Type    Package

Title   Develop Treatment Rules with Observational Data

Version 1.1.0

Description Develop and evaluate treatment rules based on: (1) the standard indirect approach of split-regression, which fits regressions separately in both treatment groups and assigns an individual to the treatment option under which predicted outcome is more desirable; (2) the direct approach of outcome-weighted-learning proposed by Yingqi Zhao, Donglin Zeng, A. John Rush, and Michael Kosorok (2012) <doi:10.1080/01621459.2012.695674>; (3) the direct approach, which we refer to as direct-interactions, proposed by Shuai Chen, Lu Tian, Tianxi Cai, and Meng-gang Yu (2017) <doi:10.1111/biom.12676>. Please see the vignette for a walkthrough of how to start with an observational dataset whose design is understood scientifically and end up with a treatment rule that is trustworthy statistically, along with an estimation of rule benefit in an independent sample.

Depends R (>= 3.2.0)

License GPL (>= 2)

Encoding UTF-8

LazyData true

RoxygenNote 7.0.2

VignetteBuilder knitr

Imports glmnet, DynTxRegime, modelObj

Suggests dplyr, knitr, rmarkdown

NeedsCompilation no

Author Jeremy Roth [cre, aut],
Noah Simon [aut]

Maintainer Jeremy Roth <jbroth@uw.edu>

Repository CRAN

Date/Publication 2020-03-20 17:40:05 UTC
**R topics documented:**

- BuildRule .................................................. 2
- CompareRulesOnValidation ........................................ 6
- EvaluateRule .................................................. 11
- obsStudyGeneExpressions ........................................ 15
- PredictRule .................................................. 16
- SplitData ..................................................... 18

**Index**

<table>
<thead>
<tr>
<th>BuildRule</th>
<th>Build a Treatment Rule</th>
</tr>
</thead>
</table>

**Description**

Perform principled development of a treatment rule (using the IPW approach to account for potential confounding) on a development dataset (i.e. training set) that is independent of datasets used for model selection (i.e. validation set) and rule evaluation (i.e. test set).

