Generic Machine Learning Inference on Heterogeneous Treatment Effects Using the Package `GenericML`

Max Welz[1,2] Andreas Alfons[1] Mert Demirer[3]
Victor Chernozhukov[3]

[1]Erasmus School of Economics, Erasmus University Rotterdam

[2]Erasmus University Medical Center (Erasmus MC)

[3]Massachusetts Institute of Technology

useR!, June 21, 2022

Motivation

Recent literature in causal inference is focused on heterogeneous treatment effects

- Often based on Machine Learning (ML) techniques
- Goal: Consistent estimation and uniformly valid inference on conditional average treatment effect (CATE)

$\longrightarrow$ Difficult w/o strong assumptions, especially in high dimensions!

$\longrightarrow$ Generic Machine Learning Inference (Generic ML; Chernozhukov, Demirer, Duflo, and Fernández-Val, 2020) remedies this in randomized experiments

Motivation

Recent literature in causal inference is focused on heterogeneous treatment effects

- Often based on Machine Learning (ML) techniques
- Goal: Consistent estimation and uniformly valid inference on conditional average treatment effect (CATE)

$\longrightarrow$ Difficult w/o strong assumptions, especially in high dimensions!

$\longrightarrow$ Generic Machine Learning Inference (Generic ML; Chernozhukov, Demirer, Duflo, and Fernández-Val, 2020) remedies this in randomized experiments

Motivation

Recent literature in causal inference is focused on heterogeneous treatment effects

- Often based on Machine Learning (ML) techniques
- Goal: Consistent estimation and uniformly valid inference on conditional average treatment effect (CATE)

$\longrightarrow$ Difficult w/o strong assumptions, especially in high dimensions!

$\longrightarrow$ Generic Machine Learning Inference (Generic ML; Chernozhukov, Demirer, Duflo, and Fernández-Val, 2020) remedies this in randomized experiments

Setup

Let

- Y be the outcome
- Z be a possibly high-dimensional vector of covariates
- D be a binary treatment assignment variable

$\longrightarrow$ Observe $(Y_i, Z_i, D_i)_{i=1}^{N}$ as i.i.d. copies of (Y, Z, D)

$\longrightarrow$ Assume unconfoundedness and random treatment assignment

Causal Functions

The assumptions identify the causal functions (b_0, s_0) in

$$Y = b_0(Z) + Ds_0(Z) + U, \qquad \mathrm{E}[U \mid Z, D] = 0,$$

where

$$b_0(Z) = \mathrm{E}[Y \mid D = 0, Z]$$

is the baseline conditional average (BCA), and

$$s_0(Z) = \mathrm{E}[Y \mid D = 1, Z] - \mathrm{E}[Y \mid D = 0, Z]$$

is the conditional average treatment effect (CATE)

Causal Functions

The assumptions identify the causal functions (b_0, s_0) in

$$Y = b_0(Z) + Ds_0(Z) + U, \qquad \mathrm{E}[U \mid Z, D] = 0,$$

where

$$b_0(Z) = \mathrm{E}[Y \mid D = 0, Z]$$

is the baseline conditional average (BCA), and

$$s_0(Z) = \mathrm{E}[Y \mid D = 1, Z] - \mathrm{E}[Y \mid D = 0, Z]$$

is the conditional average treatment effect (CATE)

Causal Functions

The assumptions identify the causal functions (b_0, s_0) in

$$Y = b_0(Z) + D s_0(Z) + U, \qquad \mathrm{E}[U \mid Z, D] = 0,$$

where

$$b_0(Z) = \mathrm{E}[Y \mid D = 0, Z]$$

is the baseline conditional average (BCA), and

$$s_0(Z) = \mathrm{E}[Y \mid D = 1, Z] - \mathrm{E}[Y \mid D = 0, Z]$$

is the conditional average treatment effect (CATE)

Focus of Generic ML

Generic ML focuses on estimation and inference on

key features of $s_0(Z)$ rather than $s_0(Z)$ itself

The key features are

$\longrightarrow$ Best Linear Predictor (BLP)

