75 + ed + married, data=nsw)
fit1 <- estModel(model1)
causalTLSE(fit1)
```

```
## Causal Effect using Semiparametric TLSE
## ***************************************
## Selection method: SLSE
##
## ACE = 814.3083
## ACT = 831.8856
## ACN = 802.0249
```

We see that the selection method used to select the knots are set to SLSE. This is explained in the section "Setting up the Model". The method returns an object of class `causaltlse`. We see above what its `print` method returns. The following shows its `summary` method:

```
ce <- causalTLSE(fit1)
summary(ce)
```

```
## Causal Effect using Semiparametric TLSE
## ****************************************
## Selection method: SLSE
##     Estimate Std. Error t value Pr(>|t|)
## ACE    814.3      482.1   1.689   0.0912 .
## ACT    831.9      499.5   1.665   0.0958 .
## ACN    802.0      498.9   1.608   0.1079
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

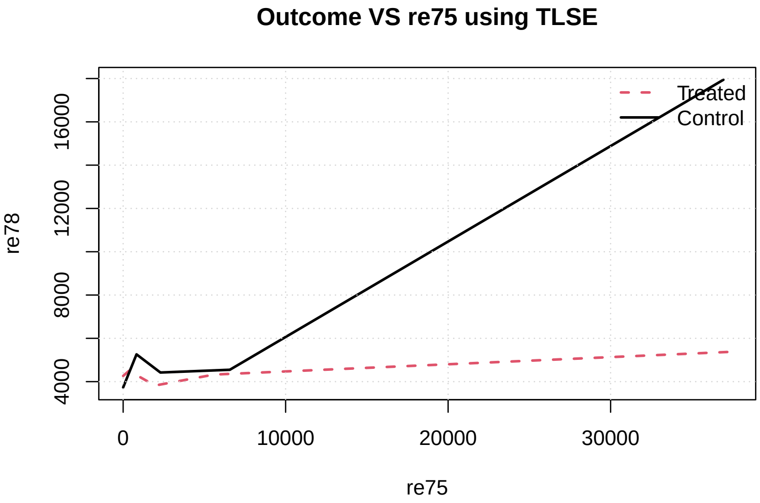
By default, the standard errors are computed using an analytic expression derived in the paper. In the following, we estimate the standard errors using the HC3 type of heteroskedasticity robust standard errors, which is the default when `seType="lm"`.

```
ce2 <- causalTLSE(fit1, seType="lm")
summary(ce2)
```

```
## Causal Effect using Semiparametric TLSE
## ****************************************
## Selection method: SLSE
##     Estimate Std. Error t value Pr(>|t|)
## ACE    814.3      506.1   1.609    0.108
## ACT    831.9      527.4   1.577    0.115
## ACN    802.0      514.2   1.560    0.119
```

The object `causaltlse` inherits from the class `tlseFit`, so we can apply the `plot` (or the `predict`) method directly on this object.

```
plot(ce2, "re75")
```



2.5.1 The extract method

The package comes with an `extract` method for objects of class `causaltlse`, which is a required method for creating Latex tables using the `texreg` package of Leifeld (2013). For example, we can compare different methods in a single table.

```
library(texreg)
c1 <- causalTLSE(fit1)
fit2 <- estModel(selTLSE(model1, selType="BTLSE"))
fit3 <- estModel(selTLSE(model1, selType="FTLSE"))
c2 <- causalTLSE(fit2)
c3 <- causalTLSE(fit3)
texreg(list(SLSE=c1, BTLSE=c2, FTLSE=c3), table=FALSE, digits=4)
```

	SLSE	BTLSE	FTLSE
ACE	814.3083	818.1598	824.4901
	(482.1393)	(482.8785)	(481.8267)
ACT	831.8856	837.0768	852.4659
	(499.4948)	(501.3497)	(496.6795)
ACN	802.0249	804.9401	804.9401
	(498.8671)	(491.0229)	(490.4101)
Num. knots (Control)	12	8	6
Num. knots (Treated)	11	4	4
Num. covariates	4	4	4
Num. obs. (Control)	425	425	425
Num. obs. (Treated)	297	297	297
R^2	0.0869	0.0852	0.0840
R$^2_{adj}$	0.0445	0.0577	0.0592

