Methodological Details

Drew Dimmery (ddimmery@univie.ac.at)   Edward Kennedy (edward@stat.cmu.edu)

July 27, 2023

Abstract

In this paper, we introduce the `tidyhte` package for estimation of heterogeneous treatment effects (HTE) from observational or experimental data. This package implements the methods of Kennedy (2020) and presents them through a tidy-style user-facing API. The design principles undergirding this package are (1) the APIs should be tidy-friendly, (2) Analyses should be easy to replicate with minor changes, (3) specifying complex ensembles for the nuisance functions should be straightforward, and (4) sensible diagnostics should be easily accessible. Plotting and formatting of the results are left for the end-user to customize.

Summary

This document details how `tidyhte` constructs estimates of heterogeneous treatment effects. It will highlight a variety of the features of the package and discuss the mathematics which undergird them.

After a brief introduction to the methods of HTE estimation of Kennedy (2020), the structure will generally follow the estimation API of `tidyhte`: it will begin by discussing the creation of cross-validation folds, then highlight nuisance function estimation, proceed to the construction of "pseudo outcomes", a concept from Kennedy (2020), and conclude by demonstrating the calculation of a few varieties of Quantities of Interest: the actual statistics which are desired by the end-user.

Preliminaries

Problem Setting

Our data is defined by the triple $Z_i = (X_i, A_i, Y_i)$, with $X \in \mathbb{R}^d$, $Y \in \mathbb{R}$ and $A \in \{0, 1\}$. Define the following nuisance functions:

$$\pi(x) \equiv \mathbb{P}(A = 1 \mid X = x)\mu_a(x) \equiv \mathbb{E}(Y \mid X = x, A = a)$$

Heterogeneous treatment effects are defined as the difference in conditional expectations under treatment and control, $\tau(x) \equiv \mu_1(x) - \mu_0(x)$. Throughout, we will maintain the following assumptions:

1. **Consistency**: $Y_i = Y_i(A_i)$
2. **No Unmeasured Confounding**: $A \perp (Y(1), Y(0)) \mid X$
3. **Positivity**: $\epsilon \leq \pi \leq 1 - \epsilon$ with probability 1

Under these assumptions, $\tau(x) = \mathbb{E}[Y(1) - Y(0) \mid X = x]$.

For the remainder of this paper, we will use a semi-simulated running example based on the `penguins` dataset of Horst, Hill, and Gorman (2020). We will imagine a randomly assigned nutritional intervention for which we wish to measure the causal effect on body mass. We also have a lower variance measurement of the average food consumed per day over an observation period. Gentoo penguins gain weight by this intervention on average, while vice-versa for Adelie penguins. The average change in weight for Chinstrap penguins is zero.

```r
penguins <- within(penguins, {
  id <- 1:n
  propensity_score <- 0.5
  treatment <- rbinom(n, 1, propensity_score)
  tau <- 0.2 * (species == "Gentoo") - 0.2 * (species == "Adelie") + rnorm(n, sd = 0.05)
  food_consumed_g <- rnorm(n, 500, 5) * (1 + tau * treatment)
  body_mass_g <- body_mass_g * (1 + tau * treatment)
})
```

Overview of Method

We begin by introducing the following "DR-Learner" algorithm:

DR-learner algorithm. This consists of three main steps:

1. **Given**: estimates of nuisance functions trained elsewhere: $(\hat{\pi}, \hat{\mu}_0, \hat{\mu}_1)$
2. **Construct "pseudo-outcome**: This quantity is a transformation of the provided nuisance function estimators.
$$\hat{\psi}(Z) = \frac{A - \hat{\pi}(X)}{\hat{\pi}(X)(1 - \hat{\pi}(X))}\left(Y - \hat{\mu}_A(X)\right) + \hat{\mu}_1(X) - \hat{\mu}_0(X)$$
3. **Second-stage regression**: Construct a smoothing model over the transformed pseudo-outcome.
$$\hat{\tau}_{dr}(x) = \hat{\mathbb{E}}[\hat{\psi}(Z) \mid X = x]$$

This algorithm is written assuming estimates of nuisance functions are trained on separate data and, therefore, can be treated as fixed. The easiest way to do this is with a sample-splitting procedure, after which results are averaged across splits.