$\longrightarrow$ Group Average Treatment Effects (GATES)

$\longrightarrow$ Classification Analysis (CLAN)

Focus of Generic ML

Generic ML focuses on estimation and inference on

key features of $s_0(Z)$ rather than $s_0(Z)$ itself

The key features are

$\longrightarrow$ Best Linear Predictor (BLP)

$\longrightarrow$ Group Average Treatment Effects (GATES)

$\longrightarrow$ Classification Analysis (CLAN)

Generic ML

1. Randomly partition the data in two disjoint sets A and M
2. On set A, use some machine learner to obtain estimates $B(Z)$ and $S(Z)$ of $b_0(Z)$ and $s_0(Z)$, respectively
3. On set M, calculate the key features of $s_0(Z)$

Two sources of uncertainty:

- Estimation uncertainty (conditional on set A) from Step 2
- Splitting uncertainty from the sample splitting in Step 1

$\longrightarrow$ Address by repeating Steps 1–3 many times

Generic ML

1. Randomly partition the data in two disjoint sets A and M
2. On set A, use some machine learner to obtain estimates $B(Z)$ and $S(Z)$ of $b_0(Z)$ and $s_0(Z)$, respectively
3. On set M, calculate the key features of $s_0(Z)$

Two sources of uncertainty:

- Estimation uncertainty (conditional on set A) from Step 2
- Splitting uncertainty from the sample splitting in Step 1

$\longrightarrow$ Address by repeating Steps 1–3 many times

Generic ML

1. Randomly partition the data in two disjoint sets A and M
2. On set A, use some machine learner to obtain estimates $B(Z)$ and $S(Z)$ of $b_0(Z)$ and $s_0(Z)$, respectively
3. On set M, calculate the key features of $s_0(Z)$

Two sources of uncertainty:

- Estimation uncertainty (conditional on set A) from Step 2
- Splitting uncertainty from the sample splitting in Step 1

$\longrightarrow$ Address by repeating Steps 1–3 many times

Inference

Variational Estimation and Inference (VEIN):

- Fix significance level $\alpha \in (0, 0.5)$
- Calculate the key features across S splits of the data
- Take medians across the S splits of each key feature parameter

$\longrightarrow$ Inference on each key feature parameter with size control of level 2α

$\longrightarrow$ Can be repeated for many machine learners (report the “best” one)

Inference

Variational Estimation and Inference (VEIN):

- Fix significance level $\alpha \in (0, 0.5)$
- Calculate the key features across S splits of the data
- Take medians across the S splits of each key feature parameter

$\longrightarrow$ Inference on each key feature parameter with size control of level 2α

$\longrightarrow$ Can be repeated for many machine learners (report the "best" one)

Inference

Variational Estimation and Inference (VEIN):

- Fix significance level $\alpha \in (0, 0.5)$
- Calculate the key features across S splits of the data
- Take medians across the S splits of each key feature parameter

$\longrightarrow$ Inference on each key feature parameter with size control of level 2α

$\longrightarrow$ Can be repeated for many machine learners (report the “best” one)

Inference

Variational Estimation and Inference (VEIN):

- Fix significance level $\alpha \in (0, 0.5)$
- Calculate the key features across S splits of the data
- Take medians across the S splits of each key feature parameter

$\longrightarrow$ Inference on each key feature parameter with size control of level 2α

$\longrightarrow$ Can be repeated for many machine learners (report the "best" one)

Software Implementation

Package `GenericML` (Welz, Alfons, Demirer, and Chernozhukov, 2022)

- CRAN: `https://cran.r-project.org/package=GenericML`
- GitHub: `https://github.com/mwelz/GenericML`

$\longrightarrow$ Flexible, user-friendly, fast, object-oriented

$\longrightarrow$ Based on `mlr3` ecosystem of Lang et al. (2019)

Empirical Example: Setup

We revisit Crépon et al.'s (2015) study on the effects of microcredits[1]

$\longrightarrow$ Sample: 162 villages in rural Morocco, divided into 81 similar pairs

