# vignette

2022-09-03

`causalsieve` allows for estimation of and inference for a rich class of target parameters in the single time-point treatment setting, specifically targer parameters that are linear in the outcome regression. Notable target parameters include the average treatment effect, the adjusted treatment-specific mean, the average treatment effect among the treated, the dose-response function, the effect of a shift-intervention, working models and marginal structural working models for the conditional average treatment effect.

The package allows the users to specify a possibly very-large and data-adaptive linear regression model for the outcome regression function. Smaller linear regression models will lead to more precise estimates and inference at the cost of estimation bias due to model misspecification. Even when the user-supplied linear model is misspecified, the resulting estimates and inference are correct for target parameter evaluated at the best approximation of the true regression function within the misspecifed regression model. As a consequence, estimates and inference provided by this package are relatively robust to model misspecification, although the estimates need to be interpreted with care.

To characterize the uncertainty of parameter estimates, both confidence intervals based on the asymptotic-normality of the estimates and more robust bootstrap-based confidence intervals are provided.

## The data-structure and target parameter

This package considered the single time-point treatment data-structure `O = (X, A, Y) \sim P` where `X` is a vector of baseline variables, `A` is a discrete or continuous treatment assignment, `Y` is a numeric outcome. Denote the outcome regression function $g(A, X) := E_P[Y \mid A, X]$ and suppose $g$ falls in a linear regression model $\Gamma$.

This package can be used to estimate any target parameter $\theta(P)$ that is linear in the outcome regression function $g(A, X) := E[Y|A, X]$. That is, we consider target parameters that can be decomposed as

$$\theta(P) \equiv \theta(P_{X,A}, g) \text{ where } \theta(P_{X,A}, ag' + bg) = a\theta(P_{X,A}, g') + b\theta(P_{X,A}, g).$$

A subclass of parameters that fall under the above umbrella are as of the form

$$\theta(P) = E_P\left[m_{P_{X,A}}(X, A, g)\right].$$

where $g \mapsto m_{P_{X,A}}(X, A, g)$ be is a linear function-valued mapping. For instance if $A \in \{0, 1\}$, possible choices for \$ m_{P_{X,A}}\$ are \$ m(X,A,g) = g(1,X) - g(0,X)\$, $m(X, A, g) = g(1, X)$, and \$ m_{P_{X,A}}(X,A,g) = A/mean(A) * (g(1,X) - g(0,X))\$.

We will see in this vignette that such target parameters can be specified by supplying a formula expression for \$ m_{P_{X,A}}(X,A,g)\$. This class of parameters includes multiple well-known parameters including the average treatment effect $E_P[g(1, X) - g(0, X)]$, treatment-specific mean $E_P[g(A = a, X)]$ and the average treatment-effect among the treated \$E_P[A / mean(A) * (g(1,X) - g(0,X))].

The above class of parameters capture marginal(ized) treatment-effects. In practice, it may be of interest to learn conditional or subject-specific treatment effects. We can generalize the above class of parameters to handle many such cases including conditional average treatment effect working-models and marginal structural working models. Let $V$ be a subset of baseline variables $X$. Let $\underline{f}(V)$ be a transformation $V$.

As an example, we may have $X = (X_1, X_2, X_3)$, $V = (X_1, X_2)$ and $\underline{f}(V) = c(exp(X_1), X_2^2)$. Consider the conditional target parameter

$$E_P[m_{P_{X,A}}(X, A, g) \mid V].$$

In order to estimate and obtain inference for this parameter, we need to specify a parametric form for $V \mapsto E_P[m_{P_{X,A}}(X, A, g) \mid V]$. Rather than assume this parametric form is correct, we instead use it as a marginal structural working model and estimate the best approximation of the true conditional target parameter within the working model. Formally, this package provides estimates and inference for marginal structural coefficient parameter of the form

$$\beta_P = \text{argmin}_\beta \, E_P \left\{ m_{P_{X,A}}(X, A, g) - \beta^T \underline{f}(V) \right\}^2,$$

which can equivalently be written as

$$\beta_P = \text{argmin}_\beta \, E_P \left\{ E_P[m_{P_{X,A}}(X, A, g) \mid V] - \beta^T \underline{f}(V) \right\}^2.$$