The crucial result, Theorem 2 of Kennedy (2020), shows that the error of the second stage regression will match the error of an oracle regression of the true individual treatment effects on covariates X up to a factor which depends on the product of the errors of the two estimated nuisance functions (that is, the errors in π and in μ).

When the second-stage regression is simple, like subgroup averages within cells defined by X, the DR-Learner inherits unbiasedness and efficiency like an AIPW estimator (Robins, Rotnitzky, and Zhao 1995; Van der Laan, Laan, and Robins 2003; Tsiatis 2006; Tsiatis et al. 2008; Chernozhukov et al. 2018).

The approximation results of Kennedy (2020) applied to regressions on this transformed outcome allow for a lot of flexibility in quantities of interest that may be estimated. We will use this fact to allow the estimation of a variety of models *as if* they were estimated on the individual causal effects themselves: for example, variable importance measures operate off the assumption that various second-stage regressions will be accurate.

Principles

The `tidyhte` package is premised on the idea of breaking up the analysis of HTEs into a few distinct parts, which can then be mixed together as desired. The first step is to define a configuration object (or recipe) describing at a high-level how HTE estimation should be performed. During estimation, the specific variables of interest are indicated. This design allows for repeating very similar analyses multiple times with very little overhead. Instances when this might be useful are when the user wishes to explore heterogeneity across a variety of outcomes, or when there are a variety of treatment contrasts of particular interest. Each of these analyses will tend to share common features: similar classes of models will be included in the ensembles for nuisance estimation, for example.

Also important is that the methods provided support the usage of common

Clustered data

A common feature of real-world data is that treatment may be clustered. In other words, if one unit receives treatment, there may be other units who are then also more likely to receive treatment. A common example of this sort of design might be when villages or townships are assigned to treatment, but measurement occurs at the individual level. As discussed by Abadie et al. (2023), this design implies that standard errors should be clustered at the level at which treatment was assigned. The `tidyhte` package supports clustering as a first class citizen, and resulting estimates will all receive the proper statistical uncertainty estimates based around this clustering. In practice, the end-user simply specifies the individual unit-id when constructing cross-validation splits (`make_splits()`), and all subsequent analyses take this into account.

Population weights

Another common feature of designs used in practice is that they come from samples that do not perfectly represent the larger populations from which they are drawn. The most common solution to this problem is through the use of weights to make the sample more closely resemble the population. This, too, is supported by `tidyhte`, by simply specifying weights when models are estimated (`produce_plugin_estimates()`). Downstream analyses will then take weights into account appropriately.

Recipe API

In order to build up definitions around how HTE should be estimated, `tidyhte` provides an API to progressively build up a configuration object. A `basic_config` function creates a bare-bones configuration consisting of only linear models for the respective nuisance functions and a number of diagnostics.

```r
cfg <- basic_config() %>%
    add_known_propensity_score("propensity_score") %>%
    add_outcome_model("SL.glmnet", alpha = c(0.0, 1.0)) %>%
    add_moderator("Stratified", species, island, sex, year) %>%
    add_moderator("KernelSmooth", bill_length_mm, bill_depth_mm, flipper_length_mm) %>%
    add_vimp(sample_splitting = FALSE)
```

Since the subject of interest is an experimental intervention, the propensity score is known. Using this known propensity score provides unbiasedness for many quantities of interest (although this may leave some efficiency on the table CITE). In this case, in addition to the (default) linear model included in the SuperLearner ensemble, we add an elastic-net regression (Zou and Hastie 2005). We sweep over a variety of mixing parameters between LASSO and ridge, throwing each of these models into the ensemble. This means that SuperLearner will perform model selection and averaging to identify the best hyper-parameter values (Van der Laan, Polley, and Hubbard 2007). Furthermore, we define all of the moderators of interest and how their results should be collected and displayed. Discrete moderators will just take stratified averages at each level of the moderator, while continuous moderators will use local polynomial regression (Fan and Gijbels 2018; Calonico, Cattaneo, and Farrell 2019). Finally, we add a variable importance measure from Williamson et al. (2021).

