Package ‘DNetCausalPATT’

June 4, 2024

Type Package

Title Estimation of Conditional Average Treatment Effects (CATE) and Population Average Treatment Effects on the Treated (PATT)

Version 0.0.103

Maintainer Nguyen Huynh <nguyenhuynh831@gmail.com>

Description DNetCausalPATT is an R package that provides functions to estimate Conditional Average Treatment Effects (CATE) and Population Average Treatment Effects on the Treated (PATT) from experimental or observational data using the Super Learner (SL) ensemble method and Deep neural networks. The package first provides functions to implement meta-learners such as the Single-learner (S-learner) and Two-learner (T-learner) described in Künzel et al. (2019) <doi:10.1073/pnas.1804597116> for estimating the CATE. The S- and T-learner are each estimated using the SL ensemble method and deep neural networks. It then provides functions to implement the Ottoboni and Poulos (2020) <doi:10.1515/jci-2018-0035> PATT-C estimator to obtain the PATT from experimental data with noncompliance by using the SL ensemble method and deep neural networks.

License GPL-3

Encoding UTF-8

LazyData true

Imports ROCR, xgboost, SuperLearner, class, randomForest, glmnet, gam, e1071, gbm, Hmisc, weights

Suggests testthat, ggplot2, tidyr

Roxygen list(markdown = TRUE)

RoxygenNote 7.2.3

Depends R (>= 3.5.0)

NeedsCompilation no

Author Nguyen K. Huynh [aut, cre] (<https://orcid.org/0000-0002-6234-7232>),
Bumba Mukherjee [aut],
Irvin (Chen-Yu) Lee [aut]

R topics documented:

`complier_mod` *Train complier model using ensemble methods*

Description

Train model using group exposed to treatment with compliance as binary outcome variable and covariates.

Usage

```
complier_mod(
  exp.data,
  complier.formula,
  treat.var,
  ID = NULL,
  SL.library = NULL
)
```

Arguments

`exp.data`	list object of experimental data.
`ID`	string for name of indentifier variable.
`SL.library.`	Employs extreme gradient boosting, elastic net regression, random forest, and neural nets.

Value

model object of trained model.

complier_predict *Complier model prediction*

Description

Predict Compliance from control group in experimental data

Usage

```
complier_predict(complier.mod, exp.data, treat.var, compl.var)
```

Arguments

complier.mod	output from trained ensemble superlearner model.
exp.data	experimental dataset

Value

`data.frame` object with true compliers, predicted compliers in the control group, and all compliers (actual + predicted).

IND_exp_data *Survey Experiment of Support for Populist Policy*

Description

Shortened version of survey response data that incorporates a vignette survey experiment. The vignette describes an international crisis between country A and B. After reading this vignette, respondents are randomly assigned to the control group or to one of two treatments: policy prescription to said crisis by strong (populist) leader and centrist (non-populist) leader. The respondents are then asked whether they are willing to support the policy decision to fight a war against country A, which is the dependent variable.

Usage

```
data(IND_exp_data)
```

Format

`IND_exp_data`**:**

A data frame with 257 rows and 12 columns:

Female Gender.

Age Age of participant.

Income Monthly household income.

Religion Religious denomination

Imp_rel Importance of religion in life.

Education Educational level of participant.

Ideol_lr Political ideology of participant.

Empl_status Employment status of participant.
Marital_status Marital status of participant.
job_worry Concern about job loss.
Exp1trt Binary treatment measure of leader type.
Exp1_dv1 Binary outcome measure for willingness to fight war. #' ...

Source

Yadav and Mukherjee (2024)

`IND_pop_data` *World Value Survey India Sample*

Description

World Value Survey (WVS) Data for India in 2022. The variables drawn from the said WVS India data match the covariates from the India survey experiment sample.

Usage

```
data(IND_pop_data)
```

Format

`IND_pop_data`**:**
A data frame with 846 rows and 13 columns:

Female Respondent's Sex.
Age Age of respondent.
Income Income group of Household.
Religion Religious denomination
Imp_rel Importance of religion in respondent's life.
Education Educational level of respondent.
Ideol Political ideology of respondent.
Empl_status Employment status and full-time employee.
Marital Marital status of respondent.
job_worry Concern about job loss.
Exp1_trt Binary treatment measure of leader type.
Exp1_dv_willing Binary (Yes/No) outcome measure for willingness to fight war.
strong_leader Binary measure of preference for strong leader. ...

Source

Haerpfer, C., Inglehart, R., Moreno, A., Welzel, C., Kizilova, K., Diez-Medrano J., M. Lagos, P. Norris, E. Ponarin & B. Puranen et al. (eds.). 2020. World Values Survey: Round Seven – Country-Pooled Datafile. Madrid, Spain & Vienna, Austria: JD Systems Institute & WVSA Secretariat. <doi.org/10.14281/18241.1>

`neuralnet_complier_mod` *Train compliance model using neural networks*

Description

Train model using group exposed to treatment with compliance as binary outcome variable and covariates.

