Package 'DynTxRegime'

November 24, 2023

Type Package

Title Methods for Estimating Optimal Dynamic Treatment Regimes

Version 4.15

Date 2023-11-24

Author S. T. Holloway, E. B. Laber, K. A. Linn, B. Zhang, M. Davidian, and A. A. Tsiatis

Maintainer Shannon T. Holloway <shannon.t.holloway@gmail.com>

Description Methods to estimate dynamic treatment regimes using Interactive Q-Learning, Q-Learning, weighted learning, and value-search methods based on Augmented Inverse Probability Weighted Estimators and Inverse Probability Weighted Estimators. Dynamic Treatment Regimes: Statistical Methods for Precision Medicine, Tsiatis, A. A., Davidian, M. D., Holloway, S. T., and Laber, E. B., Chapman & Hall/CRC Press, 2020, ISBN:978-1-4987-6977-8.

License GPL-2

Depends methods, modelObj, stats

Suggests MASS, rpart, nnet

Imports kernlab, rgenoud, dfoptim

NeedsCompilation no

Repository CRAN

Encoding UTF-8

RoxygenNote 7.2.1

Collate 'A_generics.R' 'A_List.R' 'A_DecisionPointList.R' 'A_OptimalInfo.R' 'A_OptimalObj.R' 'A_DynTxRegime.R' 'A_ModelObjSubset.R' 'A_SubsetList.R' 'A_ModelObj_SubsetList.R' 'A_ModelObj_DecisionPointList.R' 'A_newModelObjSubset.R' 'B_TxInfoBasic.R' 'B_TxInfoFactor.R' 'B_TxInfoInteger.R' 'B_TxObj.R' 'B_TxInfoNoSubsets.R' 'B_TxSubset.R' 'B_TxSubsetInteger.R' 'B_TxSubsetFactor.R' 'B_TxInfoWithSubsets.R' 'B_TxInfoList.R' 'C_TypedFit.R' 'C_TypedFit_SubsetList.R' 'C_TypedFit_fSet.R' 'C_TypedFitObj.R' 'D_OutcomeNoFit.R' 'D_newModel.R' 'D_OutcomeSimpleFit.R' 'D_OutcomeSimpleFit_fSet.R' 'D_OutcomeIterateFit.R'

'D_OutcomeSimpleFit_SubsetList.R' 'D_OutcomeObj.R'
'E_class_QLearn.R' 'E_class_IQLearnSS.R' 'E_class_IQLearnFS.R'
'E_class_IQLearnFS_C.R' 'E_class_IQLearnFS_ME.R'
'E_class_IQLearnFS_VHet.R' 'E_iqLearnFSC.R' 'E_iqLearnFSM.R'
'E_iqLearnFSV.R' 'E_iqLearnSS.R' 'E_qLearn.R'
'F_PropensityFit.R' 'F_PropensityFit_fSet.R'
'F_PropensityFit_SubsetList.R' 'F_PropensityObj.R' 'G_Regime.R'
'G_RegimeObj.R' 'H_class_OptimalSeq.R'
'H_class_OptimalSeqCoarsened.R' 'H_class_OptimalSeqMissing.R'
'H_optimalSeq.R' 'I_ClassificationFit.R'
'I_ClassificationFit_SubsetList.R' 'I_ClassificationFit_fSet.R'
'I_ClassificationObj.R' 'J_class_OptimalClass.R'
'J_optimalClass.R' 'K_Kernel.R' 'K_MultiRadialKernel.R'
'K_RadialKernel.R' 'K_PolyKernel.R' 'K_LinearKernel.R'
'K_KernelObj.R' 'L_Surrogate.R' 'L_ExpSurrogate.R'
'L_HingeSurrogate.R' 'L_HuberHingeSurrogate.R'
'L_LogitSurrogate.R' 'L_SmoothRampSurrogate.R'
'L_SqHingeSurrogate.R' 'M_MethodObject.R' 'M_OptimBasic.R'
'M_OptimKernel.R' 'M_OptimObj.R' 'N_CVBasic.R' 'N_CVInfo.R'
'N_CVInfoLambda.R' 'N_CVInfokParam.R' 'N_CVInfo2Par.R'
'N_CVInfoObj.R' 'N_OptimStep.R' 'O_LearningObject.R'
'O_Learning.R' 'O_LearningMulti.R' 'P_class_.owl.R'
'P_class_OWL.R' 'P_owl.R' 'Q_class_.rwl.R' 'Q_class_RWL.R'
'Q_rwl.R' 'R_class_BOWLBasic.R' 'R_class_BOWL.R' 'R_bowl.R'
'S_class_.earl.R' 'S_class_EARL.R' 'S_earl.R'
'checkFSetAndOutcomeModels.R' 'checkFSetAndPropensityModels.R'
'checkInputs.R' 'internalTest.R' 'titleIt.R'

Date/Publication 2023-11-24 17:10:05 UTC

R topics documented:

`bmiData` *Adolescent BMI dataset (generated toy example)*

Description

A dataset generated to mimic data from a two-stage randomized clinical trial that studied the effect of meal replacement shakes on adolescent obesity. The dataset contains the following covariates collected at the start of the first stage: "gender," "race," "parentBMI," and "baselineBMI." At the second-stage, "month4BMI" was collected. Variables "A1" and "A2" are the randomized treatments at stages one and two, and "month12BMI" is the primary outcome collected at the end of stage two.