In other words, $\beta_P$ are the coefficients of the linear regression of $m_{P_{X,A}}(X, A, g)$ onto the transformed variables $\underline{f}(V)$. We can conveniently specify the parameter in a formulaic manner as

$$m_{P_{X,A}}(X, A, g) \sim \text{formula}$$

where `formula` is a formula expression that utilizes the baseline variables $X$. One notable specification is the intercept model for the CATE

$$g(1, X) - g(0, X) \sim 1,$$

which provides nonparametric estimates of the ATE $E_P[g(1, X) - g(0, X)]$. More generally, intercept models will recover the first class of estimands we considered. More generally we can consider

$$g(1, X) - g(0, X) \sim 1 + X_1$$

which models the CATE as linear in the first baseline variable $X_1$ and is also equivalent to the marginal structural working model $E[g(1, X) - g(0, X) \mid X_1] \sim 1 + X_1$.

Let us now see how to use causalsieve in practice.

## Using causalsieve

Let us generate a dataset consisting of baseline variable `X`, a binary treatment `A`, and a continuous outcome `Y`.

```r
library(autocausalML)
n <- 250
d <- 3
X <- replicate(d, runif(n, -1 , 1))
A <- rbinom(n, 1, plogis(rowMeans(X)))
Y <- rnorm(n, rowMeans(X) + A + A*X[,1])
```

To begin estimating treatment-effects using this package, we need to create a causalsieve object, which is an R6 class. The causalsieve object has four required arguments:

- ‘X‘: A matrix with the ith row and jth column being the observed value of the jth baseline variable for the ith observation.
- ‘A‘: A numeric vector with the observed values of a binary, numerically-encoded categorical, or continuous treatment variable.

- ‘Y‘: A numeric vector with the observed outcome values.
- ‘g_basis_generator‘: A function containing named arguments ‘X‘ and ‘A‘ that outputs a (n x k) matrix contained the evaluation of k basis functions for a linear-model/sieve for the outcome regression function ‘g(A,X) := E[Y | A, X]‘ evaluated at ‘X‘ and ‘A‘.

The package comes with built-in functions that output a data-adaptively chosen `g\_basis\_generator` function using either LASSO (glmnet) or Highly Adaptive Lasso (HAL). The function `make\_g_basis\_generator_LASSO` takes in the arguments `X`, `A` and `Y` and a formula object that specifies a linear regression model for the outcome regression function `g(A,X)`. LASSO is used to screen out unimportant variables, which is useful when `X` is high-dimensional. The function `make\_g\_basis\_generator_HAL` takes in the arguments `X`, `A` and `Y` as well as arguments to be passed to the `fit_hal` functions of the `hal9001` R package. `make\_g\_basis\_generator\_HAL` can be viewed as a nonparametric highly flexible version of `make\_g\_basis\_generator\_LASSO` where a high dimensional linear-model is generated for the outcome regression from the linear span of spline basis functions. A sparse set of spline basis functions is selected using the multivariate total variation penalty (which is equivalent in optimization to using the LASSO penalty).

Lets model the outcome regression as linear in the baseline variables and include all baseline variable - treatment interactions.

```
g_basis_gen_LASSO <- make_g_basis_generator_LASSO(X,A,Y, formula = ~. + A*.)


as.data.frame(head(g_basis_gen_LASSO(X,A)))
```

```
## Warning in terms.formula(object, data = data): 'varlist' has changed (from
## nvar=4) to new 5 after EncodeVars() -- should no longer happen!

## Warning in terms.formula(formula, data = data): 'varlist' has changed (from
## nvar=4) to new 5 after EncodeVars() -- should no longer happen!

##            V2          V3         V4 A X:A
## 1  0.73569574 -0.92870315 -0.9549175 0   0
## 2 -0.95779577  0.01447715  0.9783894 1   1
## 3  0.06874779  0.94985528  0.1755057 0   0
## 4 -0.47738183 -0.53881087  0.6443533 0   0
## 5 -0.61435045 -0.25893337  0.2804360 1   1
## 6  0.13017345  0.78994827  0.6968607 0   0
```