Usage

```
neuralnet_complier_mod(
  complier.formula,
  exp.data,
  treat.var,
  algorithm = "rprop+",
  hidden.layer = c(4, 2),
  ID = NULL,
  stepmax = 1e+08
)
```

Arguments

`complier.formula`	formula for complier variable as outcome and covariates (c ~ x)
`exp.data`	`data.frame` for experimental data.
`treat.var`	string for treatment variable.
`algorithm`	string for algorithm for training neural networks. Default set to the Resilient back propagation with weight backtracking (rprop+). Other algorithms include backprop', rprop-', 'sag', or 'slr' (see `neuralnet` package).
`hidden.layer`	vector for specifying hidden layers and number of neurons.
`ID`	string for identifier variable
`stepmax`	maximum number of steps.

Value

trained complier model object

`neuralnet_pattc_counterfactuals` *Assess Population Data counterfactuals*

Description

Create counterfactual datasets in the population for compliers and noncompliers. Then predict potential outcomes using trained model from `neuralnet_response_model`.

Usage

```
neuralnet_pattc_counterfactuals(
  pop.data,
  neuralnet.response.mod,
  ID = NULL,
  cluster = NULL
)
```

Arguments

`pop.data`	population data.
`neuralnet.response.mod`	trained model from. `neuralnet_response_model`.
`ID`	string for identifier variable.
`cluster`	string for clustering variable (currently unused).

Value

`data.frame` of predicted outcomes of response variable from counterfactuals.

`neuralnet_predict` *Predicting Compliance from experimental data*

Description

Predicting Compliance from control group experimental data

Usage

```
neuralnet_predict(neuralnet.complier.mod, exp.data, treat.var, compl.var)
```

Arguments

`neuralnet.complier.mod`	results from `neuralnet_complier_mod`
`exp.data`	`data.frame` of experimental data
`treat.var`	string for treatment variable
`compl.var`	string for compliance variable

Value

`data.frame` object with true compliers, predicted compliers in the control group, and all compliers (actual + predicted).

`neuralnet_response_model`
Modeling Responses from experimental data Using Deep NN

Description

Model Responses from all compliers (actual + predicted) in experimental data using neural network.

Usage

```
neuralnet_response_model(
  response.formula,
  exp.data,
  neuralnet.compliers,
  compl.var,
  algorithm = "rprop+",
  hidden.layer = c(4, 2),
  stepmax = 1e+08
)
```

Arguments

`response.formula`
: formula for response variable and covariates (y ~ x)

`exp.data`
: `data.frame` of experimental data.

`neuralnet.compliers`
: `data.frame` of compliers (actual + predicted) from `neuralnet_predict`.

`compl.var`
: string of compliance variable

`algorithm`
: neural network algorithm, default set to `"rprop+"`.

`hidden.layer`
: vector specifying hidden layers and number of neurons.

`stepmax`
: maximum number of steps for training model.

Value

trained response model object

`pattc_counterfactuals` *Assess Population Data counterfactuals*

Description

Create counterfactual datasets in the population for compliers and noncompliers. Then predict potential outcomes from counterfactuals.

Usage

```
pattc_counterfactuals(
  pop.data,
  response.mod,
  ID = NULL,
  cluster = NULL,
  potential.outcome = TRUE
)
```

Arguments

pop.data	population dataset
response.mod	trained model from response_model.
potential.outcome	

---

patt_deep_nn — *Estimate PATT_C using Deep NN*

Description

estimates the Population Average Treatment Effect of the Treated from experimental data with noncompliers using Deep Neural Networks.

Usage

```
patt_deep_nn(
  response.formula,
  exp.data,
  pop.data,
  treat.var,
  compl.var,
  compl.algorithm = "rprop+",
  response.algorithm = "rprop+",
  compl.hidden.layer = c(4, 2),
  response.hidden.layer = c(4, 2),
  compl.stepmax = 1e+08,
  response.stepmax = 1e+08,
  ID = NULL,
  cluster = NULL,
  bootse = FALSE,
  bootp = FALSE,
  bootn = 999
)
```

Arguments

response.formula
: formula of response variable as outcome and covariates (y ~ x)

exp.data — data.frame of experimental data. Must include binary treatment and compliance variables.

pop.data — data.frame of population data. Must include binary compliance variable

treat.var — string for treatment variable.

compl.var — string for compliance variable

compl.algorithm — string for algorithim to train neural network for compliance model. Default set to "rprop+". See (neuralnet package for available algorithms).

response.algorithm — string for algorithim to train neural network for response model. Default set to "rprop+". See (neuralnet package for available algorithms).

compl.hidden.layer — vector for specifying hidden layers and number of neurons in complier model.

response.hidden.layer — vector for specifying hidden layers and number of neurons in response model.

compl.stepmax — maximum number of steps for complier model

response.stepmax — maximum number of steps for response model

ID — string for identifier variable

cluster — string for cluster variable.

bootse — logical for bootstrapped standard erros.

bootp — logical for bootstrapped p values.

bootn — logical for number of bootstraps.