Format

A matrix with rows corresponding to patients.

Source

Generated by Kristin A. Linn in R

`bowl` *Backwards Outcome Weighted Learning.*

Description

Function performs a single step of the bowl method. Multiple decision points can be analyzed by repeated calls, as is done for qLearn() and optimalClass().

Usage

```
bowl(
  ...,
  moPropen,
  data,
  reward,
  txName,
  regime,
  response,
  BOWLObj = NULL,
  lambdas = 2,
  cvFolds = 0L,
  kernel = "linear",
  kparam = NULL,
  fSet = NULL,
  surrogate = "hinge",
  verbose = 2L
)
```

Arguments

`...`	Used primarily to require named input. However, inputs for the optimization methods can be sent through the ellipsis. If surrogate is hinge, the optimization method is dfoptim::hjk(). For all other surrogates, stats::optim() is used.
`moPropen`	An object of class modelObj or modelObjSubset, which defines the model and R methods to be used to obtain parameter estimates and predictions for the propensity for tx. See ?moPropen for details.
`data`	A data frame of the covariates and tx histories.
`reward`	The response vector.
`txName`	A character object. The column header of *data* that corresponds to the tx covariate

regime	A formula object or a list of formula objects. The covariates to be included in the decision function/kernel. If a list is provided, this specifies that there is an underlying subset structure – fSet must then be defined. For subsets, the name of each element of the list must correspond to the name of a subset. If a regime is to be estimated using multiple subsets combined, each subset must be included in the name and separated by a comma (no spaces).
response	A numeric vector. The same as reward above. Allows for naming convention followed in most DynTxRegime methods.
BOWLObj	NULL or BOWL-class object returned from previous call to bowl(). If NULL, indicates that the function call is for the first STEP of the BOWL algorithm (i.e., the final decision point). If a BOWL-class object, assumed that the object was returned by the preceding step of the BOWL algorithm.
lambdas	A numeric object or a numeric vector object giving the penalty tuning parameter(s). If more than 1 is provided, the set of tuning parameter values to be considered in the cross-validation algorithm (note that cvFolds must be positive in this case).
cvFolds	If cross-validation is to be used to select the tuning parameters and/or kernel parameters, the number of folds.
kernel	A character object. Must be one of {'linear', 'poly', 'radial'}
kparam	A numeric object. If kernel = linear, kparam is ignored. If kernel = poly, kparam is the degree of the polynomial. If kernel = radial, kparam is the inverse bandwidth of the kernel. If a vector of bandwidth parameters is given, cross-validation will be used to select the parameter (note that cvFolds must be positive in this case).
fSet	A function or NULL defining subset structure. See ?fSet for details.
surrogate	The surrogate 0-1 loss function. Must be one of {'logit', 'exp', 'hinge', 'sqhinge', 'huber'}.
verbose	An integer or logical. If 0, no screen prints are generated. If 1, screen prints are generated with the exception of optimization results obtained in iterative algorithm. If 2, all screen prints are generated.

Value

a BOWL-class object

References

Yingqi Zhao, Donglin Zeng, Eric B. Laber, Michael R. Kosorok (2015) New statistical learning methods for estimating optimal dynamic treatment regimes. Journal of the American Statistical Association, 110:510, 583–598.

See Also

Other statistical methods: earl(), iqLearn, optimalClass(), optimalSeq(), owl(), qLearn(), rwl()

Other weighted learning methods: earl(), owl(), rwl()

Other multiple decision point methods: iqLearn, optimalClass(), optimalSeq(), qLearn()

Examples

```
# Load and process data set
data(bmiData)

# define the negative 12 month change in BMI from baseline
y12 <- -100*(bmiData[,6L] - bmiData[,4L])/bmiData[,4L]

# define the negative 4 month change in BMI from baseline
y4 <- -100*(bmiData[,5L] - bmiData[,4L])/bmiData[,4L]

# reward for second stage
rewardSS <- y12 - y4

#### Second-stage regression

# Constant propensity model
moPropen <- buildModelObj(model = ~1,
                          solver.method = 'glm',
                          solver.args = list('family'='binomial'),
                          predict.method = 'predict.glm',
                          predict.args = list(type='response'))

fitSS <- bowl(moPropen = moPropen,
              data = bmiData, reward = rewardSS,  txName = 'A2',
              regime = ~ parentBMI + month4BMI)

##Available methods

  # Coefficients of the propensity score regression
  coef(fitSS)

  # Description of method used to obtain object
  DTRstep(fitSS)

  # Estimated value of the optimal treatment regime for training set
  estimator(fitSS)

  # Value object returned by propensity score regression method
  fitObject(fitSS)

  # Summary of optimization routine
  optimObj(fitSS)

  # Estimated optimal treatment for training data
  optTx(fitSS)

  # Estimated optimal treatment for new data
```

```
  optTx(fitSS, bmiData)

  # Plots if defined by propensity regression method
  dev.new()
  par(mfrow = c(2,4))

  plot(fitSS)
  plot(fitSS, suppress = TRUE)

  # Value object returned by propensity score regression method
  propen(fitSS)

  # Parameter estimates for decision function
  regimeCoef(fitSS)

  # Show main results of method
  show(fitSS)

  # Show summary results of method
  summary(fitSS)

#### First-stage regression

# Constant propensity model
fitFS <- bowl(moPropen = moPropen,
              data = bmiData, reward = y4,  txName = 'A1',
              regime = ~ gender + parentBMI,
              BOWLObj = fitSS, lambdas = c(0.5, 1.0), cvFolds = 4L)

##Available methods for fitFS are as shown above for fitSS

  # Results of the cross-validation
  cvInfo(fitFS)
```

`buildModelObjSubset` *Create Model Objects for Subsets of Data*

Description

Extends the buildModelObj() function of package **modelObj**. Here, the returned model object includes a specification of the decision point and subset of the data to which the model is to be applied.

