Vignette: causal machine learning with `hte3`

2023-08-06

hte3: causal machine learning of heterogeneous treatment effects

Configuring an `hte3` learning task and nuisance function estimation

Generate point-treatment data structure

Suppose we conduct an observational or experimental study where we observe baseline covariates `W`, a treatment `A`, and an outcome `Y`. We can generate such a dataset as follows:

```
library(data.table)
n <- 1000
W1 <- runif(n, -1 ,1)
W2 <- runif(n, -1 ,1)
A <- rbinom(n, size = 1, prob = plogis(W1 + W2))
Y <- rbinom(n, size = 1, prob = plogis(W1 + W2 + A * (1 + W1 + W2)))
cate <- plogis(W1 + W2 + (1 + W1 + W2)) - plogis(W1 + W2)

data <- data.table(W1, W2, A, Y)
data$id <- 1:nrow(data) # Specify ID variable
```

In `hte3`, R and DR-type causal learners utilize estimates of the following nuisance functions:

- Outcome Regression: `mu(A=a,W=w) := E[Y | A = a, W = w]` (DR-only)
- Propensity Score: `pi(A=a | W=w) := P(A = a | W = w)` (DR-only)
- Conditional Mean of Outcome: `m(W=w) := E[Y | W = w]` (R-only)
- Conditional Mean of Treatment: `e(W=w) := E[A | W = w]` (R-only)

`hte3_Task` objects and nuisance function estimation with `sl3`

To perform training and prediction for `hte3` learners, including both DR-learners and R-learners, you need to specify an `hte3_Task`. This R6 object inherits attributes from `sl3_Task` and `tmle3_Task` in the `sl3` and `tmle3` packages, respectively. The `hte3_Task` object contains relevant training data, encodes causal relationships between variables, and stores estimates of nuisance functions.

Formally, an `hte3_Task` is an `sl3_Task` with additional attributes: a list `npsem` specifying causal structure and a `tmle3/Likelihood` object for nuisance estimates. An `hte3_task` can be conveniently instantiated using the `make_hte3_Task_tx` constructor function.

The nuisance function estimates can be externally provided by the user or estimated internally using the `sl3` ensemble learning pipeline. If you do not provide specific nuisance estimates or learner arguments, the default behavior is to utilize the `sl3` library's `auto-machine learning` approach to estimate all the nuisance functions. This approach employs an ensemble of learners, including lasso (via glmnet), generalized additive models (via mgcv), multivariate adaptive regression splines (via earth), random forests (via ranger), and gradient boosted trees with varying maximum tree depths (via xgboost).

Here's how to create an `hte3_Task` for the point-treatment data structure using minimal arguments:

```
# Specify variable names
modifiers <- c("W1", "W2") # or modifiers <-  c("W1")
confounders <- c("W1", "W2")
treatment <- "A"
outcome <- "Y"

# Create the task with auto-machine learning for nuisance functions
hte3_task <- make_hte3_Task_tx(data, modifiers, confounders, treatment, outcome)
```

The `get_tmle_node` and `get_nuisance_estimates` functions can be employed to extract data and nuisance function estimates.

```
# get specific variables
head(data.table(modifiers = hte3_task$get_tmle_node("modifiers"),
           confounders = hte3_task$get_tmle_node("confounders")))

# pi and mu provide a matrix of estimates for each treatment level.
 data.table(pi = hte3_task$get_nuisance_estimates("pi"),
           mu = hte3_task$get_nuisance_estimates("mu"),
          e = hte3_task$get_nuisance_estimates("e"),
          m = hte3_task$get_nuisance_estimates("m"))
```

Now, let's explore how to define learners for nuisance functions using the `sl3` machine learning pipeline. You can specify `sl3_Learner` objects for the corresponding `learner_<nuisance>` arguments.