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

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**: Should the number of observations be printed? The default is `TRUE`.
- **include.nknots**: Should the number of knots be printed? The default is `TRUE`.
- **include.rsquared**: Should the R^2 be printed? The default is `TRUE`.
- **include.adjrsquared**: Should the adjusted R^2 be printed? 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, BTLSE=c2, FTLSE=c3), table=FALSE,
       which="ACE-ACT", include.adjrsquared=FALSE)
```

	SLSE	BTLSE	FTLSE
ACE	814.31	818.16	824.49
	(482.14)	(482.88)	(481.83)
ACT	831.89	837.08	852.47
	(499.49)	(501.35)	(496.68)
Num. knots (Control)	12	8	6
Num. knots (Treated)	11	4	4
Num. covariates	4	4	4
Num. obs. (Control)	425	425	425
Num. obs. (Treated)	297	297	297
R^2	0.09	0.09	0.08

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

2.6 The `causalTLSE` method for `tlseModel` objects

When applied directly to `tlseModel` objects, the `causalTLSE` method offers the possibility to select the knots and estimate the causal effects all at once. The method also returns an object of class `causaltlse`. The arguments are the same as the method for `tlseFit` objects, plus the necessary arguments for the knots selection. The following are the arguments not already defined for objects of class `tlseFit`. The details of these arguments are presented in the section Optimal selection of knots.

- **object**: An object of class `tlseModel`.
- **selType**: This is the selection method. We have the choice between "SLSE" (the default), "FTLSE" and "BTLSE". 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 "FTLSE" or "BTLSE".

- **pvalT**: This is a function that returns the p-value threshold. We explained this argument when we presented the `selTLSE` function.

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

```
c1 <- causalTLSE(model1, selType="SLSE")
c2 <- causalTLSE(model1, selType="BTLSE")
c3 <- causalTLSE(model1, selType="FTLSE")
texreg(list(SLSE=c1, BTLSE=c2, FTLSE=c3), table=FALSE, digits=4)
```

	SLSE	BTLSE	FTLSE
ACE	814.3083	818.1598	824.4901
	(482.1393)	(482.8785)	(481.8267)
ACT	831.8856	837.0768	852.4659
	(499.4948)	(501.3497)	(496.6795)
ACN	802.0249	804.9401	804.9401
	(498.8671)	(491.0229)	(490.4101)
Num. knots (Control)	12	8	6
Num. knots (Treated)	11	4	4
Num. covariates	4	4	4
Num. obs. (Control)	425	425	425
Num. obs. (Treated)	297	297	297
R^2	0.0869	0.0852	0.0840
R^2_{adj}	0.0445	0.0577	0.0592

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

2.7 The `causalTLSE` 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 the ones from the `setModel` function and the `causalTLSE` method for `tlseModel` objects. It creates the model, select the knots and estimate the causal effects in one step. For example, we can create the previous table as follows:

```
c1 <- causalTLSE(re78 ~ treat | ~ age + re75 + ed + married, data=nsw,
                 selType="SLSE")
c2 <- causalTLSE(re78 ~ treat | ~ age + re75 + ed + married, data=nsw,
                 selType="BTLSE")
c3 <- causalTLSE(re78 ~ treat | ~ age + re75 + ed + married, data=nsw,
                 selType="FTLSE")
texreg(list(SLSE=c1, BTLSE=c2, FTLSE=c3), table=FALSE, digits=4)
```

	SLSE	BTLSE	FTLSE
ACE	814.3083	818.1598	824.4901
	(482.1393)	(482.8785)	(481.8267)
ACT	831.8856	837.0768	852.4659
	(499.4948)	(501.3497)	(496.6795)
ACN	802.0249	804.9401	804.9401
	(498.8671)	(491.0229)	(490.4101)
Num. knots (Control)	12	8	6
Num. knots (Treated)	11	4	4
Num. covariates	4	4	4
Num. obs. (Control)	425	425	425
Num. obs. (Treated)	297	297	297
R^2	0.0869	0.0852	0.0840
R^2_{adj}	0.0445	0.0577	0.0592

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

Note that this method calls `setModel`, `selTLSE`, `estModel` and the method `causalTLSE` for `tlseFit` 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.

2.8 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 starting model:

```
data(simDat1)
mod <- setModel(Y~Z | ~X, data=simDat1)
```

Then we can compare three different methods:

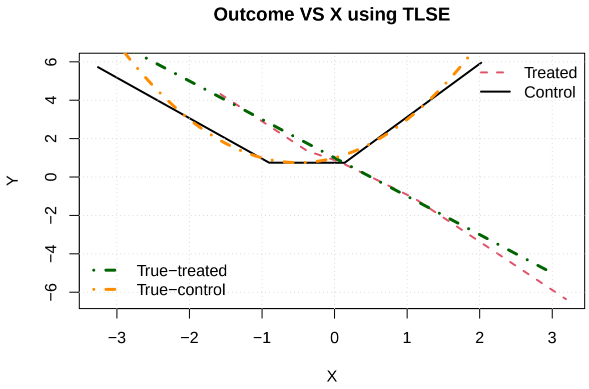
```
c1 <- causalTLSE(mod, selType="SLSE")
c2 <- causalTLSE(mod, selType="BTLSE", selCrit="BIC")
c3 <- causalTLSE(mod, selType="FTLSE", selCrit="BIC")
texreg(list(SLSE=c1, BTLSE=c2, FTLSE=c3), table=FALSE, digits=4)
```

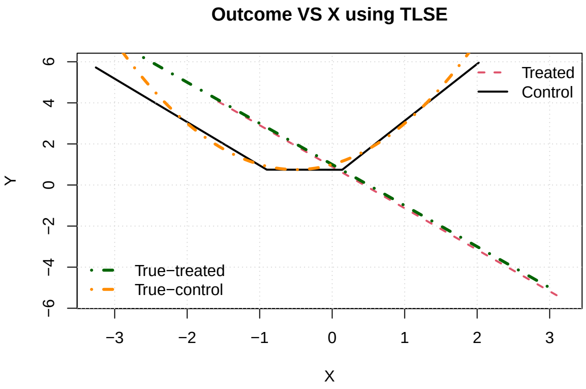
	SLSE	BTLSE	FTLSE
ACE	-1.4396^{***}	-1.4530^{***}	-1.4533^{***}
	(0.2614)	(0.2605)	(0.2599)
ACT	-1.9316^{***}	-1.9316^{***}	-1.9320^{***}
	(0.3030)	(0.3024)	(0.3030)
ACN	-0.0865	-0.1369	-0.1369
	(0.3263)	(0.3224)	(0.3224)
Num. knots (Control)	2	2	1
Num. knots (Treated)	4	0	0
Num. covariates	1	1	1
Num. obs. (Control)	80	80	80
Num. obs. (Treated)	220	220	220
R^2	0.7434	0.7386	0.7303
R^2_{adj}	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 (we ignore the FTLSE because the selected knots are the same):

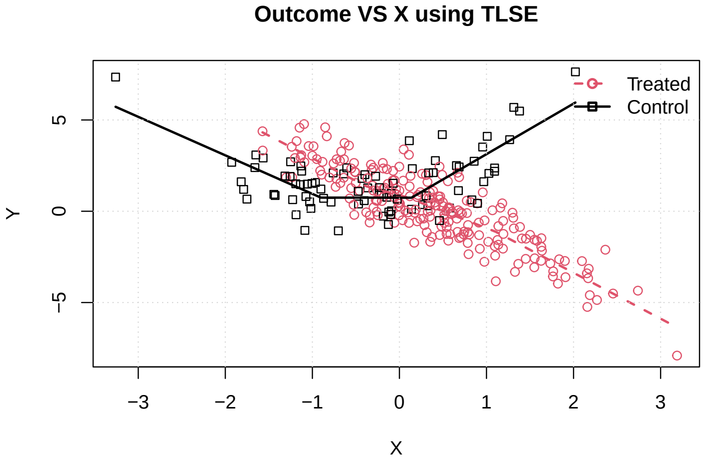
```
plot(c1, "X")
curve(1-2*x, -3,3, col="darkgreen", lty=4, lwd=3, add=TRUE)
curve(1+x+x^2, -3,3, col="darkorange", lty=4, lwd=3, add=TRUE)
legend("bottomleft", c("True-treated","True-control"),
       col=c("darkgreen","darkorange"), lty=4, lwd=3, bty='n')
plot(c2, "X")
curve(1-2*x, -3,3, col="darkgreen", 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-control"),
       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 $f_2(x)$. We can see from the folllowing graph how the lines are fit through the observations by group.