Usage

```
buildModelObjSubset(
  ...,
  model,
  solver.method,
```

```
  solver.args = NULL,
  predict.method = NULL,
  predict.args = NULL,
  dp = 1L,
  subset = NA
)
```

Arguments

`...`	ignored. Included to require named input.
`model`	An object of class `formula`. The symbolic description of the model to be fitted. If the regression method specified in solver.method accepts as input a `formula` object, model is passed to the solver.method function. If the regression method instead accepts a matrix of covariates as the model to fit, `model` is used to obtain the model matrix that is passed to the solver.method function.
`solver.method`	An object of class `character`. The name of the R function to be used to obtain parameter estimates, e.g., 'lm', 'glm', or 'rpart'. The specified function MUST have a corresponding predict method, which can be the generic predict() function.

`solver.args`	An object of class `list`. Additional arguments to be sent to the function specified in solver.method. This argument must be provided as a named list, where the name of each element matches a formal argument of the function specified in solver.method. For example, if a logistic regression using 'glm' is desired,
	`solver.method = "glm"`
	`solver.args = list("family"=binomial)`
	See Details section for further information.

`predict.method`	An object of class `character`. The name of the R function to be used to obtain predictions, e.g., 'predict.lm', 'predict', or 'predict.glm'. If no function is explicitly given, the generic `predict()` is assumed. For many regression methods, the generic `predict()` method is appropriate.

`predict.args`	An object of class `list`. Additional arguments to be sent to the function specified in predict.method. This argument must be provided as a named list, where the name of each element matches a formal argument of the function specified in predict.method. For example, if a logistic regression using 'glm' was used to fit the model and predictions on the scale of the response are desired,
	`predict.method = "predict.glm"`
	`predict.args = list("type"="response").`
	See Details section for further information.

`dp`	An object of class `integer`. The decision point for which this model and subset are defined.
`subset`	An object of class `character`. A nickname for the subset for which model and methods are to be used. This argument will be used by the methods of **DynTxRegime** to "link" input arguments. In the event that a model is to be fit using more than 1 subset, collapse the subset names into a single character string separating each with a comma. For example, if the model is to be fit using

patients in both subsets "a" and "b," the subset nickname should be "a,b" (no space).

Details

In some settings, an analyst may want to use different models for unique subsets of the data. `buildModelObjSubset()` provides a mechanism for users to define models for such subset. Specifically, models are specified in connection with the decision point and subset to which they are to be applied.

See ?modelObj for further details

Value

An object of class `ModelObjSubset`, which contains a complete description of the conditions under which a model is to be used and the R methods to be used to obtain parameter estimates and predictions.

Examples

```
# Consider a 2 decision point trial. At the 1st decision point, the subset of
# treatment options available to each patient is always set "set1."
# At the 2nd decision point, some patients are eligible to receive
# treatment from set "set2a" and others from set "set2b." The outcome
# for these subsets will be modeled as ~ x1 + x2 and ~ x2 + x3, respectively.
#
# All parameter estimates are to be obtained used lm and predictions obtained using predict.
#
# The following illustrates how to build these model objects.

  model <- list()

  model[[1]] <- buildModelObjSubset(dp = 1, subset = "set1",
                                    model = ~ x1 + x2 + x3, solver.method = 'lm')

  model[[2]] <- buildModelObjSubset(dp = 2, subset = "set2a",
                                    model = ~ ~ x1 + x2, solver.method = 'lm')

  model[[3]] <- buildModelObjSubset(dp = 2, subset = "set2b",
                                    model = ~ x2 + x3, solver.method = 'lm')
```

Call *Retrieve Unevaluated Original Call*

Description

Returns the unevaluated original call to a DynTxRegime statistical method.

Usage

```
Call(name, ...)
```

Arguments

name	Object for which call is desired
...	Optional additional input required by R's base call().

Details

Methods are defined for all statistical methods implemented in DynTxRegime.

---

classif *Retrieve Classification Regression Analysis*

Description

Method retrieves the value object returned by the user specified classification regression modeling object(s). Exact structure of the returned object will vary.

Usage

```
classif(object, ...)

## S4 method for signature 'OptimalClass'
classif(object, ...)
```

Arguments

object	Value object returned from a method that uses classification regression
...	Ignored.

---

coef *Extract Model Coefficients From Objects Returned by Modeling Functions*

Description

A list is returned, one element for each regression step required by the statistical method.

Usage

```
coef(object, ...)
```

Arguments

`object`	Value object returned by any statistical method implemented in DynTxRegime.
`...`	Optional additional inputs defined by coefficient methods of selected regression functions.

