Package ‘PTE’

January 31, 2019

Type Package
Title Personalized Treatment Evaluator
Version 1.7
Date 2019-01-24
Author Adam Kapelner, Alina Levine & Justin Bleich
Maintainer Adam Kapelner <kapelner@qc.cuny.edu>
Description We provide inference for personalized medicine models. Namely, we answer the questions: (1) how much better does a purported personalized recommendation engine for treatments do over a business-as-usual approach and (2) is that difference statistically significant?
License GPL-3
Depends R (>= 3.0)
Imports foreach, parallel, doParallel, graphics, grDevices, stats, survival
RoxygenNote 6.0.1
NeedsCompilation no
Repository CRAN
Date/Publication 2019-01-31 00:03:19 UTC

R topics documented:

continuous_example ............................................. 2
plot.PTE_bootstrap_results .................................. 2
print.PTE_bootstrap_results ................................. 3
PTE ............................................................. 3
PTE_bootstrap_inference ..................................... 4
summary.PTE_bootstrap_results .............................. 7
survival_example ............................................. 8

Index 9
**continuous_example**  
Mock RCT data with a continuous endpoint.

### Description

A list with two objects (a) \(X\), a dataframe with \(n\) rows representing clinical subjects and columns: treatment, \(x_1, x_2, x_3, x_4\) and \(x_5\) where treatment is binary indicating the two arms of the clinical trial and \(x_1, ..., x_5\) are covariates that were collected about each subject and (b) \(y\), a length \(n\) vector storing the continuous response values where, in this mock dataset, larger values indicate "better" outcomes for the subjects.

### Author(s)

My Name <kapelner@qc.cuny.edu>

---

**plot.PTE_bootstrap_results**  
Plots a summary of the bootstrap samples.

### Description

Plots a summary of the bootstrap samples.

### Usage

```r
## S3 method for class 'PTE_bootstrap_results'
plot(x, ...)
```

### Arguments

- **x**  
  A `PTE_bootstrap_results` model object built via running the `PTE_bootstrap_inference` function.

- **...**  
  Other methods passed to `plot`

### Author(s)

Adam Kapelner
print.PTEBootstrapResults

Description
Prints a summary of the model to the console

Usage
## S3 method for class 'PTEBootstrapResults'
print(x, ...)

Arguments
x A PTEBootstrapResults model object built via running the PTEBootstrapInference function.
... Other methods passed to print

Author(s)
Adam Kapelner

PTE

Personalized Medicine Inference

Description
Personalized Medicine...

Author(s)
Adam Kapelner <kapelner@qc.cuny.edu>, Alina Levine and Justin Bleich

References
PTE_bootstrap_inference

Bootstrap inference for a prespecified personalization / recommendation model

Description

Runs B bootstrap samples using a prespecified model then computes the two I estimates based on cross validation. `p` values of the two I estimates are computed for a given `H_0 : \mu_I = \mu_0` and confidence intervals are provided using the basic, percentile methods by default and the BCa method as well if desired.

Usage

```r
PTE_bootstrap_inference(X, y, regression_type = "continuous",
incidence_metric = "odds_ratio", personalized_model_build_function = NULL,
censored = NULL, predict_function = function(mod, Xyleftout) {
predict(mod, Xyleftout) }, difference_function = NULL,
cleanup_mod_function = NULL, y_higher_is_better = TRUE, verbose = FALSE,
full_verbose = FALSE, h_P_mu_equals = NULL, pct_leave_out = 0.1,
m_prop = 1, B = 3000, alpha = 0.05, run_bca_bootstrap = FALSE,
display_adversarial_score = FALSE, num_cores = NULL)
```