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



2.9 A simulated data set from Model 2

The dataset `datSim2` was generated using the following data generating process. It is a change point regression model (with unknown location of change points).

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

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, BTLSE with AIC and BTLSE with BIC.

```
data(simDat2)
mod <- setModel(Y~Z | ~X, data=simDat2)
```

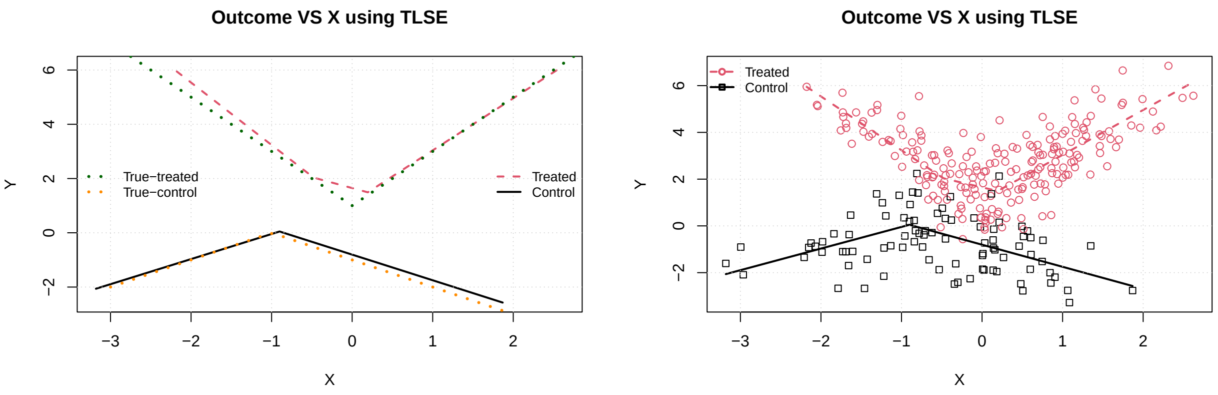
```
c1 <- causalTLSE(mod, selType="SLSE")
c2 <- causalTLSE(mod, selType="BTLSE", selCrit="BIC")
c3 <- causalTLSE(mod, selType="BTLSE", selCrit="AIC")
texreg(list(SLSE=c1, BTLSE.BIC=c2, BTLSE.AIC=c3), table=FALSE, digits=4)
```

	SLSE	BTLSE.BIC	BTLSE.AIC
ACE	3.9290***	3.9201***	3.9201***
	(0.1703)	(0.1717)	(0.1717)
ACT	3.9552***	3.9404***	3.9404***
	(0.1891)	(0.1904)	(0.1904)
ACN	3.8670***	3.8721***	3.8721***
	(0.2371)	(0.2362)	(0.2362)
Num. knots (Control)	2	1	1
Num. knots (Treated)	3	2	2
Num. covariates	1	1	1
Num. obs. (Control)	89	89	89
Num. obs. (Treated)	211	211	211
R^2	0.7833	0.7829	0.7829
R^2_{adj}	0.7774	0.7784	0.7784

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

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

```
plot(c2, "X", legendPos="right", cex=.8)
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-control"),
       col=c("darkgreen","darkorange"), lty=3, lwd=3, bty='n', cex=.8)
plot(c2, "X", addPoints=TRUE, legendPos="topleft", cex=.8)
```



2.10 A simulated data set from Model 3

In the package, the data set `datSim3` is generated using the following data generating process with a sample size of 300. This model is presented as a case of multiple covariates.