We can use a more flexible regression model with HAL. Let’s model the outcome regression as the sum of an additive function in the baseline variables and bi-additive in the interaction between treatment and the baseline-variable. This model can be specified using the `formula_hal` argument with `formula_hal = ~ h(.) + h(., A)`. We specify first-order (linear) splines by add the argument `s = 1` to formula_hal as `formula_hal = ~ h(., s=1) + h(., A,s=1)`. To generate the main-term basis functions from 10 knot points and the two-way treatment interaction basis functions from 5 knot points, we can add the `k` argument to formula_hal as `formula_hal = ~ h(., s=1, k= 10) + h(., A, s=1, k= 5)`. Alternatively, we can pass the arguments `smoothness_orders = 1` and `num_knots = c(10,5)` directly to `make_g_basis_generator_HAL`. If you do not want to screen the spline basis functions using HAL then you can set `screen_basis = FALSE` (note overly correlated basis functions are still removed).

```
g_basis_gen_HAL <- make_g_basis_generator_HAL(X,A,Y, formula_hal = ~ h(., s=1, k= 10) + h(., A, s=1, k=
```

```
## Loading required package: Rcpp
```

```
## hal9001 v0.4.3: The Scalable Highly Adaptive Lasso
## note: fit_hal defaults have changed. See ?fit_hal for details
```

```
as.data.frame(head(g_basis_gen_HAL(X,A)))
```

```
##   V1        V2       V3        V4
## 1  1 0.9965758 0.769625 0.0000000
## 2  1 1.9965758 1.769625 0.9783894
## 3  1 0.9965758 0.769625 0.1755057
## 4  1 0.9965758 0.769625 0.6443533
## 5  1 1.9965758 1.769625 0.2804360
## 6  1 0.9965758 0.769625 0.6968607
```

```
g_basis_gen_HAL <- make_g_basis_generator_HAL(X,A,Y, formula_hal = ~ h(., s=1, k= 10) + h(., A, s=1, k=
as.data.frame(head(g_basis_gen_HAL(X,A)))
```

```
##   V1        V2       V3        V4        V5        V6         V7       V8
## 1  1 0.9965758 0.769625 0.5904338 0.4244636 0.3013918 0.08376121 0.000000
## 2  1 1.9965758 1.769625 1.5904338 1.4244636 1.3013918 1.08376121 0.924386
## 3  1 0.9965758 0.769625 0.5904338 0.4244636 0.3013918 0.08376121 0.000000
## 4  1 0.9965758 0.769625 0.5904338 0.4244636 0.3013918 0.08376121 0.000000
## 5  1 1.9965758 1.769625 1.5904338 1.4244636 1.3013918 1.08376121 0.924386
## 6  1 0.9965758 0.769625 0.5904338 0.4244636 0.3013918 0.08376121 0.000000
##          V9       V10       V11       V12
## 1 0.0000000 0.0000000 0.0000000 0.0000000
## 2 0.7284856 0.4525605 0.2570787 0.9783894
## 3 0.0000000 0.0000000 0.0000000 0.1755057
## 4 0.0000000 0.0000000 0.0000000 0.6443533
## 5 0.7284856 0.4525605 0.2570787 0.2804360
## 6 0.0000000 0.0000000 0.0000000 0.6968607
```

We are finally ready to run some causal analysis using the causalsieve package.

```
# Make g_basis_generator using LASSO
g_basis_gen_LASSO <- make_g_basis_generator_LASSO(X,A,Y, formula = ~. + A*.)
# Initialize causalsieve object
causal_sieve <- causalsieve$new(X, A, Y, g_basis_gen_LASSO)
```

Once we have created a causalsieve object, we need to tell it what parameters/estimands we would like to learn and obtain inference for. This is done using the `add_target_parameter` function. To specify the target parameter we would like to add, `add_target_parameter` requires only one key argument

- ‘formula‘: a formula object of the form $m(X, A, g) \sim$ <expression in X >] specifying a linear regression of a conditional target parameter onto a working parametric model.