After the configuration is completed, it can be attached to the dataset.

```r
penguins %<>% attach_config(cfg)
```

Cross-validation

The first step in an analysis of heterogeneous treatment effects following this procedure is to define how to construct splits to be used for cross-validation. `tidyhte` accomplishes this by using blocking methods to construct lower-variance splits than a purely randomized splitting procedure would entail. Existing methods for generating splits often provide options for stratifying based on a binary outcome to ensure there is variance in the outcome in all splits, even when the outcome is very sparse. For instance, `SuperLearner::SuperLearner.CV.control` provides such an option.

The appropriate function in `tidyhte` is `make_splits`. This function takes in a dataframe and determines a set of splits which accord with the provided unit identifier and number of splits. If any covariates are provided to this function, it will include them in a blocking design for constructing randomized splits using the methods of Higgins, Sävje, and Sekhon (2016) as implemented in the `quickblock` package.

There are a few relevant methodological notes about how this blocking for cross-validation strata works. In short, blocks are constructed which are sized to be at least as large as the number of splits to use. These blocks are constructed to minimize the within-block distance between units (where distances are Euclidean based on the provided covariates). When more than one row shares an identifier, blocking is performed on the average covariate value within each identifier. Rows with the same identifier are then assigned to the same split. Precise details on the construction of blocks may be found in `higgins2016improving`. Within each block, a vector of split IDs is constructed which has a marginal distribution as close to the uniform distribution over splits as possible (up to integer division errors). This set of split IDs is then randomly permuted within blocks.

Consistent with tidy semantics, the original dataframe is returned, but with the addition of a column `.split_id` representing these newly constructed splits, and a few attributes for bookkeeping (e.g. the column name of the identifier). Since the object returned is the same as what was passed in, this makes for easy chaining of commands using `dplyr`.

```r
penguins %<>% make_splits(id, species, sex, flipper_length_mm, .num_splits = 3)
```

```
## `num_splits` must be even if VIMP is requested as a QoI. Rounding up.
```

```
## Dropped 11 of 344 rows (3.2%) through listwise deletion.
```

Note that `tidyhte` gracefully handles missing data via listwise deletion. More advanced imputation methods are not yet supported.

Nuisance function estimation

Estimation of nuisance functions such as the propensity score and the outcome regression are typically handled by the SuperLearner library. Specifying a full array of models with diverse hyperparameters is much simplified through the `tidyhte` API. To specify a cross-validated learner using SuperLearner syntax requires substantially more boilerplate:

```r
learners <- create.Learner(
    "SL.glmnet",
    tune = list(
        alpha = c(0.05, 0.15, 0.2, 0.25, 0.5, 0.75)
    ),
    detailed_names = TRUE,
    name_prefix = paste0("SLglmnet")
)

CV.SuperLearner(label, covariates, SL.library = learners$names)
```

In contrast, the `tidyhte` Recipe API requires only the following one line:

```r
add_outcome_model(cfg, "SL.glmnet", alpha = c(0.0, 0.25, 0.5, 0.75, 1.0))
```

```r
penguins %<>% produce_plugin_estimates(
  # outcome
  food_consumed_g,
  # treatment
  treatment,
  # covariates
  species, island, sex, year, bill_length_mm, bill_depth_mm, flipper_length_mm
)
```

Pseudo-outcome construction

Once nuisance functions are estimated, it is simply a matter of combining these results together into the appropriate pseudo-outcome. For typical HTE estimation, the pseudo-outcome of interest is the one analyzed by Kennedy (2020): the uncentered influence function of the average treatment effect (Robins, Rotnitzky, and Zhao 1995).

```
penguins %<>% construct_pseudo_outcomes(food_consumed_g, treatment)
```

Quantities of interest

Finally, it comes to the most glamorous part of the analysis, when effects are estimated and put into charts.