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

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, FTLSE with AIC and FTLSE with BIC.

```
data(simDat3)
mod <- setModel(Y~Z | ~X1+X2, data=simDat3)
```

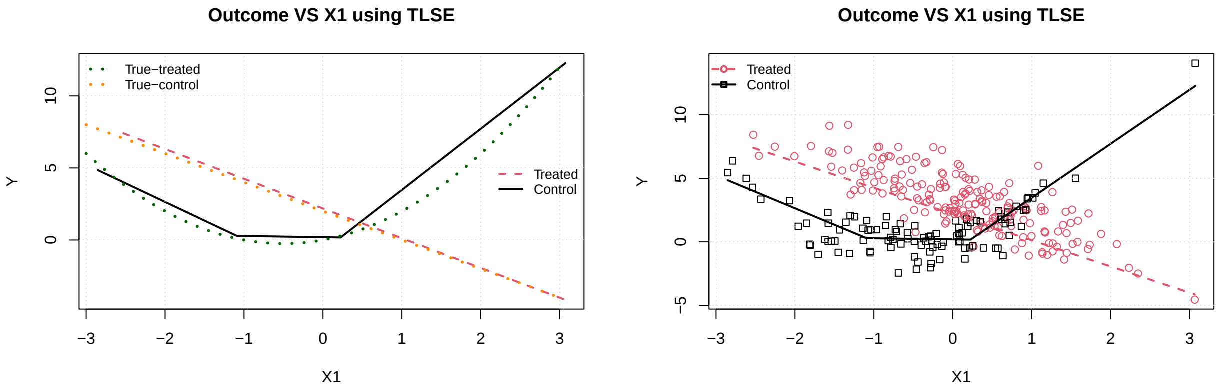
```
c1 <- causalTLSE(mod, selType="SLSE")
c2 <- causalTLSE(mod, selType="FTLSE", selCrit="BIC")
c3 <- causalTLSE(mod, selType="FTLSE", selCrit="AIC")
texreg(list(SLSE=c1, FTLSE.BIC=c2, FTLSE.AIC=c3), table=FALSE, digits=4)
```

	SLSE	FTLSE.BIC	FTLSE.AIC
ACE	2.4699***	2.4866***	2.4725***
	(0.2684)	(0.2675)	(0.2684)
ACT	2.0653***	2.0688***	2.0688***
	(0.3397)	(0.3380)	(0.3402)
ACN	3.2323***	3.2739***	3.2334***
	(0.3445)	(0.3425)	(0.3436)
Num. knots (Control)	6	5	5
Num. knots (Treated)	6	3	4
Num. covariates	2	2	2
Num. obs. (Control)	104	104	104
Num. obs. (Treated)	196	196	196
R^2	0.8630	0.8614	0.8625
R^2_{adj}	0.8547	0.8551	0.8558

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

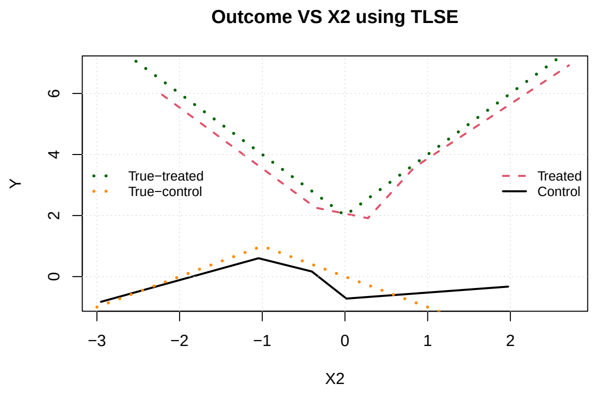
To illustrate the method, since we have two covariates, we need to plot the outcome against one covariate holding the other fixed. The default is to fix it to its sample mean. For the true curve, we fix it to its population mean, which is 0. We first look at the outcome against X_1. By fixing X_2 to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