$\longrightarrow$ Randomly select one village in each pair and make microcredits available for the residents

$\longrightarrow$ Measure if total borrowing changes

$\longrightarrow$ Household-level data on 5,513 households

$\longrightarrow$ 97 control variables (after encoding)

[1] We thank Esther Duflo for making the data available to us

Empirical Example: Setup

We revisit Crépon et al.'s (2015) study on the effects of microcredits[1]

$\longrightarrow$ Sample: 162 villages in rural Morocco, divided into 81 similar pairs

$\longrightarrow$ Randomly select one village in each pair and make microcredits available for the residents

$\longrightarrow$ Measure if total borrowing changes

$\longrightarrow$ Household-level data on 5,513 households

$\longrightarrow$ 97 control variables (after encoding)

[1] We thank Esther Duflo for making the data available to us

Empirical Example: Setup

We revisit Crépon et al.'s (2015) study on the effects of microcredits[1]

$\longrightarrow$ Sample: 162 villages in rural Morocco, divided into 81 similar pairs

$\longrightarrow$ Randomly select one village in each pair and make microcredits available for the residents

$\longrightarrow$ Measure if total borrowing changes

$\longrightarrow$ Household-level data on 5,513 households

$\longrightarrow$ 97 control variables (after encoding)

[1] We thank Esther Duflo for making the data available to us

Empirical Example: Setup

We revisit Crépon et al.'s (2015) study on the effects of microcredits[1]

$\longrightarrow$ Sample: 162 villages in rural Morocco, divided into 81 similar pairs

$\longrightarrow$ Randomly select one village in each pair and make microcredits available for the residents

$\longrightarrow$ Measure if total borrowing changes

$\longrightarrow$ Household-level data on 5,513 households

$\longrightarrow$ 97 control variables (after encoding)

[1] We thank Esther Duflo for making the data available to us

Empirical Example: Setup

We revisit Crépon et al.'s (2015) study on the effects of microcredits[1]

$\longrightarrow$ Sample: 162 villages in rural Morocco, divided into 81 similar pairs

$\longrightarrow$ Randomly select one village in each pair and make microcredits available for the residents

$\longrightarrow$ Measure if total borrowing changes

$\longrightarrow$ Household-level data on 5,513 households

$\longrightarrow$ 97 control variables (after encoding)

[1] We thank Esther Duflo for making the data available to us

Empirical Example: Setup

We revisit Crépon et al.'s (2015) study on the effects of microcredits[1]

$\longrightarrow$ Sample: 162 villages in rural Morocco, divided into 81 similar pairs

$\longrightarrow$ Randomly select one village in each pair and make microcredits available for the residents

$\longrightarrow$ Measure if total borrowing changes

$\longrightarrow$ Household-level data on 5,513 households

$\longrightarrow$ 97 control variables (after encoding)

[1] We thank Esther Duflo for making the data available to us

Empirical Example: Baseline Results

Crépon et al. (2015) find that microcredit availability has...

- low take-up (17% in treatment group)
- significant effect on total borrowing: ATE of MAD[2] 1,206 ($p < 0.01$)

$\longrightarrow$ Use `GenericML` to investigate heterogeneity in this effect!

[2] MAD = Moroccan Dirham

Empirical Example: Baseline Results

Crépon et al. (2015) find that microcredit availability has...

- low take-up (17% in treatment group)
- significant effect on total borrowing: ATE of MAD[2] 1,206 ($p < 0.01$)

$\longrightarrow$ Use `GenericML` to investigate heterogeneity in this effect!