**Usage**

```r
BuildRule(
  development.data,
  study.design,
  prediction.approach,
  name.outcome,
  type.outcome,
  name.treatment,
  names.influencing.treatment = NULL,
  names.influencing.rule,
  desirable.outcome,
  rule.method = NULL,
  propensity.method,
  additional.weights = rep(1, nrow(development.data)),
  truncate.propensity.score = TRUE,
  truncate.propensity.score.threshold = 0.05,
  type.observation.weights = NULL,
  propensity.k.cv.folds = 10,
  rule.k.cv.folds = 10,
  lambda.choice = c("min", "1se"),
  OWL.lambda.seq = NULL,
  OWL.kernel = "linear",
  OWL.kparam.seq = NULL,
  OWL.cvFolds = 10,
  OWL.verbose = TRUE,
  OWL.framework.shift.by.min = TRUE,
  direct.interactions.center.continuous.Y = TRUE,
)```

direct.interactions.exclude.A.from.penalty = TRUE

Arguments

development.data
A data frame representing the *development* dataset (i.e. training set) used for building a treatment rule.

study.design
Either ‘observational’, ‘RCT’, or ‘naive’. For the observational design, the function uses inverse-probability-of-treatment observation weights (IPW) based on estimated propensity scores with predictors names.influencing.treatment; for the RCT design, the function uses IPW based on propensity scores equal to the observed sample proportions; for the naive design, all observation weights will be uniformly equal to 1.

prediction.approach
One of ‘split.regression’, ‘direct.interactions’, ‘OWL’, or ‘OWL.framework’.

name.outcome
A character indicating the name of the outcome variable in development.data.

type.outcome
Either ‘binary’ or ‘continuous’, the form of name.outcome.

name.treatment
A character indicating the name of the treatment variable in development.data.

names.influencing.treatment
A character vector (or single element) indicating the names of the variables in development.data that are expected to influence treatment assignment in the current dataset. Required for study.design=‘observational’.

names.influencing.rule
A character vector (or single element) indicating the names of the variables in development.data that may influence response to treatment and are expected to be observed in future clinical settings.

desirable.outcome
A logical equal to TRUE if higher values of the outcome are considered desirable (e.g. for a binary outcome, a 1 is more desirable than a 0). The OWL.framework and OWL prediction approaches require a desirable outcome.

rule.method
One of ‘glm.regression’, ‘lasso’, or ‘ridge’. For type.outcome=‘binary’, ‘glm.regression’ leads to logistic regression; for a type.outcome=‘continuous’, ‘glm.regression’ specifies linear regression. This is the underlying regression model used to develop the treatment rule.

propensity.method
One of ‘logistic.regression’, ‘lasso’, or ‘ridge’. This is the underlying regression model used to estimate propensity scores for study.design=‘observational’.

additional.weights
A numeric vector of observation weights that will be multiplied by IPW weights in the rule development stage, with length equal to the number of rows in development.data. This can be used, for example, to account for a non-representative sampling design or to apply an IPW adjustment for missingness. The default is a vector of 1s.
**BuildRule**

**truncate.propensity.score**
A logical variable dictating whether estimated propensity scores less than `truncate.propensity.score.threshold` away from 0 or 1 should be truncated to be no more than `truncate.propensity.score.threshold` away from 0 or 1.

**truncate.propensity.score.threshold**
A numeric value between 0 and 0.25.

**type.observation.weights**
Default is NULL, but other choices are ‘IPW.L’, ‘IPW.L.and.X’, and ‘IPW.ratio’, where L indicates `names.influencing.treatment`, X indicates `names.influencing.rule`. The default behavior is to use the ‘IPW.ratio’ observation weights (propensity score based on X divided by propensity score based on L and X) for `prediction.approach='split.regression'` and to use ‘IPW.L’ observation weights (inverse of propensity score based on L) for the ‘direct.interactions’, ‘OWL’, and ‘OWL.framework’ prediction approaches.

**propensity.k.cv.folds**
An integer specifying how many folds to use for K-fold cross-validation that chooses the tuning parameters when `propensity.method` is ‘lasso’ or ‘ridge’. Default is 10.

**rule.k.cv.folds**
An integer specifying how many folds to use for K-fold cross-validation that chooses the tuning parameter when `rule.method` is lasso or ‘ridge’. Default is 10.

**lambda.choice**
Either ‘min’ or ‘1se’, corresponding to the s argument in `predict.cv.glmnet()` from the glmnet package. Only used when `propensity.method` or `rule.method` is ‘lasso’ or ‘ridge’. Default is ‘min’.

**OWL.lambda.seq**
Used when `prediction.approach='OWL'`, a numeric vector that corresponds to the lambdas argument in the `owl()` function from the DynTxRegime package. Defaults to `2^seq(-5,5,1)`.

**OWL.kernel**
Used when `prediction.approach='OWL'`, a character equal to either ‘linear’ or ‘radial’. Corresponds to the kernel argument in the `owl()` function from the DynTxRegime package. Default is ‘linear’.

**OWL.kparam.seq**
Used when `prediction.approach='OWL'` and `OWL.kernel='radial'`. Corresponds to the kparam argument in the `owl()` function from the DynTxRegime package. Defaults to `2^seq(-10,10,1)`.

**OWL.cvFolds**
Used when `prediction.approach='OWL'`, an integer corresponding to the cvFolds argument in the `owl()` function from the DynTxRegime package. Defaults to 10.

**OWL.verbose**
Used when `prediction.approach='OWL'`, a logical corresponding to the verbose argument in the `owl()` function from the DynTxRegime package. Defaults to TRUE.

**OWL.framework.shift.by.min**
Logical, set to TRUE by default in recognition of our empirical observation that, with a continuous outcome, OWL framework performs far better in simulation studies when the outcome was shifted to have a minimum of just above 0.

**direct.interactions.center.continuous.Y**
Logical, set to TRUE by default in recognition of our empirical observation that, with a continuous outcome, direct-interactions performed far better in simulation studies when the outcome was mean-centered.
direct.interactions.exclude.A.from.penalty

Logical, set to TRUE by default in recognition of our empirical observation that, with a continuous outcome and lasso/ridge used specified as the rule.method, direct-interactions performed far better in simulation studies when the coefficient corresponding to the treatment variable was excluded from the penalty function.

Value

A list with some combination of the following components (depending on specified prediction.approach)

- type.outcome: The type.outcome specified above (used by other functions that are based on BuildRule())
- prediction.approach: The prediction.approach specified above (used by other functions that are based on BuildRule())
- rule.method: The rule.method specified above (used by other functions that are based on BuildRule())
- lambda.choice: The lambda.choice specified above (used by other functions that are based on BuildRule())
- propensity.score.object: A list containing the relevant regression object from propensity score estimation. The list has two elements for type.observation.weights='IPW.ratio' (the default for prediction.approach='split.regression'), has one element for type.observation.weights='IPW.L' (the default for 'OWL', 'OWL.framework' and 'direct.interactions'), has one element when type.observation.weights='IPW.L.and.X', and is simply equal to NA if study.design='RCT' (in which case propensity score would just be the inverse of sample proportion receiving treatment).
- owl.object: For prediction.approach='OWL' only, the object returned by the owl() function in the DynTxRegime package.
- observation.weights: The observation weights used for estimating the treatment rule
- rule.object: For prediction.approach='OWL.framework' or prediction.approach='direct.interactions', the regression object returned from treatment rule estimation (to which the coef() function could be applied, for example)
- rule.object.control: For prediction.approach='split.regression' the regression object returned from treatment rule estimation (to which the coef() function could be applied, for example) that estimates the outcome variable for individuals who do not receive treatment.
- rule.object.treatment: For prediction.approach='split.regression' the regression object returned from treatment rule estimation (to which the coef() function could be applied, for example) that estimates the outcome variable for individuals who do receive treatment.

References


• Jeremy Roth and Noah Simon (2019). Using propensity scores to develop and evaluate treatment rules with observational data (Manuscript in progress)

• Jeremy Roth and Noah Simon (2019). Elucidating outcome-weighted learning and its comparison to split-regression: direct vs. indirect methods in practice. (Manuscript in progress)

Examples