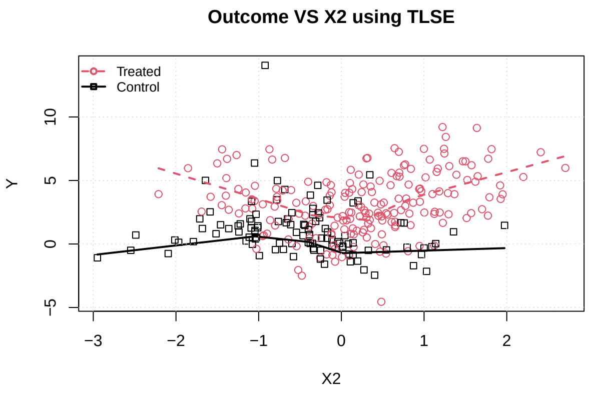
```
plot(c2, "X1", legendPos="right", cex=.8)
curve(x+x^2, -3,3, col="darkgreen", lty=3, lwd=3, add=TRUE)
curve(2-2*x, -3,3, col="darkorange", lty=3, lwd=3, add=TRUE)
legend("topleft", c("True-treated","True-control"),
       col=c("darkgreen","darkorange"), lty=3, lwd=3, bty='n', cex=.8)
plot(c2, "X1", addPoints=TRUE, legendPos="topleft", cex=.8)
```



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 control 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 FTLSE-AIC.

```
plot(c2, "X2", legendPos="right", cex=.8)
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-control"),
       col=c("darkgreen","darkorange"), lty=3, lwd=3, bty='n', cex=.8)
plot(c2, "X2", addPoints=TRUE, legendPos="topleft", cex=.8)
```





2.11 A simulated data set from Model 4

In the package, the data set `datSim5` is generated using the following data generating process with a sample size of 300. This model is presented as a case of multiple covariates with interactions..

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

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, FTLSE with AIC and FTLSE with BIC.

```
data(simDat5)
mod <- setModel(Y~Z | ~X1*X2, data=simDat5)
```

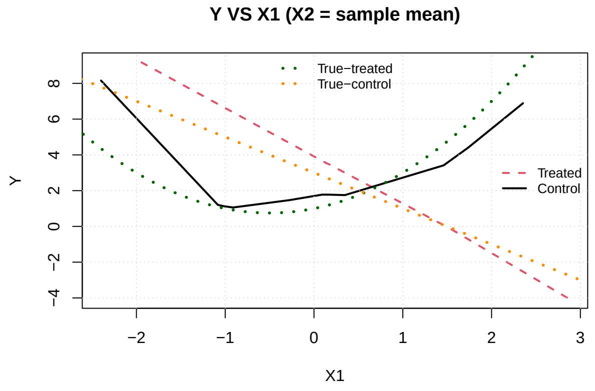
```
c1 <- causalTLSE(mod, selType="SLSE")
c2 <- causalTLSE(mod, selType="BTLSE", selCrit="BIC")
c3 <- causalTLSE(mod, selType="BTLSE", selCrit="AIC")
texreg(list(SLSE=c1, FTLSE.BIC=c2, FTLSE.AIC=c3), table=FALSE, digits=4)
```

	SLSE	FTLSE.BIC	FTLSE.AIC
ACE	1.7990***	1.7797***	1.7744***
	(0.3566)	(0.3615)	(0.3613)
ACT	1.2582**	1.2091*	1.2091*
	(0.4722)	(0.4796)	(0.4803)
ACN	2.8183***	2.8550***	2.8399***
	(0.4402)	(0.4400)	(0.4378)
Num. knots (Control)	9	8	8
Num. knots (Treated)	9	5	6
Num. covariates	3	3	3
Num. obs. (Control)	104	104	104
Num. obs. (Treated)	196	196	196
R^2	0.8909	0.8879	0.8894
R^2_{adj}	0.8809	0.8799	0.8811

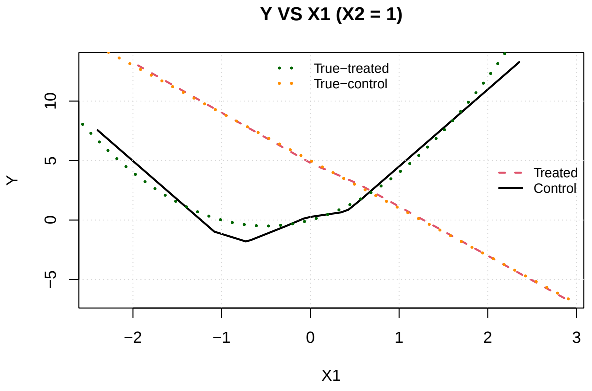
$^{***}p < 0.001$; $^{**}p < 0.01$; $^{*}p < 0.05$