Value

results of weighted t test as PATTC estimate.

Examples

```
# load datasets
data(IND_exp_data) #experimental data
data(IND_pop_data) #population data
specify models and estimate PATTC
set.seed(123456)
pattc_neural <- patt_deep_nn(response.formula = outcome ~ age + male +
                               income + education +
                               employed + married +
                               Hindu + job_worry,
                             exp.data = expdata,
                             pop.data = popdata,
                             treat.var = "trt1",
                             compl.var = "compl1",
                             compl.algorithm = "rprop+",
                             response.algorithm = "rprop+",
                             compl.hidden.layer = c(4,2),
                             response.hidden.layer = c(4,2),
                             compl.stepmax = 1e+09,
                             response.stepmax = 1e+09,
                             ID = NULL,
                             cluster = NULL,
```

```
                              bootse = FALSE,
                              bootp = FALSE,
                              bootn = 999)

summary(pattc)
```

---

patt_ensemble *PATT_C SL Ensemble*

Description

`PATT_C_SL_Ensemble` estimates the Population Average Treatment Effect of the Treated from experimental data with noncompliers using the super learner ensemble that includes extreme gradient boosting, glmnet (elastic net regression), random forest and neural nets.

Usage

```
patt_ensemble(
  response.formula,
  exp.data,
  pop.data,
  treat.var,
  compl.var,
  createSL = TRUE,
  SL.library = NULL,
  ID = NULL,
  cluster = NULL,
  bootse = FALSE,
  bootp = FALSE,
  bootn = 999
)
```

Arguments

`response.formula`	formula for the effects of covariates on outcome variable (y ~ x).
`exp.data`	`data.frame` object for experimental data. Must include binary treatment and compliance variable.
`pop.data`	`data.frame` object for population data. Must include binary compliance variable.
`treat.var`	string for binary treatment variable.
`compl.var`	string for binary compliance variable.
`createSL`	logical. If `TRUE` will call on `create.SL` to create SL wrappers.
`ID`	string for name of identifier.
`cluster`	string for name of cluster variable.
`bootse`	logical for bootstrapped standard errors.
`bootp`	logical for bootstrapped p values.
`bootn`	number of bootstrap sample.

Value

results of weighted t test as PATTC estimate.

Examples

```
# load datasets
data(IND_exp_data) #experimental data
data(IND_pop_data) #population data
#attach SuperLearner package (model will not recognize learner if package is not loaded)
library(SuperLearner)
specify models and estimate PATTC
pattc_ensemble <- patt_ensemble(response.formula = outcome ~ age +
                                 income + education +
                                 employed + job_worry,
                               exp.data = expdata,
                              pop.data = popdata,
                              treat.var= "trt1",
                              compl.var = "compl1",
                              createSL = TRUE,
                              SL.library = c("SL.gbm.adaboost",
                                             "SL.gbm.bernoulli",
                                             "SL.glmnet"),
                              ID = NULL,
                              cluster = NULL,
                              bootse = FALSE,
                              bootp = FALSE,
                              bootn = 999)
summary(pattc)
```

`response_model` *Response model from experimental data using SL ensemble*

Description

Train response model (response variable as outcome and covariates) from all compliers (actual + predicted) in experimental data using SL ensemble.

Usage

```
response_model(
  response.formula,
  exp.data,
  compl.var,
  exp.compliers,
  family = "binomial",
  ID = NULL,
  SL.library = NULL
)
```

Arguments

exp.data	experimental dataset.
exp.compliers	data.frame object of compliers from complier_predict.
family	string for "gaussian" or "binomial".
ID	string for identifier variable.
SL.library	vector of names of ML algorithms used for ensemble model.

Value

trained response model.

ST_learner_DeepNN *S_T-learner DeepNN*

Description

ST_learner_DeepNN implements the S-learner and T-learner for estimating CATE using Deep Neural Networks. The Resilient back propagation (Rprop) algorithm is used for training neural networks.