The design of `tidyhte` chooses to leave the charting to the end-user, but merely returns a tidy tibble with all of the requested quantities of interest. The package focuses on a few types of quantities of interest: - Marginal Conditional Average Treatment Effects (MCATEs): the standard Conditional Average Treatment Effects (CATEs) in the literature, in which all covariates except one are marginalized over, providing one average effect for one level of a single covariate. This is in contrast to a “Partial” CATE, in which other variables are “controlled for” in some way. This does not provide a satisfying causal interpretation without assumptions of joint randomization of treatment and covariates. - Variable Importance (VIMP): Using Williamson et al. (2021), `tidyhte` calculates how much each moderator contributes to the overall reduction in mean-squared-error in a joint model of the heterogeneous effects. - Diagnostics: A wide variety of diagnostics are provided for all models fit as part of the `tidyhte` estimation process. These include single-number summaries like mean-squared-error or AUC, as well as entire receiver operating characteristic curves and coefficients in the SuperLearner ensembles.

```
penguins %>%
  estimate_QoI(species, island, sex, year, bill_length_mm, bill_depth_mm, flipper_length_mm) ->
  results
```

The resulting tibble provides looks like the following:

```
results
```

```
## # A tibble: 337 x 6
##    estimand term    value level     estimate std_error
##    <chr>    <chr>   <dbl> <chr>        <dbl>     <dbl>
##  1 MCATE    species    NA Adelie     -97.9        3.06
##  2 MCATE    species    NA Gentoo     101.         3.87
##  3 MCATE    species    NA Chinstrap   -7.70       5.37
##  4 MCATE    island     NA Torgersen  -94.2        6.14
##  5 MCATE    island     NA Biscoe      47.1        7.59
##  6 MCATE    island     NA Dream      -49.7        5.54
##  7 MCATE    sex        NA male        -7.35       7.77
##  8 MCATE    sex        NA female      -9.85       7.27
##  9 MCATE    year     2007 <NA>       -13.1        9.36
## 10 MCATE    year     2008 <NA>         0.809      9.41
## # i 327 more rows
```

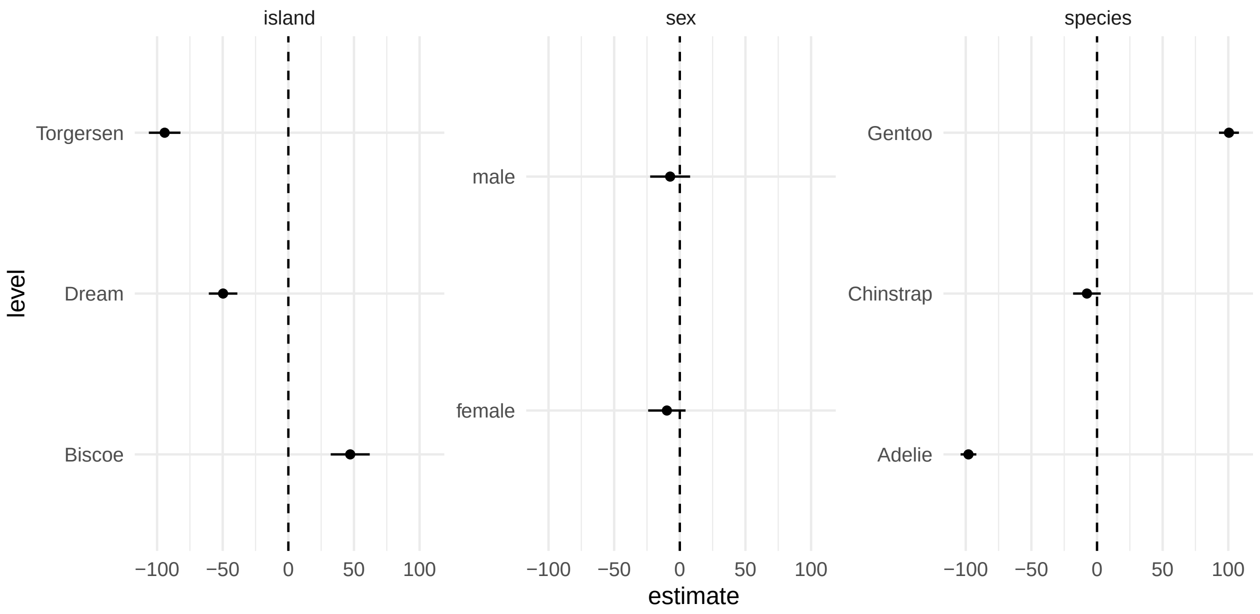
The `estimand` column denotes the class of Quantities of Interest to be estimated, and include such values as

"MCATE" and "VIMP". The `term` column denotes the covariate being referred to (if relevant) in the results. For example, a calculated MCATE refers to a particular covariate, indicated by this column. The columns `value` and `level` refer to quantities which indicate a more granular division within a particular term, with the former representing a numeric value while the latter indicates a categorical one. For example, in the case of the MCATE, if the covariate is discrete, the value at which the MCATE was calculated would be in the `level` column, while if it were continuous, it would be in the `value` column. Each of these columns is, therefore, type-stable. The final two columns for `estimate` and `std_error` are self explanatory. The details on their calculation lie with the particular Quantity of Interest requested.