Let what formulas we should specify for a number of popular target parameters:

1. ATE: $E_P[g(A=1, X=X) - g(A=0, X=X)$ do $formula = g(A=1, X=X) - g(A=0, X=X) \sim 1$.
2. Treatment-specific mean: $E_P[g(A=a, X=X)]$ do $formula = g(A=a, X=X) \sim 1$.
3. ATT: $E_P[(A/P(A=1)) * \{g(A=1, X=X) - g(A=0, X=X)\}]$ do $formula = (A/mean(A)) * (g(A=1, X=X) - g(A=0, X=X)) \sim 1$
4. Working model for CATE: $g(A=1, X=X) - g(A=0, X=X)$ do $formula = g(A=1, X=X) - g(A=0, X=X) \sim 1 + X_1 + X_2$.
5. Working model for CATT: $g(A=1, X=X) - g(A=0, X=X)$ do $formula = (A/mean(A)) * (g(A=1, X=X) - g(A=0, X=X)) \sim 1 + X_1 + X_2$.
6. Shift-intervention: Let $\delta \in \mathbb{R}$, $E_P[g(A+delta, X)]$ do $formula = g(A+delta, X) \sim 1$
7.

Let us add the ATE, ATT and TSM target parameters.

```r
# ATE
causal_sieve$add_target_parameter(
  formula = g(A=1, X) - g(A=0, X) ~ 1, name = "ATE"
)

# ATT
causal_sieve$add_target_parameter(
  formula = A / mean(A) * (g(A=1, X) - g(A=0, X)) ~ 1, name = "ATT"
)

# TSM
causal_sieve$add_target_parameter(
  formula =  g(A=1, X)   ~ 1, name = "TSM_A1"
)
causal_sieve$add_target_parameter(
  formula =  g(A=0, X)   ~ 1 , name = "TSM_A0"
)
```

We can see what parameters have been added by taking a look at the `target_parameters` attribute of our `causalsieve` object. `target_parameters` is a list that contains for each target parameter information that is required for estimation.

```r
names(causal_sieve$target_parameters)
```

```
## [1] "ATE"    "ATT"    "TSM_A1" "TSM_A0"
```

Once we have added all our target parameters, we can obtain estimates and inference using the `estimate` function. The `estimates` attribute of the causalsieve object will contain parameter estimates, standard errors, confidence intervals among other relevant information. The `summarize` function can be used to provide a more user friendly summary of the parameter estimates.

```r
causal_sieve$estimate()
estimates <- causal_sieve$estimates

# parameter estimate
estimates$ATE$estimate
```

```
##           [,1]
## [1,] 0.8761927
```

```
# parameter confidence interval using asymptotic analysis
estimates$ATE$CI
```

```
##           [,1]     [,2]
## [1,] 0.6865431 1.065842
```

```
# parameter confidence interval using bootstrap
estimates$ATE$CI_boot
```

```
##           [,1]     [,2]
## [1,] 0.7050825 1.047303
```

```
# Summary
causal_sieve$summary(ndigits = 3)
```

```
##              Param Estimate     se               CI se_boot          CI_boot
## 1    ATE_intercept  0.87600 0.0968     (0.687,1.07) 0.08730    (0.705,1.05)
## 2    ATT_intercept  0.94700 0.1210     (0.711,1.18) 0.08720    (0.776,1.12)
## 3 TSM_A1_intercept  0.87300 0.0980     (0.681,1.06) 0.08700    (0.702,1.04)
## 4 TSM_A0_intercept -0.00338 0.0177 (-0.0381,0.0314) 0.00693 (-0.017,0.0102)
```

## More target parameters

### Binary treatment effects

```
n <- 250
d <- 3
X <- replicate(d, runif(n, -1 , 1))
colnames(X) <- paste0("X", 1:d)
A <- rbinom(n, 1, plogis(rowMeans(X)))
Y <- rnorm(n, rowMeans(X) + A + A*X[,1])


# Make g_basis_generator using LASSO
g_basis_gen_LASSO <- make_g_basis_generator_LASSO(X,A,Y, formula = ~. + A*.)
# Initialize causalsieve object
causal_sieve <- causalsieve$new(X, A, Y, g_basis_gen_LASSO)

# ATE
causal_sieve$add_target_parameter(g(A=1,X=X) - g(A=0,X=X) ~ 1, name = "ATE")
# CATE working model
causal_sieve$add_target_parameter(g(A=1,X) - g(A=0,X) ~ 1 + X1, name = "CATE~1+X1")
# CATE additive working model
causal_sieve$add_target_parameter(g(A=1,X) - g(A=0,X) ~ ., name = "CATE~.")

causal_sieve$estimate()
causal_sieve$summary()
```

```
##                  Param    Estimate         se                       CI
## 1       ATE_intercept  9.0516e-01 8.6895e-02         (0.73485,1.0755)
## 2 CATE~1+X1_intercept  9.0904e-01 8.4809e-02         (0.74282,1.0753)
## 3        CATE~1+X1_X1  5.4092e-01 2.3712e-01        (0.076172,1.0057)
## 4    CATE~._intercept  9.0904e-01 8.4809e-02         (0.74282,1.0753)
## 5           CATE~._X1  5.4092e-01 2.3712e-01        (0.076172,1.0057)
## 6           CATE~._X2 -1.0665e-16 1.5857e-16 (-4.1745e-16,2.0414e-16)
## 7           CATE~._X3  2.2273e-16 1.6298e-16 (-9.6704e-17,5.4216e-16)
##      se_boot                  CI_boot
## 1 8.7463e-02          (0.73374,1.0766)
## 2 8.7299e-02          (0.73794,1.0801)
## 3 2.3446e-01         (0.081386,1.0005)
## 4 8.7299e-02          (0.73794,1.0801)
## 5 2.3446e-01         (0.081386,1.0005)
## 6 3.5393e-17 (-1.7602e-16,-3.7285e-17)
## 7 6.9794e-17   (8.5934e-17,3.5952e-16)
```