[2] MAD = Moroccan Dirham

Empirical Example: Specification of Learners

⟶ Specify a suite of learners with `mlr3` syntax

⟶ Here: random forest, elastic net, support vector machine, gradient boosting

```
R> library("GenericML")
R>
R> # load data, available in GitHub repo mwelz/GenericML
R> load("slides/data/morocco_preprocessed.Rdata")
R>
R> # specify learners
R> learners <-
+    c("random_forest",
+      "mlr3::lrn('cv_glmnet', s = 'lambda.min', alpha = 0.5)",
+      "mlr3::lrn('svm')",
+      "mlr3::lrn('xgboost')")
```

Empirical Example: Specification of Learners

⟶ Specify a suite of learners with `mlr3` syntax

⟶ Here: random forest, elastic net, support vector machine, gradient boosting

```
R> library("GenericML")
R>
R> # load data, available in GitHub repo mwelz/GenericML
R> load("slides/data/morocco_preprocessed.Rdata")
R>
R> # specify learners
R> learners <-
+    c("random_forest",
+      "mlr3::lrn('cv_glmnet', s = 'lambda.min', alpha = 0.5)",
+      "mlr3::lrn('svm')",
+      "mlr3::lrn('xgboost')")
```

Empirical Example: Customization

Spatial data of 81 village pairs

$\longrightarrow$ Include fixed effects for each pair

$\longrightarrow$ Cluster standard errors on the village level

$\longrightarrow$ `GenericML` allows this through setup functions

$\longrightarrow$ Support for `sandwich` covariance estimators (Zeileis, 2004)

Empirical Example: Customization

Spatial data of 81 village pairs

$\longrightarrow$ Include fixed effects for each pair

$\longrightarrow$ Cluster standard errors on the village level

$\longrightarrow$ `GenericML` allows this through setup functions

$\longrightarrow$ Support for `sandwich` covariance estimators (Zeileis, 2004)

Empirical Example: Customization

Spatial data of 81 village pairs

$\longrightarrow$ Include fixed effects for each pair

$\longrightarrow$ Cluster standard errors on the village level

$\longrightarrow$ `GenericML` allows this through setup functions

$\longrightarrow$ Support for `sandwich` covariance estimators (Zeileis, 2004)

Empirical Example: Customization

Spatial data of 81 village pairs

$\longrightarrow$ Include fixed effects for each pair

$\longrightarrow$ Cluster standard errors on the village level

$\longrightarrow$ `GenericML` allows this through setup functions

$\longrightarrow$ Support for `sandwich` covariance estimators (Zeileis, 2004)

Empirical Example: Customization

`setup_X1()` customizes inclusion of controls and fixed effects

```
R> # include BCA and CATE controls
R> # add fixed effects along variable "vil_pair"
R> X1 <- setup_X1(funs_Z = c("B", "S"),
+                 fixed_effects = vil_pair)
```

`setup_vcov()` customizes covariance estimation

```
R> # calls functions from the "sandwich" package
R> # cluster standard errors along "demi_paire"
R> vcov <- setup_vcov(estimator = "vcovCL",
+                     arguments = list(cluster = demi_paire))
```

Empirical Example: Customization

`setup_X1()` customizes inclusion of controls and fixed effects

```
R> # include BCA and CATE controls
R> # add fixed effects along variable "vil_pair"
R> X1 <- setup_X1(funs_Z = c("B", "S"),
+                  fixed_effects = vil_pair)
```

`setup_vcov()` customizes covariance estimation

```
R> # calls functions from the "sandwich" package
R> # cluster standard errors along "demi_paire"
R> vcov <- setup_vcov(estimator = "vcovCL",
+                      arguments = list(cluster = demi_paire))
```

Empirical Example: Customization

`setup_X1()` customizes inclusion of controls and fixed effects

```
R> # include BCA and CATE controls
R> # add fixed effects along variable "vil_pair"
R> X1 <- setup_X1(funs_Z = c("B", "S"),
+                 fixed_effects = vil_pair)
```

`setup_vcov()` customizes covariance estimation

```
R> # calls functions from the "sandwich" package
R> # cluster standard errors along "demi_paire"
R> vcov <- setup_vcov(estimator = "vcovCL",
+                     arguments = list(cluster = demi_paire))
```

Empirical Example: Customization

`setup_X1()` customizes inclusion of controls and fixed effects

```
R> # include BCA and CATE controls
R> # add fixed effects along variable "vil_pair"
R> X1 <- setup_X1(funs_Z = c("B", "S"),
+                 fixed_effects = vil_pair)
```