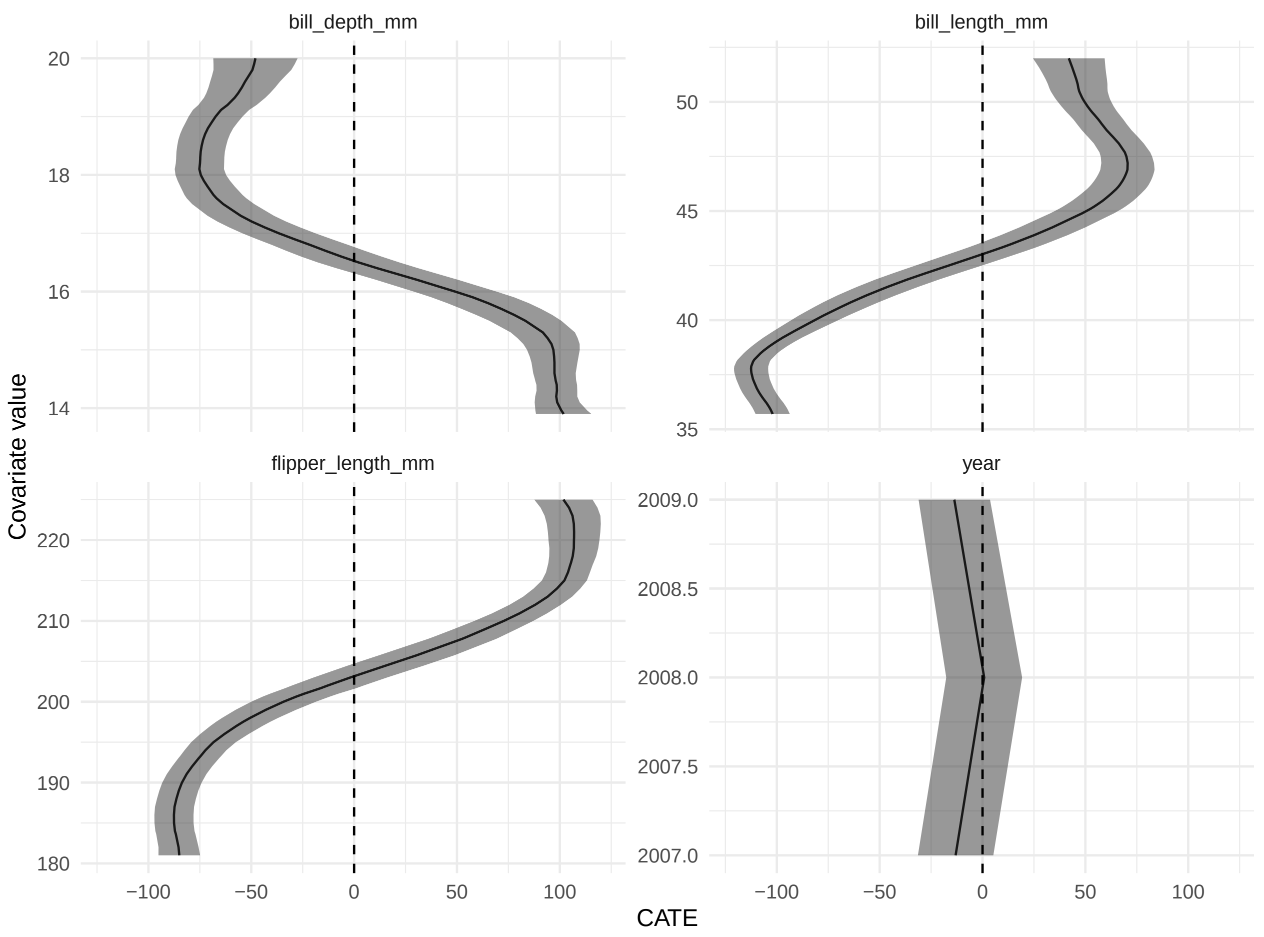
Plotting MCATEs

It's then a simple matter to plot results using code like the following:

```r
filter(results, estimand == "MCATE", is.na(value)) %>%
ggplot(aes(level, estimate)) +
geom_point() +
geom_linerange(aes(ymin = estimate - 1.96 * std_error, ymax = estimate + 1.96 * std_error)) +
geom_hline(yintercept = 0, linetype = "dashed") +
coord_flip() +
facet_wrap(~term, scales = "free_y")
```