Details

Methods are defined for all statistical methods implemented in DynTxRegime.

The exact structure of the returned list will vary depending on the statistical method. For methods that include a propensity regression, the returned list will include an element named 'propen'. For methods that include an outcome regression, the returned list will include an element named 'outcome'.

`cvInfo` *Extract Cross-Validation Results*

Description

Extract cross-validation results from the value object returned by a weighted learning statistical method of DynTxRegime.

Usage

```
cvInfo(object, ...)
```

Arguments

`object`	A value object returned by a weighted learning statistical method of DynTxRegime
`...`	Ignored.

Details

Methods are developed for all weighted learning methods implemented in DynTxRegime. Specifically, OWL, RWL, BOWL, and EARL.

DTRstep *Identify Statistical Method Used to Obtain Result*

Description

Prints are displays a brief description of the statistical method used to obtain the input object.

Usage

```
DTRstep(object)
```

Arguments

object	Value object returned by any statistical method of DynTxRegime

Details

Methods are defined for all statistical methods implemented in DynTxRegime.

earl *Efficient Augmentation and Relaxation Learning*

Description

Efficient Augmentation and Relaxation Learning

Usage

```
earl(
  ...,
  moPropen,
  moMain,
  moCont,
  data,
  response,
  txName,
  regime,
  iter = 0L,
  fSet = NULL,
  lambdas = 0.5,
  cvFolds = 0L,
  surrogate = "hinge",
  kernel = "linear",
  kparam = NULL,
  verbose = 2L
)
```

Arguments

...	Used primarily to require named input. However, inputs for the optimization methods can be sent through the ellipsis. If surrogate is hinge, the optimization method is dfoptim::hjk(). For all other surrogates, stats::optim() is used.
moPropen	An object of class modelObj or modelObjSubset, which defines the model and R methods to be used to obtain parameter estimates and predictions for the propensity for treatment. See ?moPropen for details.
moMain	An object of class modelObj or modelObjSubset, which defines the model and R methods to be used to obtain parameter estimates and predictions for the main effects of the outcome. See ?modelObj for details.
moCont	An object of class modelObj or modelObjSubset, which defines the model and R methods to be used to obtain parameter estimates and predictions for the contrasts of the outcome. See ?modelObj for details.
data	A data frame of the covariates and tx histories
response	The response variable.
txName	A character object. The column header of *data* that corresponds to the tx covariate
regime	A formula object or a list of formula objects. The covariates to be included in classification. If a list is provided, this specifies that there is an underlying subset structure – fSet must then be defined.
iter	Maximum number of iterations for outcome regression
fSet	A function or NULL defining subset structure
lambdas	A numeric object or a numeric vector object giving the penalty tuning parameter. If more than 1 is provided, the finite set of values to be considered in the cross-validation algorithm
cvFolds	If cross-validation is to be used to select the tuning parameters, the number of folds.
surrogate	The surrogate 0-1 loss function must be one of logit, exp, hinge, sqhinge, huber
kernel	A character object. must be one of {"linear", "poly", "radial"}
kparam	A numeric object of NULL. If kernel = linear, kparam is ignored. If kernel = poly, kparam is the degree of the polynomial If kernel = radial, kparam is the inverse bandwidth of the kernel. If a vector of bandwidth parameters is given, cross-validation will be used to select the parameter
verbose	An integer or logical. If 0, no screen prints are generated. If 1, screen prints are generated with the exception of optimization results obtained in iterative algorithm. If 2, all screen prints are generated.

Value

an EARL object

References

Ying-Qi Zhao, Eric Laber, Sumona Saha and Bruce E. Sands (2016+) Efficient augmentation and relaxation learning for treatment regimes using observational data

See Also

Other statistical methods: bowl(), iqLearn, optimalClass(), optimalSeq(), owl(), qLearn(), rwl()

Other single decision point methods: optimalClass(), optimalSeq(), owl(), qLearn(), rwl()

Other weighted learning methods: bowl(), owl(), rwl()

Examples

```
# Load and process data set
data(bmiData)

# define the negative 12 month change in BMI from baseline
y12 <- -100*(bmiData[,6L] - bmiData[,4L])/bmiData[,4L]

# propensity model
moPropen <- buildModelObj(model = ~parentBMI+month4BMI,
                          solver.method = 'glm',
                          solver.args = list('family'='binomial'),
                          predict.method = 'predict.glm',
                          predict.args = list(type='response'))

# outcome model
moMain <- buildModelObj(model = ~parentBMI+month4BMI,
                        solver.method = 'lm')

moCont <- buildModelObj(model = ~parentBMI+month4BMI,
                        solver.method = 'lm')

fitEARL <- earl(moPropen = moPropen, moMain = moMain, moCont = moCont,
              data = bmiData, response = y12,  txName = 'A2',
              regime = ~ parentBMI + month4BMI,
              surrogate = 'logit', kernel = 'poly', kparam = 2)

##Available methods

  # Coefficients of the regression objects
  coef(fitEARL)

  # Description of method used to obtain object
  DTRstep(fitEARL)

  # Estimated value of the optimal treatment regime for training set
  estimator(fitEARL)

  # Value object returned by regression methods
  fitObject(fitEARL)

  # Summary of optimization routine
  optimObj(fitEARL)
```

```
# Estimated optimal treatment for training data
optTx(fitEARL)

# Estimated optimal treatment for new data
optTx(fitEARL, bmiData)

# Value object returned by outcome regression method
outcome(fitEARL)

# Plots if defined by regression methods
dev.new()
par(mfrow = c(2,4))

plot(fitEARL)
plot(fitEARL, suppress = TRUE)

# Value object returned by propensity score regression method
propen(fitEARL)

# Parameter estimates for decision function
regimeCoef(fitEARL)

# Show main results of method
show(fitEARL)

# Show summary results of method
summary(fitEARL)
```

EARL-class *Class* EARL

Description

Class EARL contains results for an EARL analysis.