Arguments

- **X** A \( n \times p \) dataframe of covariates where one column is labeled "treatment" and it is a binary vector of treatment allocations in the study.
- **y** An \( n \)-length numeric vector which is the response
- **regression_type** A string indicating the regression problem. Legal values are "continuous" (the response \( y \) is a real number with no missing data, the default), "incidence" (the response \( y \) is either 0 or 1) and "survival". If the type is "survival", the user must also supply additional data via the parameter `censored`.
- **incidence_metric** Ignored unless the `regression_type` is "incidence" and `difference_function` is set to NULL (in the latter case, you have specified a more custom metric). Then, this parameter allows the user to select which of the three standard metrics to use for comparison: "probability_difference", "risk_ratio", "odds_ratio" where the default is "odds_ratio".
- **personalized_model_build_function** An R function that will be evaluated to construct the personalized medicine / recommendation model. In the formula for the model, the response is "y", the treatment vector is "treatment" and the data is "Xytrain". This function must return some type of object that can be used for prediction later via predict_function. Here are the defaults for each `regression_type`. They are linear models with first order interactions:
personalized_model_build_function = switch(regression_type, continuous = function(Xytrain) #default is OLS regression lm(y ~ . * treatment, data = Xytrain), incidence = function(Xytrain) #default is logistic regression glm(y ~ . * treatment, data = Xytrain, family = "binomial"), survival = function(Xytrain) #default is Weibull regression survreg(Surv(Xytrain$y, Xytrain$censored) ~ (. - censored) * treatment, data = Xytrain, dist = "weibull")
)

censored
Only required if the regression_type is "survival". In this case, this vector is of length n and is binary where 0 indicates censored and 1 indicates uncensored. In a clinical trial, someone who is still alive at the end of the study or was lost to follow up will receive a censor value of 0, while someone who died during the study will receive a censor value of 1. n and is binary where 0 indicates censorship (e.g. the patient died).

predict_function
An R function that will be evaluated on left out data after the model is built with the training data. This function uses the object "mod" that is the result of the personalized_model_build_function and it must make use of "Xyleftout", a subset of observations from X. This function must return a scalar numeric quantity for comparison. The default function is predict(mod, obs_left_out) e.g. the default looks like:
function(mod, Xyleftout) predict(mod, Xyleftout)

difference_function
A function which takes the result of one out of sample experiment (bootstrap or not) of all n samples and converts it into a difference that will be used as a score in a score distribution to determine if the personalization model is statistically significantly able to distinguish subjects. The function looks as follows:
function(results, indices_1_1, indices_0_0, indices_0_1, indices_1_0) ... c(rec_vs_non_rec_diff_score, rec_vs_all_score, rec_vs_best_score)
where results is a matrix consisting of columns of the estimated response of the treatment administered, the estimated response of the counterfactual treatment, the administered treatment, the recommended treatment based on the personalization model, the real response, and if this subject was censored (0 if so). Here are a couple of example entries:
est_true est_counterfactual given_tx rec_tx real_y censored 166.8 152.2 1 1 324 1 1679.1 2072.0 1 0 160 0
The arguments indices_1_1, indices_0_0, indices_0_1, indices_1_0 give the indices of the subjects whose treatment was administered as 1 and whose optimal was 1, whose treatment was administered 0 and whose optimal was 0, etc.
This function should return three numeric scores: the recommend vs. the non-recommended (adversarial), the recommended vs. all (all) and the recommended vs. the best average treatment (best) as a 3-dimensional vector as illustrated above.
By default, this parameter is NULL which means for continuous and incidence the average difference is used and for survival, the median Kaplan-Meier survival is used.
A function that is called at the end of a cross validation iteration to cleanup the model in some way. This is used for instance if you would like to release the memory your model is using but generally does not apply. The default is NA for "no function."

**y_higher_is_better**

True if a response value being higher is clinically "better" than one that is lower (e.g. cognitive ability in a drug trial for the mentally ill). False if the response value being lower is clinically "better" than one that is higher (e.g. amount of weight lost in a weight-loss trial). Default is TRUE.

**verbose**

Prints out a dot for each bootstrap sample. This only works on some platforms.

**full_verbose**

Prints out full information for each cross validation model for each bootstrap sample. This only works on some platforms.

**H_0.muEquals**

The $\mu_{H_0}$ value in $H_0$. Default is NULL which specifies 0 for regression types continuous, survival and incidence (with incidence metric "probability_difference") or 1 if the regression type is incidence and the incidence metric is "risk_ratio" or "odds_ratio". These defaults essentially answer the question: does my allocation procedure do better than the business-as-usual / naive allocation procedure?