`setup_vcov()` customizes covariance estimation

```
R> # calls functions from the "sandwich" package
R> # cluster standard errors along "demi_paire"
R> vcov <- setup_vcov(estimator = "vcovCL",
+                     arguments = list(cluster = demi_paire))
```

`GenericML` Interface

```
R> x <- GenericML(
+     Z = Z, D = D, Y = Y,                          # observed data
+     learners_GenericML = learners,                # learners
+     learner_propensity_score = "constant",        # = 0.5 (RCT)
+     num_splits = 100L,                            # number splits
+     quantile_cutoffs = c(0.2, 0.4, 0.6, 0.8), # grouping
+     significance_level = 0.05,                    # significance level
+     X1_BLP = X1, X1_GATES = X1,                   # regression setup
+     vcov_BLP = vcov, vcov_GATES = vcov,           # covariance setup
+     parallel = TRUE, num_cores = 6L,              # parallelization
+     seed = 20220621)                              # RNG seed
```

. . . and many more arguments for fine-tuning!

$\longrightarrow$ stratified sampling, Horvitz-Thompson transformation. . .

GenericML Interface

```
R> x <- GenericML(
+    Z = Z, D = D, Y = Y,                        # observed data
+    learners_GenericML = learners,              # learners
+    learner_propensity_score = "constant",      # = 0.5 (RCT)
+    num_splits = 100L,                          # number splits
+    quantile_cutoffs = c(0.2, 0.4, 0.6, 0.8),  # grouping
+    significance_level = 0.05,                  # significance level
+    X1_BLP = X1, X1_GATES = X1,                 # regression setup
+    vcov_BLP = vcov, vcov_GATES = vcov,         # covariance setup
+    parallel = TRUE, num_cores = 6L,            # parallelization
+    seed = 20220621)                            # RNG seed
```

…and many more arguments for fine-tuning!

⟶ stratified sampling, Horvitz-Thompson transformation…

Analysis of `GenericML` Objects

Methods for the analysis of the *key features* of CATE

- `get_BLP()`
- `get_GATES()`
- `get_CLAN()`

$\longrightarrow$ linked to rich `plot()` and `print()` methods

Empirical Example: `get_BLP()`

Best Linear Predictor (BLP): Estimates some (β_0, β_1) via OLS:

- $\beta_0 = \mathrm{E}s_0(Z)$ is the ATE
- $\beta_1 \neq 0$ if there is heterogeneity in $s_0(Z)$ and $S(Z)$ predicts it well

```
R> get_BLP(x, plot = TRUE)
BLP generic targets
---
         Estimate   CB lower CB upper p-value
beta.1 1113.50155  273.02645 1935.274 0.00945 **
beta.2    0.35315   -0.04384    0.698 0.08613 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
---
Level of confidence of the confidence bounds (CB): 90 %
```

Empirical Example: `get_BLP()`

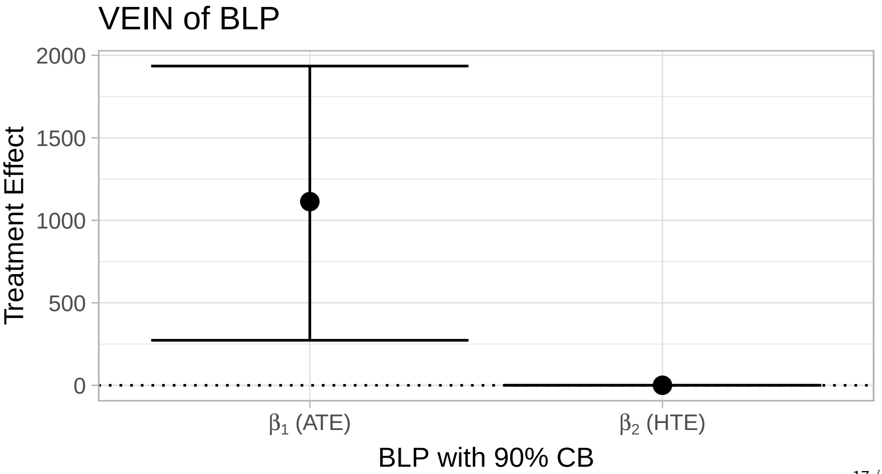
Best Linear Predictor (BLP): Estimates some (β_0, β_1) via OLS:

- $\beta_0 = \mathrm{E}s_0(Z)$ is the ATE
- $\beta_1 \neq 0$ if there is heterogeneity in $s_0(Z)$ and $S(Z)$ predicts it well

```
R> get_BLP(x, plot = TRUE)
BLP generic targets
---
          Estimate   CB lower CB upper p-value
beta.1 1113.50155  273.02645 1935.274 0.00945 **
beta.2    0.35315   -0.04384    0.698 0.08613 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
---
Level of confidence of the confidence bounds (CB): 90 %
```

Empirical Example: `get_BLP()`

```
R> get_BLP(x, plot = TRUE)
```



Empirical Example: `get_GATES()`

Sorted Group Average Treatment Effects (GATES): Build groups

$$G_k := \{S(Z) \in I_k\}, \quad k = 1, \ldots, K,$$

where $I_k = [\ell_{k-1}, \ell_k)$ divide the support of $S(Z)$ into regions

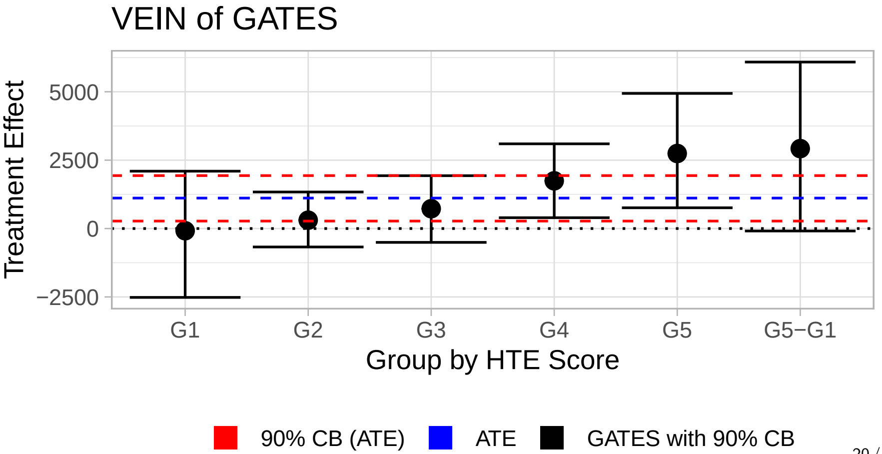
$\longrightarrow$ Estimate group-ATE $\gamma_k := \mathrm{E}[s_0(Z) \mid G_k]$ via OLS

Empirical Example: `get_GATES()`

```
R> get_GATES(x, plot = TRUE)
GATES generic targets
---
               Estimate CB lower CB upper p-value
gamma.1          -80.44 -2517.30     2097 0.93525
gamma.2          305.50  -674.10     1336 0.49251
gamma.3          725.63  -505.53     1932 0.19349
gamma.4         1744.51   395.93     3097 0.01225 *
gamma.5         2743.76   759.85     4940 0.00911 **
gamma.5-gamma.1 2922.13   -89.43     6087 0.05536 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
---
Level of confidence of the confidence bounds (CB): 90 %
```

Empirical Example: `get_GATES()`

```
R> get_GATES(x, plot = TRUE)
```



Empirical Example: `get_CLAN()`

Classification Analysis (CLAN): Observed within-group averages, δ_k, of a variable for groups G_k