In the case of multiple covariates with interactions, the shape of the fitted outcome with respect to one covariate depends on the value of the other covariates. Without interaction, changing the value of the other covariates 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 its mean (left graph) and 1 (right graph). When X_2 is equal to its population mean, the true curves are $(1+x+x^2)$ for the treated and $(3-2x)$ for the control. If $X_2 = 1$, the true curves become $2x + 2x^2$ for the treated and $(5 - 4x)$ for the control.

```
plot(c2, "X1", legendPos="right", cex=.8,
     main="Y VS X1 (X2 = sample mean)")
curve(1+x+x^2, -3,3, col="darkgreen", lty=3, lwd=3, add=TRUE)
curve(3-2*x, -3,3, col="darkorange", lty=3, lwd=3, add=TRUE)
legend("top", c("True-treated","True-control"),
       col=c("darkgreen","darkorange"), lty=3, lwd=3, bty='n', cex=.8)
plot(c2, "X1", newdata=c(X2=1), legendPos="right", cex=.8,
     main="Y VS X1 (X2 = 1)")
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-control"),
       col=c("darkgreen","darkorange"), lty=3, lwd=3, bty='n', cex=.8)
```



Y VS X1 (X2 = sample mean)

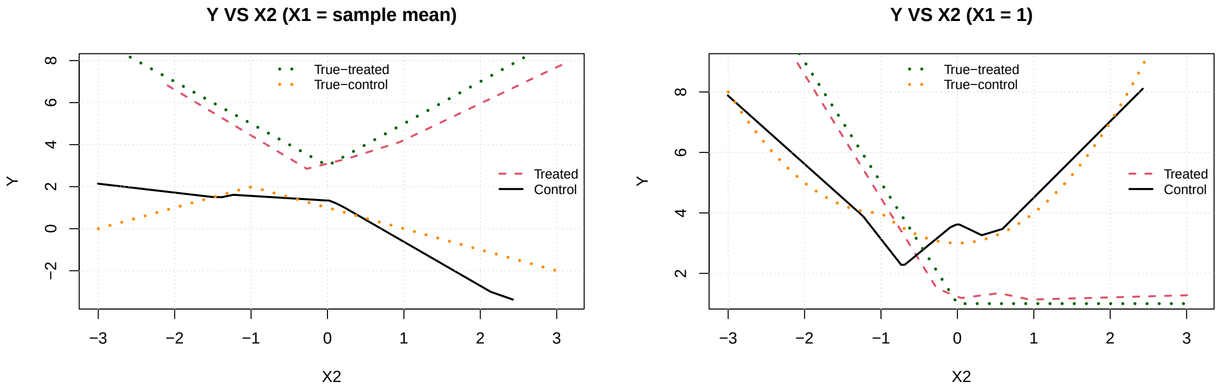


Y VS X1 (X2 = 1)

The following graphs illustrate the relationship between Y and X_2 for a given X_1. When X_1 is equal to its population mean, the true curves are $[2 + (1 - 2x)(x \leq 0) + (1 + 2x)(x > 0)]$ for the treated and $[2 + (1 + x)(x \leq -1) + (-1 - x)(x > -1)]$ for the control. 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 control.

```
plot(c2, "X2", legendPos="right", cex=.8,
     main="Y VS X2 (X1 = sample mean)")
```

```
curve(2+(1-2*x)*(x<=0)+(1+2*x)*(x>0), -3,3,
      col="darkgreen", lty=3, lwd=3, add=TRUE)
curve(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-control"),
       col=c("darkgreen","darkorange"), lty=3, lwd=3, bty='n', cex=.8)
plot(c2, "X2", newdata=c(X1=1), legendPos="right", cex=.8,
     main="Y VS X2 (X1 = 1)")
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-control"),
       col=c("darkgreen","darkorange"), lty=3, lwd=3, bty='n', cex=.8)
```



References

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

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.