Slots

`analysis` Contains a Learning or LearningMulti object.

`analysis@txInfo` Feasible tx information.

`analysis@propen` Propensity regression analysis.

`analysis@outcome` Outcome regression analysis.

`analysis@cvInfo` Cross-validation analysis if single regime.

`analysis@optim` Optimization analysis if single regime.

`analysis@optimResult` list of cross-validation and optimization results if multiple regimes. optimResult[[i]]@cvInfo and optimResult[[i]]@optim.

`analysis@optimal` Estimated optimal Tx and value.

`analysis@call` Unevaluated call to statistical method.

Methods For Post-Processing of Regression Analysis

outcome : Retrieve value object returned by outcome regression methods.

propen : Retrieve value object returned by propensity regression methods.

coef : Retrieve parameter estimates for all regression steps.

fitObject : Retrieve value object returned by regression methods.

plot : Generate plots for regression analyses.

Methods For Post-Processing of Optimization Analysis

cvInfo : Retrieve cross-validation results.

optimObj : Retrieve value object returned by optimization method(s).

regimeCoef : Retrieve estimated parameters for optimal tx regime.

Methods For Accessing Main Results

DTRstep : Retrieve description of method used to create object.

estimator : Retrieve the estimated value of the estimated optimal regime for the training data set.

optTx : Retrieve/predict the estimated decision functions and/or optimal tx.

print : Print main results of analysis.

show : Show main results of analysis.

summary : Retrieve summary information.

---

estimator *Retrieve the Estimated Value*

Description

Retrieve the value as estimated by the statistical method.

Usage

```
estimator(x, ...)

## S4 method for signature 'IQLearnFS'
estimator(x, w = NULL, y = NULL, z = NULL, dens = NULL)

## S4 method for signature 'IQLearnSS'
estimator(x, w = NULL, y = NULL, z = NULL, dens = NULL)
```

Arguments

x	a DynTxRegime Object.
...	Optional additional input. Ignored.
w	If IQ-Learning, object of class IQLearnSS, IQLearnFS_C, IQLearnFS_ME, or IQLearnFS_VHet
y	If IQ-Learning, object of class IQLearnSS, IQLearnFS_C, IQLearnFS_ME, or IQLearnFS_VHet
z	If IQ-Learning, object of class IQLearnSS, IQLearnFS_C, IQLearnFS_ME, or IQLearnFS_VHet
dens	If IQ-Learning, one of {"norm", "nonpar"}

---

fitObject *Objects Returned by Modeling Functions*

Description

Returns a list of the objects returned by all modeling functions

Usage

```
fitObject(object, ...)
```

Arguments

object	Value object returned by a statistical method of DynTxRegime
...	Optional additional inputs

Details

Methods are defined for all statistical methods implemented in DynTxRegime.

The exact structure of the returned list will vary depending on the statistical method. For methods that include a propensity regression, the returned list will include an element named 'propen'. For methods that include an outcome regression, the returned list will include an element named 'outcome'.

fittedCont — *Retrieve the Fitted Contrast Component from Second Stage IQ-Learning*

Description

Extracts the contrasts component of the fitted outcome regression the second-stage analysis of the interactive Q-Learning algorithm.

Usage

```
fittedCont(object, ...)

## S4 method for signature 'IQLearnSS'
fittedCont(object, ...)
```

Arguments

object	An object of class IQLearnSS
...	Ignored.

fittedMain — *Retrieve the Fitted Main Effects Component from Second Stage IQ-Learning*

Description

Extracts the main effects component of the fitted outcome regression for the second-stage analysis of the interactive Q-Learning algorithm.

Usage

```
fittedMain(object, ...)

## S4 method for signature 'IQLearnSS'
fittedMain(object, ...)
```

Arguments

object	An object of class IQLearnSS
...	Ignored.

`fSet` *Defining the fSet Input Variable*

Description

Several of the statistical methods implemented in package **DynTxRegime** allow for subset modeling or limiting of feasible treatment options. This section details how this input is to be defined.

Details

In general, input `fSet` is used to define subsets of patients within an analysis. These subsets can be specified to (1) limit available treatments, (2) use different models for the propensity score and/or outcome regressions, and/or (3) use different decision function models for each subset of patients. The combination of inputs `moPropen`, `moMain`, `moCont`, `fSet`, and/or `regimes` determines which of these scenarios is being considered. We cover some common situations below.