For variable `head_age_bl` (age of household's head):

```
R> get_CLAN(x, variable = "head_age_bl", plot = TRUE)
CLAN generic targets for variable 'head_age_bl'
---
                Estimate CB lower CB upper  p-value
delta.1            36.49    34.46   38.554  < 2e-16 ***
delta.2            43.66    42.12   45.210  < 2e-16 ***
delta.3            41.40    39.50   43.258  < 2e-16 ***
delta.4            34.75    32.55   36.853  < 2e-16 ***
delta.5            23.85    21.53   26.151  < 2e-16 ***
delta.5-delta.1   -12.52   -15.61   -9.514 4.44e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
---
Level of confidence of the confidence bounds (CB): 90 %
```

Empirical Example: `get_CLAN()`

Classification Analysis (CLAN): Observed within-group averages, δ_k, of a variable for groups G_k

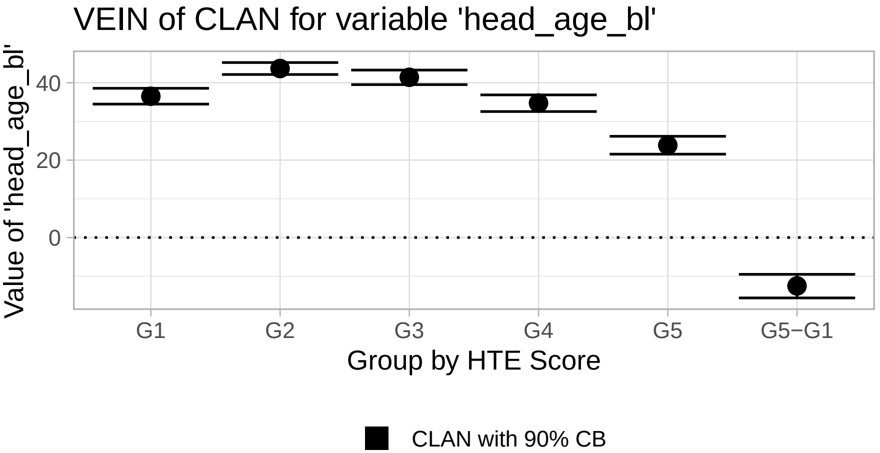
For variable `head_age_bl` (age of household's head):

```
R> get_CLAN(x, variable = "head_age_bl", plot = TRUE)
CLAN generic targets for variable 'head_age_bl'
---
                Estimate CB lower CB upper  p-value
delta.1            36.49    34.46   38.554  < 2e-16 ***
delta.2            43.66    42.12   45.210  < 2e-16 ***
delta.3            41.40    39.50   43.258  < 2e-16 ***
delta.4            34.75    32.55   36.853  < 2e-16 ***
delta.5            23.85    21.53   26.151  < 2e-16 ***
delta.5-delta.1   -12.52   -15.61   -9.514 4.44e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
---
Level of confidence of the confidence bounds (CB): 90 %
```

Empirical Example: `get_CLAN()`

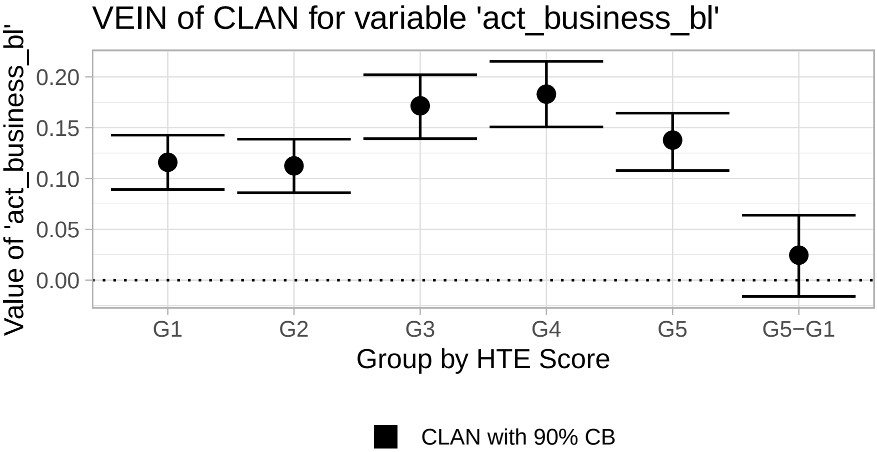
For variable `head_age_bl` (age of household's head):

```
R> get_CLAN(x, variable = "head_age_bl", plot = TRUE)
```