```r
set.seed(123)
ex <- SplitData(data=obsStudyGeneExpressions,
    n.sets=3, split.proportions=c(0.5, 0.25, 0.25))
development.data <- ex[ex$partition == "development",]
one.rule <- BuildRule(development.data=development.data,
    study.design="observational",
    prediction.approach="split.regression",
    name.outcome="no_relapse",
    type.outcome="binary",
    desirable.outcome=TRUE,
    name.treatment="intervention",
    names.influencing.treatment=c("prognosis", "clinic", "age"),
    names.influencing.rule=c("age", paste0("gene_", 1:10)),
    propensity.method="logistic.regression",
    rule.method="glm.regression")
coef(one.rule$rule.object.control)
coef(one.rule$rule.object.treatment)
```

**CompareRulesOnValidation**

*Build treatment rules on a development dataset and evaluate performance on an independent validation dataset*

**Description**

In many practical settings, BuildRule() has limited utility because it requires the specification of a single value in its prediction.approach argument (even if there is no prior knowledge about which of the split-regression, OWL framework, and direct-interactions approaches will perform best) and a single value for the ‘propensity.score’ and ‘rule.method’ arguments (even if there is no prior knowledge about whether standard or penalized GLM will perform best). CompareRulesOnValidation() supports model selection in these settings by essentially looping over calls to BuildRule() for different combinations of split-regression/OWL framework/direct-interactions and standard/lasso/ridge regression to simultaneously build the rules on a development dataset and evaluate them on an independent validation dataset.
Usage