## Continuous treatment effects

```r
n <- 250
d <- 3
X <- replicate(d, runif(n, -1 , 1))
colnames(X) <- paste0("X", 1:d)
A <- rnorm(n, rowMeans(X), 0.5)
A <- A - min(A)
Y <- rnorm(n, rowMeans(X) + A + A*X[,1])


# Make g_basis_generator using LASSO
g_basis_gen_LASSO <- make_g_basis_generator_LASSO(X,A,Y, formula = ~. + A*.)
# Initialize causalsieve object
causal_sieve <- causalsieve$new(X, A, Y, g_basis_gen_LASSO)

# dose-response at A=0.5
causal_sieve$add_target_parameter(g(A=0, X) ~ 1)
causal_sieve$add_target_parameter(g(A=0.5, X) ~ 1)
# CATE dose-response + conditional working model
causal_sieve$add_target_parameter(g(A=0.5, X) - g(A=0,X) ~ 1)
causal_sieve$add_target_parameter(g(A=0.5, X) - g(A=0,X) ~ 1 + X1)



# stochastic shift intervention (Nima Hejazi, Mark van der Laan et al. (2020))
# https://arxiv.org/abs/2003.13771
causal_sieve$add_target_parameter(g(A = A + 0, X)~ 1)
causal_sieve$add_target_parameter(g(A = A + 0.5, X)~ 1)
causal_sieve$add_target_parameter(g(A = A + 1, X)~ 1)

causal_sieve$estimate()
causal_sieve$summary()
```

```
##                                             Param  Estimate       se
## 1                      g(A = 0, X) ~ 1_intercept -0.015637 0.018199
## 2                    g(A = 0.5, X) ~ 1_intercept  0.492080 0.040725
## 3      g(A = 0.5, X) - g(A = 0, X) ~ 1_intercept  0.507720 0.032100
## 4 g(A = 0.5, X) - g(A = 0, X) ~ 1 + X1_intercept  0.519780 0.024342
## 5        g(A = 0.5, X) - g(A = 0, X) ~ 1 + X1_X1  0.532400 0.094804
## 6                  g(A = A + 0, X) ~ 1_intercept  1.400500 0.105150
## 7                g(A = A + 0.5, X) ~ 1_intercept  1.908200 0.133320
## 8                  g(A = A + 1, X) ~ 1_intercept  2.416000 0.162950
##                     CI   se_boot               CI_boot
## 1 (-0.051307,0.020032) 0.0065269 (-0.02843,-0.0028449)
## 2     (0.41226,0.5719) 0.0244700     (0.44412,0.54004)
## 3     (0.4448,0.57063) 0.0251880     (0.45835,0.55708)
## 4    (0.47207,0.56749) 0.0243010     (0.47215,0.56741)
## 5    (0.34659,0.71822) 0.0994580     (0.33747,0.72734)
## 6      (1.1944,1.6066) 0.0591980       (1.2845,1.5166)
## 7      (1.6469,2.1695) 0.0828090       (1.7459,2.0705)
## 8      (2.0966,2.7353) 0.1071400         (2.206,2.626)
```