Regardless of the purpose for specifying `fSet`, it must be a function that returns a list. There are two options for defining the function. Version 1 is that of the original **DynTxRegime** package. In this version, `fSet` defines the rules for determining the subset of treatment options for an INDIVIDUAL. The first element of the returned list is a character, which we term the subset 'nickname.' This nickname is for bookkeeping purposes and is used to link models to subsets. The second element of the returned list is a vector of available treatment options for the subset. The formal arguments of the function must include (i) 'data' or (ii) individual covariate names as given by the column headers of `data`. An example using the covariate name input form is

```
fSet <- function(a1) {
  if (a1 > 1) {
    subset <- list('subA',c(1,2))
  } else {
    subset <- list('subB',c(3,4) )
  }
  return(subset)
}
```

This function indicates that if an individual has covariate a1 > 1, they are a member of subset 'subA' and their feasible treatment options are {1,2}. If a1 <= 1, they are a member of subset 'subB' and their feasible treatment options are {3,4}.

A more efficient implementation for `fSet` is now accepted. In the second form, `fSet` defines the subset of treatment options for the full DATASET. It is again a function with formal arguments (i) 'data' or (ii) individual covariate names as given by the column headers of `data`. The function returns a list containing two elements: 'subsets' and 'txOpts.' Element 'subsets' is a list comprising all treatment subsets; each element of the list contains the nickname and treatment options for a single subset. Element 'txOpts' is a character vector indicating the subset of which each individual is a member. In this new format, the equivalent definition of `fSet` as that given above is:

```
fSet <- function(a1) {
  subsets <- list(list('subA', c(1,2)),
```

```
                  list('subB', c(3,4)))
  txOpts <- rep('subB', length(x = a1))
  txOpts[a1 > 1] <- 'subA'

  return(list("subsets" = subsets,
              "txOpts" = txOpts))
}
```

Though a bit more complicated, this version is much more efficient as it processes the entire dataset at once rather than each individual separately.

The simplest scenario involving `fSet` is to define feasible treatment options and the rules that dictate how those treatment options are determined. For example, responder/non-responder scenarios are often encountered in multiple-decision-point settings. An example of this scenario is: patients that respond to the first stage treatment remain on the original treatment; those that do not respond to the first stage treatment have all treatment options available to them at the second stage. In this case, the propensity score models for the second stage are fit using only 'non-responders' for whom more than 1 treatment option is available.

An example of an appropriate `fSet` function for the second-stage is

```
fSet <- function(data) {
   if (data\$responder  == 0L) {
     subset <- list('subA',c(1L,2L))
   } else if (data\$tx1 == 1L) {
     subset <- list('subB',c(1L) )
   } else if (data\$tx1 == 2L) {
     subset <- list('subC',c(2L) )
   }
   return(subset)
}
```

for version 1 or for version 2

```
fSet <- function(data) {
  subsets <- list(list('subA', c(1L,2L)),
                  list('subB', c(1L)),
                  list('subC', c(2L)))
  txOpts <- character(nrow(x = data))
  txOpts[data$tx1 == 1L] <- 'subB'
  txOpts[data$tx1 == 2L] <- 'subC'
  txOpts[data$responder == 0L] <- 'subA'

  return(list("subsets" = subsets,
              "txOpts" = txOpts))
}
```

The functions above specify that patients with covariate responder = 0 receive treatments from subset 'subA,' which comprises treatments A = (1,2). Patients with covariate responder = 1 receive treatment from subset 'subB' or 'subC' depending on the first stage treatment received. If `fSet` is

specified in this way, the form of the model object depends on the training data. Specifically, if the training data obeys the feasible treatment rule (here, all individuals with responder = 1 received tx in accordance with fSet), moPropen would be a "modelObj"; the propensity model will be fit using only those patients with responder = 0; those with responder = 1 always receive the appropriate second stage treatment with probability 1.0. However, if the data are from an observation study and the training data do not obey the feasible treatment rules (here, some individuals with responder = 1 received tx = 0; others tx = 1), the responder = 1 data must be modeled and moPropen must be provided as one or more ModelObjSubset() objects.

If outcome regression is used by the method, moMain and moCont can be either objects of class "modelObj" if only responder = 0 patients are to be used to obtain parameter estimates or as lists of objects of class "ModelObjSubset" if subsets are to be analyzed individually or combined for a single fit of all data.

For a scenario where all patients have the same set of treatment options available, but subsets of patients are to be analyzed using different models. We cane define fSet as

```
fSet <- function(data) {
   if (data\$a1 == 1) {
     subset <- list('subA',c(1L,2L))
   } else {
     subset <- list('subB',c(1L,2L) )
   }
   return(subset)
}
```

for version 1 or in the format of version 2

```
fSet <- function(data)
{
  subsets <- list(list('subA', c(1L,2L)),
                  list('subB', c(1L,2L)))
  txOpts <- rep('subB', nrow(x = data))
  txOpts[data$a1 == 1L] <- 'subA'

  return(list("subsets" = subsets,
              "txOpts" = txOpts))
}
```

where all patients have the same treatment options available, A = (1,2), but different regression models will be fit for each subset (case 2 above) and/or different decision function models (case 3 above) for each subset. If different propensity score models are used, moPropen must be a list of objects of class "modelObjSubset." Perhaps,

```
  propenA <- buildModelObjSubset(model = ~1,
                                 solver.method = 'glm',
                                 solver.args = list('family'='binomial'),
                                 predict.method = 'predict.glm',
                                 predict.args = list(type='response'),
                                 subset = 'subA')
```

```
propenB <- buildModelObjSubset(model = ~1,
                               solver.method = 'glm',
                               solver.args = list('family'='binomial'),
                               predict.method = 'predict.glm',
                               predict.args = list(type='response'),
                               subset = 'subB')

moPropen <- list(propenA, propenB)
```

If different decision function models are to be fit, `regimes` would take a form similar to

```
regimes <- list( 'subA' = ~x1 + x2,
                 'subB' = ~x2 )
```

Notice that the names of the elements of `regimes` and the subsets passed to buildModelObjSubset() correspond to the names defined by `fSet`, i.e., 'subA' or 'subB.' These nicknames are used for bookkeeping and link subsets to the appropriate models.