```r
CompareRulesOnValidation(
  development.data,
  validation.data,
  vec.approaches = c("split.regression", "OWL.framework", "direct.interactions"),
  vec.rule.methods = c("glm.regression", "lasso", "ridge"),
  vec.propensity.methods = "logistic.regression",
  study.design.development,
  name.outcome.development,
  type.outcome.development,
  name.treatment.development,
  names.influencing.treatment.development,
  names.influencing.rule.development,
  desirable.outcome.development,
  additional.weights.development = rep(1, nrow(development.data)),
  study.design.validation = study.design.development,
  name.outcome.validation = name.outcome.development,
  type.outcome.validation = type.outcome.development,
  name.treatment.validation = name.treatment.development,
  names.influencing.treatment.validation = names.influencing.treatment.development,
  names.influencing.rule.validation = names.influencing.rule.development,
  desirable.outcome.validation = desirable.outcome.development,
  clinical.threshold.validation = 0,
  propensity.method.validation = "logistic.regression",
  additional.weights.validation = rep(1, nrow(validation.data)),
  truncate.propensity.score = TRUE,
  truncate.propensity.score.threshold = 0.05,
  type.observation.weights = NULL,
  propensity.k.cv.folds = 10,
  rule.k.cv.folds = 10,
  lambda.choice = c("min", "1se"),
  OWL.lambda.seq = NULL,
  OWL.kernel = "linear",
  OWL.kparam.seq = NULL,
  OWL.cvFolds = 10,
  OWL.verbose = TRUE,
  OWL.framework.shift.by.min = TRUE,
  direct.interactions.center.continuous.Y = TRUE,
  direct.interactions.exclude.A.from.penalty = TRUE,
  bootstrap.CI = FALSE,
  bootstrap.CI.replications = 100
)
```

Arguments

development.data

A data frame representing the *development* dataset used to build treatment rules.
validation.data
A data frame representing the independent validation dataset used to estimate the performance of treatment rules built on the development dataset.

vec.approaches
A character vector (or element) indicating the values of the prediction.approach to be used for building the rule with BuildRule(). Default is c("split.regression", "OWL.framework")

vec.rule.methods
A character vector (or element) indicating the values of the rule.method to be used for building the rule with BuildRule(). Default is c("glm.regression", "lasso", "ridge").

vec.propensity.methods
A character vector (or element) indicating the values of propensity.method to be used for building the rule with BuildRule(). Default is 'logistic.regression' to allow for estimation of bootstrap-based CIs.

study.design.development
Either 'observational', 'RCT', or 'naive', representing the study design on the development dataset. For the observational design, the function will use inverse-probability-of-treatment observation weights (IPW) based on estimated propensity scores with predictors names.influencing.treatment; for the RCT design, the function will use IPW based on propensity scores equal to the observed sample proportions; for the naive design, all observation weights will be uniformly equal to 1.

name.outcome.development
A character indicating the name of the outcome variable in development.data.

type.outcome.development
Either 'binary' or 'continuous', the form of name.outcome.development.

name.treatment.development
A character indicating the name of the treatment variable in development.data.

names.influencing.treatment.development
A character vector (or element) indicating the names of the variables in development.data that are expected to influence treatment assignment in the current dataset. Required for study.design.development='observational'.

names.influencing.rule.development
A character vector (or element) indicating the names of the variables in development.data that may influence response to treatment and are expected to be observed in future clinical settings.

desirable.outcome.development
A logical equal to TRUE if higher values of the outcome on development.data are considered desirable (e.g. for a binary outcome, a 1 is more desirable than a 0). The OWL.framework and OWL prediction approaches require a desirable outcome.

additional.weights.development
A numeric vector of observation weights that will be multiplied by IPW weights in the rule development stage, with length equal to the number of rows in development.data. This can be used, for example, to account for a non-representative sampling design or an IPW adjustment for missingness. The default is a vector of 1s.

study.design.validation
Either 'observational', 'RCT', or 'naive', representing the study design on the development dataset. Default is the value of study.design.development.
name.outcome.validation
  A character indicating the name of the outcome variable in validation.data. 
  Default is the value of name.outcome.development.

type.outcome.validation
  Either ‘binary’ or ‘continuous’, the form of name.outcome.validation. De- 
  fault is the value of type.outcome.development.

name.treatment.validation
  A character indicating the name of the treatment variable in validation.data. 
  Default is the value of name.treatment.development

names.influencing.treatment.validation
  A character vector (or element) indicating the names of the variables in 
  validation.data that are expected to influence treatment assignment in validation.data. Re- 
  quired for Required for study.design.validation='observational'. Default 
  is the value of names.influencing.treatment.development.

names.influencing.rule.validation
  A character vector (or element) indicating the names of the variables in validation.data 
  that may influence response to treatment and are expected to be observed in fu- 
  ture clinical settings. Default is the value of names.influencing.rule.development

desirable.outcome.validation
  A logical equal to TRUE if higher values of the outcome on validation.data 