**pct_leave_out**

In the cross-validation, the proportion of the original dataset left out to estimate out-of-sample metrics. The default is 0.1 which corresponds to 10-fold cross validation.

**m_prop**

Within each bootstrap sample, the proportion of the total number of rows of $X$ to sample without replacement. $m_{prop} < 1$ ensures the number of rows sampled is less than $n$ which fixes the consistency of the bootstrap estimator of a non-smooth functional. The default is 1 since non-smoothness may not be a common issue.

**B**

The number of bootstrap samples to take. We recommend making this as high as you can tolerate given speed considerations. The default is 3000.

**alpha**

Defines the confidence interval size (1 - alpha). Defaults to 0.05.

**run_bca_bootstrap**

Do the BCA bootstrap as well. This takes double the time. It defaults to FALSE.

**display_adversarial_score**

The adversarial score records the personalization metric versus the deliberate opposite of the personalization. This does not correspond to any practical situation but it is useful for debugging. Default is FALSE.

**num_cores**

The number of cores to use in parallel to run the bootstrap samples more rapidly. Defaults to NULL which automatically sets it to one if there is one available processor or if there are multiple available processors, the number of available processors save one.

**Value**

A results object of type "PTE_bootstrap_results" that contains much information about the observed results and the bootstrap runs, including hypothesis testing and confidence intervals.
Author(s)
Adam Kapelner

Examples

```r
## Not run:
library(PTE)
B = 1000 #lower this for quicker demos

##response: continuous
data(continuous_example)
X = continuous_example$X
y = continuous_example$y
pte_results = PTE_bootstrap_inference(X, y, regression_type = "continuous", B = B)
pte_results

##response: incidence
data(continuous_example)
X = continuous_example$X
y = continuous_example$y
y = ifelse(y > quantile(y, 0.75), 1, 0) #force incidence and pretend y came to you this way
#there are three ways to assess incidence effects below:
# odds ratio, risk ratio and probability difference
pte_results = PTE_bootstrap_inference(X, y, regression_type = "incidence", B = B)
pte_results
pte_results = PTE_bootstrap_inference(X, y, regression_type = "incidence", B = B,
  incidence_metric = "risk_ratio")
pte_results
pte_results = PTE_bootstrap_inference(X, y, regression_type = "incidence", B = B,
  incidence_metric = "probability_difference")
pte_results

##response: survival
data(survival_example)
X = survival_example$X
y = survival_example$y
censored = survival_example$censored
pte_results = PTE_bootstrap_inference(X, y, censored = censored,
  regression_type = "survival",
  B = 1000)
pte_results

## End(Not run)
```
survival_example

Description

Prints a summary of the model to the console

Usage

## S3 method for class 'PTE_bootstrap_results'
summary(object, ...)

Arguments

object A PTE_bootstrap_results model object built via running the PTE_bootstrap_inference function.

... Other methods passed to summary

Author(s)

Adam Kapelner

---

survival_example  Mock RCT data with a survival endpoint.

Description

A list with three objects (a) \( X \), a dataframe with \( n \) rows representing clinical subjects and columns: treatment, \( x_1, x_2, x_3 \) and \( x_4 \) where treatment is binary indicating the two arms of the clinical trial and \( x_1, ..., x_4 \) are covariates that were collected about each subject (b) \( y \), a length \( n \) vector storing the survival response values (a time measurement) where, in this mock dataset, smaller values indicate "better" survival outcomes for the subjects and (c) censored, a length \( n \) vector storing the censor dummies where \( c_{16} = 1 \) means the response \( y_{16} \) was censored and thus the truth value of \( y_{16} \) is unknown and \( y_{16} \) only represents the moment it was censored (and \( c_{16} = 0 \) means it was uncensored and \( y_{16} \) is the true response value).

Author(s)

My Name <kapelner@qc.cuny.edu>
Index

* Topic **Personalized**
  PTE, 3
* Topic **bootstrap**
  PTE, 3
* Topic **medicine,**
  PTE, 3

continuous_example, 2
plot.PTE_bootstrap_results, 2
print.PTE_bootstrap_results, 3
PTE, 3
PTE-package (PTE), 3
PTE_bootstrap_inference, 4

summary.PTE_bootstrap_results, 7
survival_example, 8