For a single-decision-point analysis, `fSet` is a single function. For multiple-decision-point analyses, `fSet` is a list of functions where each element of the list corresponds to the decision point (1st element <- 1st decision point, etc.)

`genetic` *Retrieve the Genetic Algorithm Results*

Description

Retrieve the value object returned by rgenoud() in optimalSeq().

Usage

```
genetic(object, ...)

## S4 method for signature 'OptimalSeq'
genetic(object, ...)
```

Arguments

`object`	Value object returned by optimalSeq()
`...`	Optional inputs. Ignored.

iqLearn *Interactive Q-Learning*

Description

The complete interactive Q-Learning algorithm.

Usage

```
## Second-Stage Analysis
iqLearnSS(..., moMain, moCont, data, response, txName, iter = 0L,
          verbose = TRUE)

## First-Stage Analysis for Fitted Main Effects
iqLearnFSM(..., moMain, moCont, data, response, txName, iter = 0L,
           verbose = TRUE)

## First-Stage Analysis for Fitted Contrasts
iqLearnFSC(..., moMain, moCont, data, response, txName, iter = 0L,
           verbose = TRUE)

## First-Stage Analysis of Contrast Variance Log-Linear Model
iqLearnFSV(..., object, moMain, moCont, data, iter = 0L, verbose = TRUE)
```

Arguments

...	ignored. Provided to require named inputs.
moMain	An object of class modelObj or a list of objects of class modelObjSubset, which define the models and R methods to be used to obtain parameter estimates and predictions for the main effects component of the outcome regression. See ?modelObj and/or ?modelObjSubset for details. NULL is an acceptable value if moCont is defined.
moCont	An object of class modelObj or a list of objects of class modelObjSubset, which define the models and R methods to be used to obtain parameter estimates and predictions for the contrasts component of the outcome regression. See ?modelObj and/or ?modelObjSubset for details. NULL is an acceptable value if moMain is defined.
data	A data frame of covariates and treatment history.
response	For the second stage analysis, the response vector. For first stage analyses, the value object returned by iqLearnSS().
object	The value object returned by iqLearFSC()
txName	A character string giving column header of treatment variable in data
iter	An integer. See ?iter for details
verbose	A logical. If TRUE, screen prints are generated.

References

Laber, EB, Linn, KA, and Stefanski, LA (2014). Interactive model building for Q-Learning. Biometrika, 101, 831–847. PMCID: PMC4274394.

See Also

Other statistical methods: bowl(), earl(), optimalClass(), optimalSeq(), owl(), qLearn(), rwl()

Other multiple decision point methods: bowl(), optimalClass(), optimalSeq(), qLearn()

Examples

```
# Load and process data set
data(bmiData)

# define the negative 12 month change in BMI from baseline
y12 <- -100*(bmiData[,6L] - bmiData[,4L])/bmiData[,4L]

#### Full Interactive Q-Learning Algorithm

### Second-Stage Analysis

# outcome model
moMain <- buildModelObj(model = ~parentBMI+month4BMI,
                        solver.method = 'lm')

moCont <- buildModelObj(model = ~race + parentBMI+month4BMI,
                        solver.method = 'lm')

fitSS <- iqLearnSS(moMain = moMain, moCont = moCont,
                   data = bmiData, response = y12,  txName = 'A2')

### First-Stage Analysis Main Effects Term

# main effects model
moMain <- buildModelObj(model = ~parentBMI+baselineBMI,
                        solver.method = 'lm')

moCont <- buildModelObj(model = ~race + parentBMI+baselineBMI,
                        solver.method = 'lm')

fitFSM <- iqLearnFSM(moMain = moMain, moCont = moCont,
                     data = bmiData, response = fitSS,  txName = 'A1')

### First-Stage Analysis Contrasts Term

# contrasts model
moMain <- buildModelObj(model = ~parentBMI+baselineBMI,
                        solver.method = 'lm')
```

```
moCont <- buildModelObj(model = ~race + parentBMI+baselineBMI,
                        solver.method = 'lm')

fitFSC <- iqLearnFSC(moMain = moMain, moCont = moCont,
                     data = bmiData, response = fitSS,  txName = 'A1')

### First-Stage Analysis Contrasts Variance - Log-linear

# contrasts variance model
moMain <- buildModelObj(model = ~baselineBMI,
                        solver.method = 'lm')

moCont <- buildModelObj(model = ~baselineBMI,
                        solver.method = 'lm')

fitFSV <- iqLearnFSV(object = fitFSC, moMain = moMain, moCont = moCont,
                     data = bmiData)

####Available methods

  ### Estimated value
  estimator(x = fitFSC, y = fitFSM, z = fitFSV, w = fitSS, dens = 'nonpar')

  ## Estimated optimal treatment and decision functions for training data
  ## Second stage optimal treatments
  optTx(x = fitSS)

  ## First stage optimal treatments when contrast variance is modeled.
  optTx(x = fitFSM, y = fitFSC, z = fitFSV, dens = 'nonpar')

  ## First stage optimal treatments when contrast variance is constant.
  optTx(x = fitFSM, y = fitFSC, dens = 'nonpar')

  ## Estimated optimal treatment and decision functions for new data
  ## Second stage optimal treatments
  optTx(x = fitSS, bmiData)

  ## First stage optimal treatments when contrast variance is modeled.
  optTx(x = fitFSM, y = fitFSC, z = fitFSV, dens = 'nonpar', bmiData)

  ## First stage optimal treatments when contrast variance is constant.
  optTx(x = fitFSM, y = fitFSC, dens = 'nonpar', bmiData)

### The following methods are available for all objects: fitSS, fitFSM,
### fitFSC and fitFSV. We include only one here for illustration.

  # Coefficients of the outcome regression objects
  coef(object = fitSS)

  # Description of method used to obtain object
  DTRstep(object = fitFSM)

  # Value object returned by outcome regression method
```

```
fitObject(object = fitFSC)

# Value object returned by outcome regression method
outcome(object = fitFSV)

# Plots if defined by outcome regression method
dev.new()
par(mfrow = c(2,4))

plot(x = fitSS)
plot(x = fitSS, suppress = TRUE)

# Show main results of method
show(object = fitFSM)

# Show summary results of method
summary(object = fitFSV)
```