  are considered desirable (e.g. for a binary outcome, a 1 is more desirable than 
  a 0). The OWL.framework and OWL prediction approaches require a desirable 
  outcome. Default is the value of desirable.outcome.development

clinical.threshold.validation
  A numeric equal to a positive number above which the predicted outcome under 
  treatment must be superior to the predicted outcome under control for treatment 
  to be recommended. Only used when BuildRuleObject was specified and de- 
  rived from the split-regression or direct-interactions approach. Default is 0.

propensity.method.validation
  One of ‘logistic.regression’, ‘lasso’, or ‘ridge’. This is the underlying regression 
  model used to estimate propensity scores (for study.design='observational' on 
  validation.data. If bootstrap.CI=TRUE, then propensity.method must be 
  ‘logistic.regression’. Default is ‘logistic.regression’ to allow for estimation of 
  bootstrap-based CIs.

additional.weights.validation
  A numeric vector of observation weights that will be multiplied by IPW weights 
  in the rule evaluation stage, with length equal to the number of rows in validation.data. 
  This can be used, for example, to account for a non-representative sampling de- 
  sign or an IPW adjustment for missingness. The default is a vector of 1s.

truncate.propensity.score
  A logical variable dictating whether estimated propensity scores less than truncate.propensity.score.
  score threshold away from 0 or 1 should be truncated to be truncate.propensity.score.threshold 
  away from 0 or 1.

truncate.propensity.score.threshold
  A numeric value between 0 and 0.25.

type.observation.weights
  Default is NULL, but other choices are ‘IPW.L’, ‘IPW.L.and.X’, and ‘IPW.ratio’, 
  where L indicates the names.influencing.treatment variables, X indicates
the names.influencing.rule variables. The default behavior is to use the
‘IPW.ratio’ observation weights (propensity score based on X divided by propen-
sity score based on L and X) for prediction.approach=’split.regression’ and
to use ‘IPW.L’ observation weights (inverse of propensity score based on L) for
the ‘direct.interactions’, ‘OWL’, and ‘OWL.framework’ prediction approaches.

propensity.k.cv.folds
An integer specifying how many folds to use for K-fold cross-validation that
chooses the tuning parameter when propensity.method is ‘lasso’ or ‘ridge’.
Default is 10.

rule.k.cv.folds
An integer specifying how many folds to use for K-fold cross-validation that
chooses the tuning parameter when rule.method is ‘lasso’ or ‘ridge’. Default
is 10.

lambda.choice
Either ‘min’ or ‘1se’, corresponding to the s argument in predict.cv.glmnet()
from the glmnet package. Only used when propensity.method or rule.method
is ‘lasso’ or ‘ridge’. Default is ‘min’.

OWL.lambda.seq
Used when prediction.approach=’OWL’, a numeric vector that corresponds
to the lambdas argument in the owl() function from the DynTxRegime package.
Defaults to 2^{seq(-5,5,1)}.

OWL.kernel
Used when prediction.approach=’OWL’, a character equal to either ‘linear’
or ‘radial’. Corresponds to the kernel argument in the owl() function from the
DynTxRegime package. Default is ‘linear’.

OWL.kparam.seq
Used when prediction.approach=’OWL’ and OWL.kernel=’radial’. Corre-
sponds to the kparam argument in the owl() function from the DynTxRegime
package. Defaults to 2^{seq(-10,10,1)}.

OWL.cvFolds
Used when prediction.approach=’OWL’, an integer corresponding to the cvFolds
argument in the owl() function from the DynTxRegime package. Defaults to 10.

OWL.verbose
Used when prediction.approach=’OWL’, a logical corresponding to the verbose
argument in the owl() function from the DynTxRegime package. Defaults to
TRUE.

OWL.framework.shift.by.min
Logical, set to TRUE by default in recognition of our empirical observation that,
with a continuous outcome, OWL framework performs far better in simulation
studies when the outcome was shifted to have a minimum of just above 0.

direct.interactions.center.continuous.Y
Logical, set to TRUE by default in recognition of our empirical observation that,
with a continuous outcome, direct-interactions performed far better in simula-
tion studies when the outcome was mean-centered.

direct.interactions.exclude.A.from.penalty
Logical, set to TRUE by default in recognition of our empirical observation that,
with a continuous outcome and lasso/ridge used specified as the rule.method,
direct-interactions performed far better in simulation studies when the coeffi-
cient corresponding to the treatment variable was excluded from the penalty
function.

bootstrap.CI
Logical indicating whether the ATE/ABR estimates on the validation set should
be accompanied by 95% confidence intervals based on the bootstrap. Default is
FALSE.
bootstrap.CI.replications

An integer specifying how many bootstrap replications should underlie the computed CIs. Default is 1000.

Value

A list with components:

- list.summaries: A list with number of elements equal to the length of vec.approaches. Each element is a matrix that, for a given prediction approach, shows estimated rule performance with 5 columns if bootstrap.CI=FALSE (number of test-positives, number of test-negatives, ATE in test-positives, ATE in test-negatives, ABR) for the different combinations of vec.rule.methods or 9 columns if bootstrap.CI=TRUE (those same 5 summaries plus the bounds for 95% CIs for ATE in test-positives and ATE in test-negatives) and, in the rows, the vec.propensity.methods in addition to the two naive rules (treating all observations and treating no observations).

- list.rules: A list with number of elements equal to the length of vec.approaches. Each element is another list that, for a given prediction approach, stores the object returned by BuildRule() for the different combinations of vec.rule.methods and vec.propensity.methods in the rows.

Examples