Usage

```
ST_learner_DeepNN(
  data,
  cov.formula,
  treat.var,
  meta.learner.type,
  stepmax = 1e+05,
  nfolds = 5,
  algorithm = "rprop+",
  hidden.layer = c(4, 2),
  linear.output = FALSE
)
```

Arguments

data	data.frame object of data.
cov.formula	formula description of the model y ~ x(list of covariates).
treat.var	string for the name of treatment variable.
meta.learner.type	string specifying is the S-learner and "T.Learner" for the T-learner model.
stepmax	maximum number of steps for training model.
nfolds	number of folds for cross-validation. Currently supports up to 5 folds.
algorithm	a string for the algorithm for the neural network. Default set to rprop+, the Resilient back propagation (Rprop) with weight backtracking algorithm for training neural networks.
hidden.layer	vector of integers specifying layers and number of neurons.
linear.output	logical specifying regression (TRUE) or classification (FALSE) model.

Value

vector of CATEs estimated by the meta learners for each observation.

Examples

```
# load dataset
data(IND_exp_data)
# estimate CATEs with S Learner

slearner_nn <- ST_learner_DeepNN(cov.formula = outcome ~ age +
                                  income  +
                                  employed  + job_worry,
                                  data = expdata,
                                  treat.var = "trt1",
                                  meta.learner.type = "S.Learner",
                                  stepmax=1e+9,
                                  nfolds=5,
                                  algorithm = "rprop+",
                                  hidden.layer = c(4,2),
                                  linear.output = FALSE)
# estimate CATEs with T Learner
tlearner_nn <- ST_learner_DeepNN(cov.formula = outcome ~ age +
                                   income  +
                                   employed  + job_worry,
                                 data = expdata,
                                 treat.var = "trt1",
                                 meta.learner.type = "T.Learner",
                                 stepmax = 1e+9,
                                 nfolds = 5,
                                 algorithm = "rprop+",
                                 hidden.layer = c(2,1),
                                 linear.output = FALSE)

## Not run:
#Model may not converge with low stepmax
slearner_nn <- ST_learner_DeepNN(cov.formula = outcome ~ age +
                                   income  +
                                   employed  + job_worry,
                                 data = expdata,
                                treat.var = "trt1",
                                 meta.learner.type="S.Learner",
                                 stepmax=1e+4,
                                 nfolds=5,
                                 algorithm = "rprop+",
                                 hidden.layer=c(4,2),
                                 linear.output = FALSE)

#Other learners not supported
slearner_nn <- ST_learner_DeepNN(cov.formula = outcome ~ age +
                                  income  +
                                  employed  + job_worry,
                                  data = expdata,
                                  treat.var = "trt1",
                                  meta.learner.type="R.Learner",
                                  stepmax=1e+4,
                                  nfolds=5,
```

```
                                            algorithm = "rprop+",
                                            hidden.layer=c(4,2),
                                            linear.output = FALSE)

## End(Not run)
```

`ST_learner_ensemble` *S_T-learner Ensemble*

Description

`ST_learner_ensemble` implements the S-learner and T-learner for estimating CATE using the super learner ensemble method. The super learner in this case includes the following machine learning algorithms: extreme gradient boosting, glmnet (elastic net regression), random forest and neural nets.

Usage

```
ST_learner_ensemble(
  data,
  cov.formula,
  treat.var,
  meta.learner.type,
  learners = c("SL.glmnet", "SL.xgboost", "SL.ranger", "SL.nnet"),
  nfolds = 5
)
```

Arguments

`data`	`data.frame` object of data
`cov.formula`	formula description of the model y ~ x(list of covariates)
`treat.var`	string for the name of treatment variable.
`meta.learner.type`	string specifying is the S-learner and `"T.Learner"` for the T-learner model.
`learners`	vector for super learner ensemble that includes extreme gradient boosting, glmnet, random forest, and neural nets.
`nfolds`	number of folds for cross-validation. Currently supports up to 5 folds.

Value

vector of CATEs estimated by the meta learners for each observation.

Examples

```
# load dataset
data(IND_exp_data)
# estimate CATEs with S Learner
control <- SuperLearner::SuperLearner.CV.control(V=5)
# estimate CATEs with S Learner
slearner <- ST_learner_ensemble(cov.formula = outcome ~ age +
                                  income  +
```

```
                                  employed  + job_worry,
                                data = expdata,
                                treat.var = "trt1",
                                meta.learner.type = "S.Learner",
                                learners = c("SL.glmnet","SL.xgboost"),
                                nfolds = 5)
# estimate CATEs with T Learner

tlearner <- ST_learner_ensemble(cov.formula = outcome ~ age +
                                  income  +
                                  employed  + job_worry,
                                data = expdata,
                                treat.var = "trt1",
                                meta.learner.type = "T.Learner",
                                learners = c("SL.glmnet","SL.xgboost"),
                                nfolds = 5)

## Not run:
tlearner <- ST_learner_ensemble(cov.formula = outcome ~ age +
                                  income  +
                                  employed  + job_worry,
                                data = expdata,
                                treat.var = "trt1",
                                meta.learner.type = "R.Learner",
                                learners = c("SL.glmnet","SL.xgboost"),
                                nfolds = 5)

## End(Not run)
```

Index