If you're dealing with a categorical numeric treatment variable, make sure to use a binomial learner for `learner_pi`. This is because it will be automatically converted into a categorical learner using the `Lrnr_independent_binomial` wrapper. To disable this automatic conversion, you can set `multinomialize_pi = TRUE`.

```
# Specify learners for nuisance functions
# Note learners are internally converted to cross-fitted learners by default.
# To disable this, set `cross_fit_and_cv = FALSE`.
hte3_task <- make_hte3_Task_tx(data, modifiers, confounders, treatment, outcome,
                               learner_pi = Lrnr_glmnet$new(family = "binomial"),
                               learner_mu = Lrnr_gam$new(family = "binomial"),
                               learner_m = Lrnr_ranger$new(),
                               learner_e = NULL)
```

To integrate with other learning frameworks, `make_hte3_Task_tx` supports user-specified nuisance estimates through `<nuisance>.hat` arguments.

```
# Specify user-specified nuisance estimates
pi.hat <- cbind(estimates$pi.0, estimates$pi.1)
mu.hat <- cbind(estimates$mu.0, estimates$mu.1)

hte3_task <- make_hte3_Task_tx(data, modifiers, confounders, treatment, outcome,
                               pi.hat = pi.hat,
                               mu.hat = mu.hat,
```

```
                         e.hat = estimates$pi.1,
                         m.hat = estimates$m)
```

With an `hte3_Task` object containing data and nuisance estimates, you can now leverage the `hte3` causal learning framework for predicting heterogeneous treatment effects.

`hte3` meta-learners for heterogeneous treatment effect estimation

conditional average treatment effect (CATE) estimation with `hte3` meta-learners

Building on top of the `sl3` framework, the `hte3` package a range of specialized meta-learners for estimation of heterogeneous treatment effects (HTEs). Formally, `hte3` meta-learners are learner objects that inherit from the template learner class `Lrnr_hte`, which itself inherits from the `Lrnr_base` object of `sl3`.

Through the `learner` argument, you can specify the supervised learning algorithm utilized by the meta-learner. This argument accepts any sl3 learner compatible with the meta-learning algorithm. When the meta-learner is trained, the `learner` is fit on an `sl3_Task` object with covariates being the effect `modifiers`, weights being method-specific `pseudo-weights`, and outcome being method-specific `pseudo-outcomes`.

By convention, meta-learners for estimating the conditional average treatment effect (CATE) have names of the form `Lrnr_cate_method`, where `method` encodes the meta-learning strategy used for CATE estimation. `hte3` currently supports the following meta-learners: - DR-learner () - R-learner (Wager) - T-learner

For an overview of the DR-learner and R-learner algorithms, see ??? and ???, respectivelly. For other examples of meta-learners, see ???.

When `modifiers` are equal to `confounders`, both the DR-learner and R-learner estimate the CATE function `E(Y_1 - Y_0 | confounders)`. However, if `modifiers` are different from `confounders`, the R-learner and DR-learner typically estimate different measures of effect modification. Specifically, the DR-learner provides estimates of the marginal CATE `E(Y_1 - Y_0 | modifiers)`, which marginalizes over all `confounders` not included in `modifiers`. In contrast, the R-learner provides estimates of the `e(W)(1-e(W))`-weighted projection of the CATE `E(Y_1 - Y_0 | confounders)` onto functions of the effect `modifiers`. Consequently, when `modifiers != confounders`, the predictions of the R-learner may be more difficult to interpret. For further details on the differences between R-learners and DR-learners, see ???.

Here's a practical illustration of how to train a CATE DR-learner using generalized additive models (GAMs) as the chosen supervised learning algorithm:

```
# Specify a CATE DR-learner with generalized additive models (GAMs) as the supervised learning algorith
lrnr_cate_dr <- Lrnr_cate_DR$new(learner = Lrnr_gam$new(family = gaussian()))

# Train the CATE DR-learner on the hte3_task
trained_lrnr_cate_dr <- lrnr_cate_dr$train(hte3_task)

# Predict using the trained CATE DR-learner on the hte3_task
predictions <- trained_lrnr_cate_dr$predict(hte3_task)

# Calculate the correlation between predicted CATE and true CATE
correlation <- cor(predictions, cate)

# Plot the predictions vs. the true CATE values
plot(predictions, cate)
```

conditional relative average treatment effect (CRATE) estimation with `hte3` meta-learners

References