Empirical Example: `get_CLAN()`

For variable `act_business_bl` (indicator that is 1 if declared non-agricultural self-employment activity):



Conclusions and Discussion

Conclusions

- $\longrightarrow$ High-dimensional uniformly valid inference on CATE is hard
- $\longrightarrow$ Generic ML can do so under minimal assumptions by focusing on key features of CATE instead of CATE itself
- $\longrightarrow$ R package `GenericML` available on CRAN

Future work

- $\longrightarrow$ Implement monotonization of confidence bounds
- $\longrightarrow$ Enable support for deep learning, perhaps via `mlr3keras`

Conclusions and Discussion

Conclusions

$\longrightarrow$ High-dimensional uniformly valid inference on CATE is hard

$\longrightarrow$ Generic ML can do so under minimal assumptions by focusing on key features of CATE instead of CATE itself

$\longrightarrow$ R package `GenericML` available on CRAN

Future work

$\longrightarrow$ Implement monotonization of confidence bounds

$\longrightarrow$ Enable support for deep learning, perhaps via `mlr3keras`

References

Victor Chernozhukov, Mert Demirer, Esther Duflo, and Iván Fernández-Val. Generic Machine Learning Inference on Heterogenous Treatment Effects in Randomized Experiments. **arXiv preprint: arXiv:1712.04802**, 2020.

Bruno Crépon, Florencia Devoto, Esther Duflo, and William Parienté. Estimating the Impact of Microcredit on Those Who Take It Up: Evidence from a Randomized Experiment in Morocco. **American Economic Journal: Applied Economics**, 7(1):123–150, 2015.

Michel Lang, Martin Binder, Jakob Richter, Patrick Schratz, Florian Pfisterer, Stefan Coors, Quay Au, Giuseppe Casalicchio, Lars Kotthoff, and Bernd Bischl. `mlr3`: A Modern Object-Oriented Machine Learning Framework in R. **Journal of Open Source Software**, 4 (44):1903, 2019.

Max Welz, Andreas Alfons, Mert Demirer, and Victor Chernozhukov. **`GenericML`: Generic Machine Learning Inference**, 2022. URL `https://CRAN.R-project.org/package=GenericML`. R package version 0.2.2.

Achim Zeileis. Econometric Computing with HC and HAC Covariance Matrix Estimators. **Journal of Statistical Software**, 11(10):1–17, 2004.

Algorithm 1 in Chernozhukov et al. (2020)

IN: Data $= (Y_i, Z_i, D_i)_{i=1}^N$, significance level α, a suite of ML methods, number of splits S

OUT: p-values and $(1-2\alpha)$ confidence intervals of point estimates of each target parameter in GATES, BLP, and CLAN

1. Compute propensity scores $p(Z_i), i = 1, \ldots, N$
2. Do S splits of $\{1, \ldots, N\}$ into disjoint sets A and M of same size
3. **for** each ML method and each split $s = 1, \ldots, S$, **do**
   a. Tune and train each ML method to learn $B(\cdot)$ and $S(\cdot)$ on A
   b. On M, use $B(\cdot)$ and $S(\cdot)$ to estimate the BLP, GATES, CLAN target parameters
   c. Compute some performance measures for the ML methods
4. Choose the best ML method based on the medians of the performance measures
5. Calculate the medians of the confidence bounds, p-values, and point estimates of each target parameter
6. Adjust the confidence bounds and p-values

Best Learner

Compute two performance measures for each learner

$$\widehat{\Lambda} = |\widehat{\beta}_2|^2 \ \widehat{\mathrm{Var}}(S(Z)), \qquad \widehat{\bar{\Lambda}} = \frac{1}{K}\sum_{k=1}^{K} \widehat{\gamma}_k^2$$

$\longrightarrow$ Best learner maximizes their median across S splits
$\longrightarrow$ In the empirical example, that's random forest (get via `get_best()`)