```r
set.seed(123)
example.split <- SplitData(data=obsStudyGeneExpressions, 
    n.sets=3, split.proportions=c(0.5, 0.25, 0.25))
development.data <- example.split[example.split$partition == "development", ]
validation.data <- example.split[example.split$partition == "validation", ]
model.selection <- CompareRulesOnValidation(development.data=development.data, 
    validation.data=validation.data, 
    study.design.development="observational", 
    vec.approaches=c("split.regression", "OWL.framework", "direct.interactions"), 
    vec.rule.methods=c("glm.regression", "lasso"), 
    vec.propensity.methods="logistic.regression", 
    name.outcome.development="no_relapse", 
    type.outcome.development="binary", 
    name.treatment.development="intervention", 
    names.influencing.treatment.development=c("prognosis", "clinic", "age"), 
    names.influencing.rule.development=c("age", paste0("gene", 1:10)), 
    desirable.outcome.development=TRUE)
model.selection$list.summaries$split.regression
```

Description

Perform principled evaluation of a treatment rule (using the IPW approach to account for potential confounding) on a dataset that is independent of the development dataset on which the rule was developed, either to perform model selection (with a validation dataset) or to obtain trustworthy estimates of performance for a pre-specified treatment rule (with an evaluation dataset).
EvaluateRule

Usage

EvaluateRule(
  evaluation.data,
  BuildRule.object = NULL,
  B = NULL,
  study.design,
  name.outcome,
  type.outcome,
  desirable.outcome,
  separate.propensity.estimation = TRUE,
  clinical.threshold = 0,
  name.treatment,
  names.influencing.treatment,
  names.influencing.rule,
  propensity.method = NULL,
  show.treat.all = TRUE,
  show.treat.none = TRUE,
  truncate.propensity.score = TRUE,
  truncate.propensity.score.threshold = 0.05,
  observation.weights = NULL,
  additional.weights = rep(1, nrow(evaluation.data)),
  lambda.choice = c("min", "1se"),
  propensity.k.cv.folds = 10,
  bootstrap.CI = FALSE,
  bootstrap.CI.replications = 1000,
  bootstrap.type = "basic"
)

Arguments

evaluation.data
  A data frame representing the *validation* or *evaluation* dataset used to esti-
  mate the performance of a rule that was developed on an independent develop-
  ment dataset.

BuildRule.object
  The object returned by the 
  BuildRule() function. Defaults to NULL but is
  required if a treatment rule is not provided in the B argument. Only one of
  BuildRule.object and B should be specified.

B
  A numeric vector representing a pre-specified treatment rule, which must have
  length equal to the number of rows in evaluation.data and elements equal
  to 0/FALSE indicating no treatment and 1/TRUE indicating treatment. Defaults
  to NULL but is required if BuildRule.object is not specified. Only one of
  BuildRule.object and B should be specified.

study.design
  Either ‘observational’, ‘RCT’, or ‘naive’. For the observational design, the
  function will use inverse-probability-of-treatment observation weights (IPW)
  based on estimated propensity scores with predictors names.influencing.treatment;
  for the RCT design, the function will use IPW based on propensity scores equal to
the observed sample proportions; for the naive design, all observation weights will be uniformly equal to 1.

name.outcome A character indicating the name of the outcome variable in evaluation.data.

type.outcome Either ‘binary’ or ‘continuous’, the form of name.outcome.

desirable.outcome A logical equal to TRUE if higher values of the outcome are considered desirable (e.g. for a binary outcome, 1 is more desirable than 0). The OWL.framework and OWL approaches to treatment rule estimation require a desirable outcome.

separate.propensity.estimation A logical equal to TRUE if propensity scores should be estimated separately in the test-positives and test-negatives subpopulations and equal to FALSE if propensity scores should be estimated in the combined sample. Default is TRUE.

clinical.threshold A numeric equal to a positive number above which the predicted outcome under treatment must be superior to the predicted outcome under control for treatment to be recommended. Only used when BuildRuleObject was specified and derived from the split-regression or direct-interactions approach. Default is 0.

name.treatment A character indicating the name of the treatment variable in evaluation.data.

names.influencing.treatment A character vector (or element) indicating the names of the variables in evaluation.data that are expected to influence treatment assignment in the current dataset. Required for study.design='observational'.

names.influencing.rule A character vector (or element) indicating the names of the variables in evaluation.data that may influence response to treatment and are expected to be observed in future clinical settings.

propensity.method One of 'logistic.regression', 'lasso', or 'ridge'. This is the underlying regression model used to estimate propensity scores (for study.design='observational'. If bootstrap.CI=TRUE, then propensity.method must be 'logistic.regression'. Defaults to NULL.

show.treat.all A logical variable dictating whether summaries for the naive rule that assigns treatment to all observations are reported, which help put the performance of the estimated treatment rule in context. Default is TRUE.

show.treat.none A logical variable dictating whether summaries for the naive rule that assigns treatment to no observations are reported, which help put the performance of the estimated treatment rule in context. Default is TRUE.

truncate.propensity.score A logical variable dictating whether estimated propensity scores less than truncate.propensity.score.threshold away from 0 or 1 should be truncated to be truncate.propensity.score.threshold away from 0 or 1.

truncate.propensity.score.threshold A numeric value between 0 and 0.25.
observation.weights
A numeric vector equal to the number of rows in evaluation.data that provides observation weights to be used in place of the IPW weights estimated with propensity.method. Defaults to NULL. Only one of the propensity.method and observation.weights should be specified.

additional.weights
A numeric vector of observation weights that will be multiplied by IPW weights in the rule evaluation stage, with length equal to the number of rows in evaluation.data. This can be used, for example, to account for a non-representative sampling design or to apply an IPW adjustment for missingness. The default is a vector of 1s.

lambda.choice
Either ‘min’ or ‘1se’, corresponding to the s argument in predict.cv.glmnet() from the glmnet package; only used when propensity.method or rule.method is ‘lasso’ or ‘ridge’. Default is ‘min’.

propensity.k.cv.folds
An integer dictating how many folds to use for K-fold cross-validation that chooses the tuning parameter when propensity.method is ‘lasso’ or ‘ridge’. Default is 10.

bootstrap.CI
Logical indicating whether the ATE/ABR estimates returned by EvaluateRule() should be accompanied by 95% confidence intervals based on the bootstrap. Default is FALSE.

bootstrap.CI.replications
An integer specifying how many bootstrap replications should underlie the computed CIs. Default is 1000.

bootstrap.type
One character element specifying the type of bootstrap CI that should be computed. Currently the only supported option is bootstrap.type=’basic’, but this may be expanded in the future.

Value
A list with the following components

- recommended.treatment: A numeric vector of 0s and 1s, with length equal to the number of rows in evaluation.data, where a 0 indicates treatment is not recommended and a 1 indicates treatment is recommended for the corresponding observation in evaluation.data.

- fit.object: A list consisting of one of the following: the propensity scores estimated in the test-positives and in the test-negatives (if separate.propensity.estimation=TRUE, study.design=’observational’, and observation.weights=NULL); the propensity scores estimated in the combined sample (if separate.propensity.estimation=FALSE, study.design=’observational’, and observation.weights=NULL); and simply is simply null if study.design=’RCT’ (in which case propensity score would just be the inverse of the sample proportion receiving treatment)

- summaries: a matrix with columns reporting the following summaries of treatment rule performance: the number of observations in evaluation.data recommended to receive treatment (n.positives); the estimated average treatment effect among those recommended to receive treatment (ATE.positives); the number of observations in evaluation.data recommended to not receive treatment (n.negatives); the estimated average treatment effect among those recommended to not receive treatment (ATE.negatives); the estimated average benefit...
of using the rule, with the weighted average of ATE.positives and -1 * ATE.negatives where
weights are the proportions of test-positives and test-negatives (ABR). If bootstrap.CI=TRUE,
then 4 additional columns are included, showing the lower bound (LB) and upper bound (UB)
of the 95% CIs for ATE.positives and ATE.negatives.

Examples