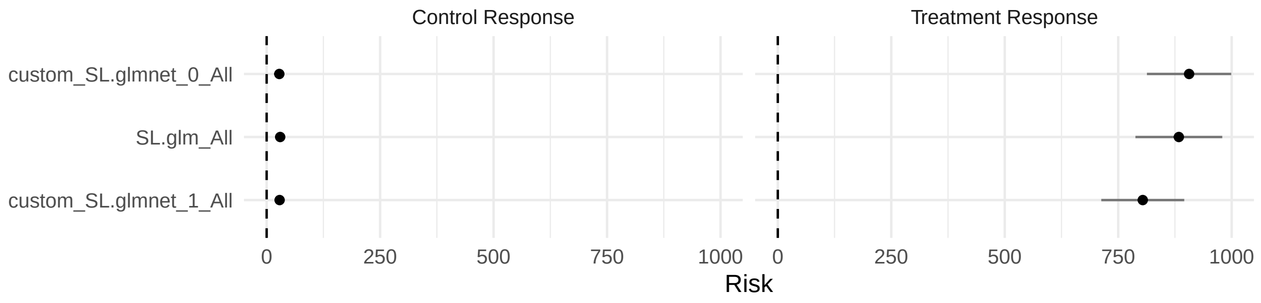
```r
filter(results, estimand == "MCATE", is.na(level)) %>%
ggplot(aes(value, estimate)) +
geom_line() +
geom_ribbon(
  aes(ymin = estimate - 1.96 * std_error, ymax = estimate + 1.96 * std_error),
  alpha = 0.5
) +
geom_hline(yintercept = 0, linetype = "dashed") +
scale_x_continuous("Covariate value") +
scale_y_continuous("CATE") +
coord_flip() +
facet_wrap(~term, scales = "free_y")
```



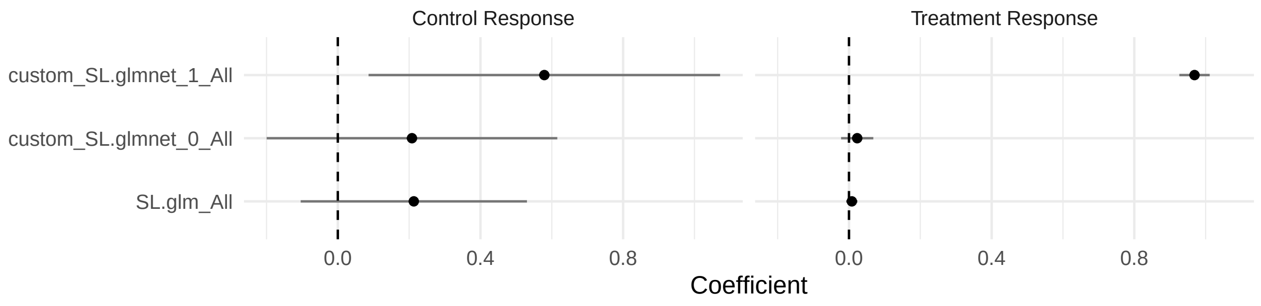
Plotting diagnostics

Similarly, it's easy to plot diagnostic information:

```r
filter(results, estimand == "SL risk") %>%
ggplot(aes(reorder(term, estimate), estimate)) +
geom_point() +
geom_linerange(
  aes(ymin = estimate - 1.96 * std_error, ymax = estimate + 1.96 * std_error),
  alpha = 0.5
) +
geom_hline(yintercept = 0, linetype = "dashed") +
scale_x_discrete("") +
scale_y_continuous("Risk") +
facet_wrap(~level) +
coord_flip()
```