```r
set.seed(123)
example.split <- SplitData(data=obsStudyGeneExpressions,
n.sets=3, split.proportions=c(0.5, 0.25, 0.25))
development.data <- example.split[example.split$partition == "development",]
validation.data <- example.split[example.split$partition == "validation",]
one.rule <- BuildRule(development.data=development.data,
study.design="observational",
prediction.approach="split.regression",
name.outcome="no_relapse",
type.outcome="binary",
desirable.outcome=TRUE,
name.treatment="intervention",
names.influencing.treatment=c("prognosis", "clinic", "age"),
names.influencing.rule=c("age", paste0("gene_", 1:10)),
propensity.method="logistic.regression",
rule.method="glm.regression"
)
split.validation <- EvaluateRule(evaluation.data=validation.data,
BuildRule.object=one.rule,
study.design="observational",
name.outcome="no_relapse",
type.outcome="binary",
desirable.outcome=TRUE,
name.treatment="intervention",
names.influencing.treatment=c("prognosis", "clinic", "age"),
names.influencing.rule=c("age", paste0("gene_", 1:10)),
propensity.method="logistic.regression",
bootstrap.CI=FALSE)
```

---

**obsStudyGeneExpressions**

*Simulated dataset for package DevTreatRule*

**Description**

Simulated observational dataset reporting an indicator of remaining relapse-free at the end of the
study period (clinical outcome) along with 14 additional baseline characteristics that may influence
either treatment assignment, response to treatment, or both.

**Usage**

obsStudyGeneExpressions
**Format**

A data frame with 5000 rows and 15 variables:

- `no_relapse`: Binary indicator of remaining relapse-free at the end of study period
- `intervention`: Binary indicator of receiving treatment
- `prognosis`: Two-level factor variable indicating whether clinic-specific measure of prognosis is “poor” or ‘good’
- `clinic`: Ten-level factor variable indicating which of 10 clinics each observation attended
- `age`: Numeric variable indicating an individual’s age (ranges from 40 to 65)
- `gene_1`: Numeric variable representing the expression level of specific gene_1
- `gene_2`: Numeric variable representing the expression level of specific gene_2
- `gene_3`: Numeric variable representing the expression level of specific gene_3
- `gene_4`: Numeric variable representing the expression level of specific gene_4
- `gene_5`: Numeric variable representing the expression level of specific gene_5
- `gene_6`: Numeric variable representing the expression level of specific gene_6
- `gene_7`: Numeric variable representing the expression level of specific gene_7
- `gene_8`: Numeric variable representing the expression level of specific gene_8
- `gene_9`: Numeric variable representing the expression level of specific gene_9
- `gene_10`: Numeric variable representing the expression level of specific gene_10

---

**PredictRule**  
*Get the treatment rule implied by BuildRule()*

**Description**

Map the object returned by BuildRule() to the treatment rule corresponding to a particular dataset

**Usage**

```r
PredictRule(
  BuildRule.object,
  new.X,
  desirable.outcome = NULL,
  clinical.threshold = 0,
  return.predicted.response = FALSE
)
```
Arguments

BuildRule.object
The object returned by the BuildRule() function.

new.X
A data frame representing the dataset for which the treatment rule is desired.

desirable.outcome
A logical equal to TRUE if higher values of the outcome are considered desirable (e.g. for a binary outcome, 1 is more desirable than 0). The OWL.framework and OWL approaches to treatment rule estimation require a desirable outcome.

clinical.threshold
A numeric equal a positive number above which the predicted outcome under treatment must be superior to the predicted outcome under control for treatment to be recommended. Only used when BuildRuleObject was specified and derived from the split-regression or direct-interactions approach. Defaults to 0.

return.predicted.response
logical indicating whether the predicted response variable (for split.regression, OWL.framework, and OWL approaches) or score function (for direct.interactions) should be returned in addition to its mapping to a binary treatment recommendation. Default is FALSE.

Value

- If return.predicted.response=FALSE (the default), then the single object returned is a numeric vector of 0s and 1s, with length equal to the number of rows in new.X, where a 0 indicates treatment is not recommended and a 1 indicates treatment is recommended for the corresponding observation in new.X.

- If return.predicted.response=TRUE, then the object returned is a list with some combination of the following components (depending on which prediction approach underlies the BuildRule.object).

  - recommended.treatment: A numeric vector of 0s and 1s, with length equal to the number of rows in new.X, where a 0 indicates treatment is not recommended and a 1 indicates treatment is recommended for the corresponding observation in new.X.

  - predicted.outcome: A numeric vector showing the predicted values of the score function mapped to recommended.treatment. Only returned if return.predicted.response=TRUE and the approach underlying BuildRule.object was 'direct.interactions'.

  - predicted.outcome.under.control: A numeric vector showing the predicted values of the outcome under no treatment which, along with predicted.outcome.under.treatment, corresponds to recommended.treatment. Only returned if return.predicted.response=TRUE and the approach underlying BuildRule.object was 'split.regression'.

  - predicted.outcome.under.treatment: A numeric vector showing the predicted values of the outcome under treatment which, along with predicted.outcome.under.control, corresponds to recommended.treatment. Only returned if return.predicted.response=TRUE and the approach underlying BuildRule.object was 'split.regression'.

  - predicted.treatment.prob: A numeric vector showing the predicted treatment probability that corresponds to recommended.treatment. Only returned if return.predicted.response=TRUE and the approach underlying BuildRule.object was 'OWL.framework'.

Examples

set.seed(123)
example.split <- SplitData(data=obsStudyGeneExpressions, 
n.sets=3, split.proportions=c(0.5, 0.25, 0.25))
development.data <- example.split[example.split$partition == "development",]
validation.data <- example.split[example.split$partition == "validation",]
one.rule <- BuildRule(development.data=development.data, 
  study.design="observational", 
  prediction.approach="split.regression", 
  name.outcome="no_relapse", 
  type.outcome="binary", 
  desirable.outcome=TRUE, 
  name.treatment="intervention", 
  names.influencing.treatment=c("prognosis", "clinic", "age"), 
  names.influencing.rule=c("age", paste0("gene_", 1:10)), 
  propensity.method="logistic.regression", 
  rule.method="glm.regression")
one.prediction <- PredictRule(BuildRule.object=one.rule, 
  new.X=validation.data[, c("age", paste0("gene_", 1:10))], 
  desirable.outcome=TRUE, 
  clinical.threshold=0)
table(one.prediction)

SplitData

Partition a dataset into independent subsets

Description

To get a trustworthy estimate of how a developed treatment rule will perform in independent samples drawn from the same population, it is critical that rule development be performed independently of rule evaluation. Further, it is common to perform model selection to settle on the form of the developed treatment rule and, in this case, it is essential that the ultimately chosen treatment rule is also evaluated on data that did not inform any stage of the model-building. The SplitData() function partitions a dataset so rule development/validation/evaluation (or development/evaluation if there is no model selection) can quickly be performed on independent datasets. This function is only appropriate for the simple setting where the rows in a given dataset are independent of one another (e.g. the same individuals are not represented with multiple rows).

Usage

SplitData(data, n.sets = c(3, 2), split.proportions = NULL)

Arguments

data A data frame representing the *development* dataset used for building a treatment rule

n.sets A numeric/integer equal to either 3 (if a development/validation/evaluation partition is desired) or 2 (if there is no model-selection and only a development/evaluation partition is desired).
SplitData

split.proportions
A numeric vector with length equal to n.sets, providing the proportion of observations in data that should be assigned to the development/evaluation partitions (if n.sets=2) or to the development/validation/evaluation partitions (if n.sets=3). The entries must sum to 1.

Value
A data.frame equal to data with an additional column named 'partition', which is a factor variable with levels equal to 'development' and 'evaluation' (if n.sets=2) or to 'development', 'validation', and 'evaluation' (if n.sets=3).

Examples
set.seed(123)
example.split <- SplitData(data=obsStudyGeneExpressions,
                           n.sets=3, split.proportions=c(0.5, 0.25, 0.25))
table(example.split$partition)
Index

*T* Topic **datasets**
  - obsStudyGeneExpressions, 15
BuildRule, 2
CompareRulesOnValidation, 6
EvaluateRule, 11
obsStudyGeneExpressions, 15
PredictRule, 16
SplitData, 18