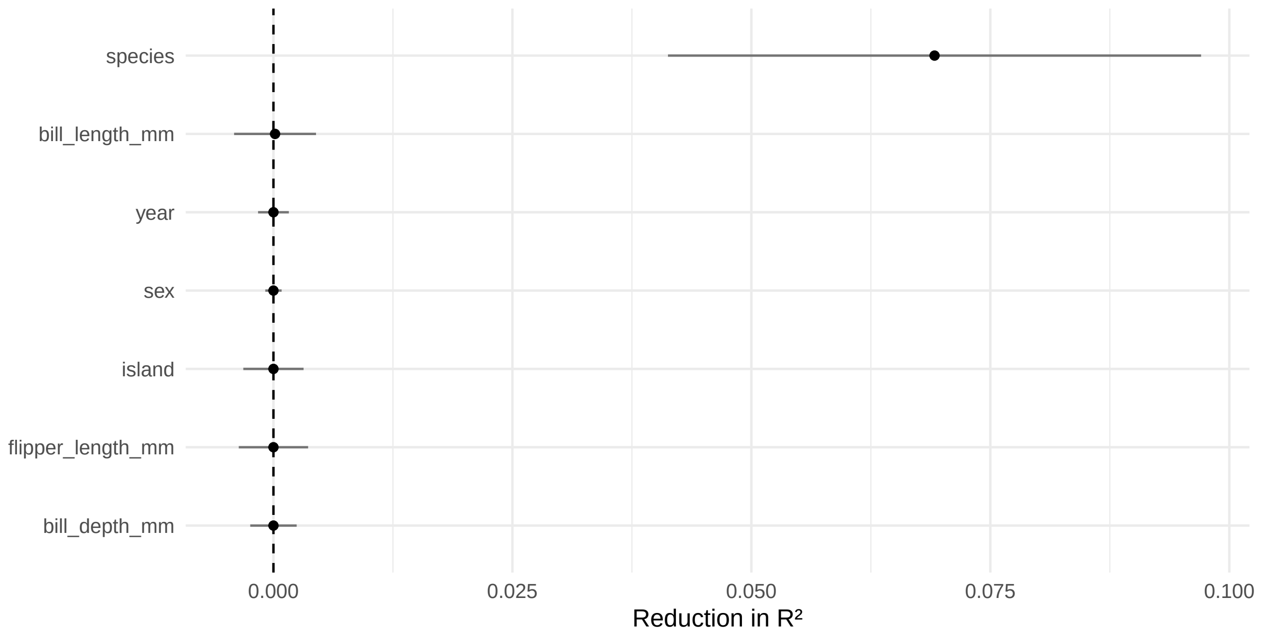
```r
filter(results, estimand == "SL coefficient") %>%
ggplot(aes(reorder(term, estimate), estimate)) +
geom_point() +
geom_linerange(
  aes(ymin = estimate - 1.96 * std_error, ymax = estimate + 1.96 * std_error),
  alpha = 0.5
) +
geom_hline(yintercept = 0, linetype = "dashed") +
scale_x_discrete("") +
scale_y_continuous("Coefficient") +
facet_wrap(~level) +
coord_flip()
```



Plotting VIMP

And finally, we can examine the variable importance measure of Williamson et al. (2021) when applied to a joint model of the pseudo-outcome:

```r
filter(results, estimand == "VIMP") %>%
ggplot(aes(reorder(term, estimate), estimate)) +
geom_point() +
geom_linerange(
  aes(ymin = estimate - 1.96 * std_error, ymax = estimate + 1.96 * std_error),
  alpha = 0.5
) +
geom_hline(yintercept = 0, linetype = "dashed") +
scale_x_discrete("") +
scale_y_continuous("Reduction in R²") +
coord_flip()
```



Replicating analyses

One of the biggest benefits of `tidyhte` is its ability to repeat similar analyses.

As an example, to perform the same analysis used above on a different outcome would require the following code:

```r
penguins %<>% produce_plugin_estimates(
  # outcome
  body_mass_g,
  # treatment
  treatment,
  # covariates
  species, island, sex, year, bill_length_mm, bill_depth_mm, flipper_length_mm
) %>%
construct_pseudo_outcomes(body_mass_g, treatment) %>%
estimate_QoI(
  species, island, sex, year, bill_length_mm, bill_depth_mm, flipper_length_mm
) -> results_mass
```

All the same quantities can be easily plotted for this outcome as well, and results may be joined together conveniently.

```r
results_all <- bind_rows(
  results %>% mutate(outcome = "food_consumed_g"),
  results_mass %>% mutate(outcome = "body_mass_g")
)
```

By allowing the user to flexibly compose HTE estimators, it drastically reduces the amount of work necessary for a typical HTE analysis which by nature tends to involve multiple moderators, models and outcomes.

Conclusion

This paper has introduced the concepts underlying the `tidyhte` package and given examples as to how the package can be used. In general, the package is written in a sufficiently general way that some features can be added with relatively little work. For instance, adding new plugin models is as simple as providing some information on configuration as well as standardizing train / predict methods. This is similarly true for providing new ways to summarize CATEs for plotting, which is handled in much the same way. This makes it relatively easy for methodologists to fit in their preferred fitting methods and thereby extending `tidyhte`'s functionality.

Acknowledgements

We gratefully acknowledge the collaboration with the US 2020 Facebook and Instagram Election Project for being a testbed for the initial versions of `tidyhte`, particularly Pablo Barberá at Meta. We received no financial support for this software project.

References

Abadie, Alberto, Susan Athey, Guido W Imbens, and Jeffrey M Wooldridge. 2023. "When Should You Adjust Standard Errors for Clustering?" *The Quarterly Journal of Economics* 138 (1): 1–35.

Calonico, Sebastian, Matias D Cattaneo, and Max H Farrell. 2019. "Nprobust: Nonparametric Kernel-Based Estimation and Robust Bias-Corrected Inference." *arXiv Preprint arXiv:1906.00198*.

Chernozhukov, Victor, Denis Chetverikov, Mert Demirer, Esther Duflo, Christian Hansen, Whitney Newey, and James Robins. 2018. "Double/Debiased Machine Learning for Treatment and Structural Parameters." *The Econometrics Journal* 21 (1): C1–68.

Fan, Jianqing, and Irene Gijbels. 2018. *Local Polynomial Modelling and Its Applications: Monographs on Statistics and Applied Probability 66*. Routledge.

Higgins, Michael J, Fredrik Sävje, and Jasjeet S Sekhon. 2016. "Improving Massive Experiments with Threshold Blocking." *Proceedings of the National Academy of Sciences* 113 (27): 7369–76.

Horst, Allison M, Alison Presmanes Hill, and Kristen B Gorman. 2020. "Allisonhorst/Palmerpenguins: V0.1.0." Zenodo. https://doi.org/10.5281/zenodo.3960218.

Kennedy, Edward H. 2020. "Optimal Doubly Robust Estimation of Heterogeneous Causal Effects." *arXiv Preprint arXiv:2004.14497*.

Robins, James M, Andrea Rotnitzky, and Lue Ping Zhao. 1995. "Analysis of Semiparametric Regression Models for Repeated Outcomes in the Presence of Missing Data." *Journal of the American Statistical Association* 90 (429): 106–21.

Tsiatis, Anastasios A. 2006. "Semiparametric Theory and Missing Data."

Tsiatis, Anastasios A, Marie Davidian, Min Zhang, and Xiaomin Lu. 2008. "Covariate Adjustment for Two-Sample Treatment Comparisons in Randomized Clinical Trials: A Principled yet Flexible Approach." *Statistics in Medicine* 27 (23): 4658–77.

Van der Laan, Mark J, MJ Laan, and James M Robins. 2003. *Unified Methods for Censored Longitudinal Data and Causality*. Springer Science & Business Media.

Van der Laan, Mark J, Eric C Polley, and Alan E Hubbard. 2007. "Super Learner." *Statistical Applications in Genetics and Molecular Biology* 6 (1).

Williamson, Brian D, Peter B Gilbert, Marco Carone, and Noah Simon. 2021. "Nonparametric Variable Importance Assessment Using Machine Learning Techniques." *Biometrics* 77 (1): 9–22.

Zou, Hui, and Trevor Hastie. 2005. "Regularization and Variable Selection via the Elastic Net." *Journal of the Royal Statistical Society Series B: Statistical Methodology* 67 (2): 301–20.