IQLearnFS_C-class *Class* IQLearnFS_C

Description

Class IQLearnFS_C contains the results for the first stage contrasts component of the interactive Q-Learning algorithm. Objects of this class are returned by iqLearnFSC().

Slots

txVec : A numeric. treatment vector from training data

residuals : A numeric. residuals of the fit

step : Not used in this context.

outcome : The outcome regression analysis

txInfo : The feasible tx information

optimal : The estimated optimal tx, decision function, and value

Methods For Post-Processing of Regression Analysis

outcome : Retrieve value object returned by outcome regression methods.

coef : Retrieve parameter estimates for all regression steps.

fitObject : Retrieve value object returned by regression methods.

plot : Generate plots for regression analyses.

Methods For Accessing Main Results

DTRstep : Retrieve description of method used to create object.

estimator : Retrieve the estimated value of the estimated optimal regime for the training data set.

optTx : Retrieve/predict the estimated decision functions and/or optimal tx.

print : Print main results of analysis.

show : Show main results of analysis.

summary : Retrieve summary information.

residuals :Retrieve the residuals of the regression.

sd :Retrieve the standard deviation of the residuals.

IQLearnFS_ME-class *Class* IQLearnFS_ME

Description

Class IQLearnFS_ME contains the results for the first stage main effects component of the interactive Q-Learning algorithm. Objects of this class are returned by iqLearnFSM().

Slots

step : Not used in this context.

outcome : The outcome regression analysis

txInfo : The feasible tx information

optimal : The estimated optimal tx, decision function, and value

Methods For Post-Processing of Regression Analysis

outcome : Retrieve value object returned by outcome regression methods.

coef : Retrieve parameter estimates for all regression steps.

fitObject : Retrieve value object returned by regression methods.

plot : Generate plots for regression analyses.

Methods For Accessing Main Results

DTRstep : Retrieve description of method used to create object.

estimator : Retrieve the estimated value of the estimated optimal regime for the training data set.

optTx : Retrieve/predict the estimated decision functions and/or optimal tx.

print : Print main results of analysis.

show : Show main results of analysis.

summary : Retrieve summary information.

IQLearnFS_VHet-class *Class* IQLearnFS_VHet

Description

Class IQLearnFS_VHet contains the results for the first stage residuals component of the interactive Q-Learning algorithm. Objects of this class are returned by iqLearnFSV().

Slots

residuals : Standardized residuals of contrast after modeling

scale : Scaling factor for stdization

step : Not used in this context.

outcome : The outcome regression analysis

txInfo : The feasible tx information

optimal : The estimated optimal tx, decision function, and value

Methods For Post-Processing of Regression Analysis

outcome : Retrieve value object returned by outcome regression methods.

coef : Retrieve parameter estimates for all regression steps.

fitObject : Retrieve value object returned by regression methods.

plot : Generate plots for regression analyses.

Methods For Accessing Main Results

DTRstep : Retrieve description of method used to create object.

estimator : Retrieve the estimated value of the estimated optimal regime for the training data set.

optTx : Retrieve/predict the estimated decision functions and/or optimal tx.

print : Print main results of analysis.

show : Show main results of analysis.

summary : Retrieve summary information.

residuals :Retrieve the residuals of the regression.

qqplot QQ plot of the residuals for the interactive Q-Learning algorithm.

IQLearnSS-class *Class* IQLearnSS

Description

Class IQLearnSS contains the results for the second stage of the interactive Q-Learning algorithm. Objects of this class are returned by iqLearnSS().

Slots

yContHat : A numeric. Estimated contrast component

yMainHat : A numeric. Estimated main effects component

delta : A numeric. Indicator of compliance * response used for value calc

step : Not used in this context.

outcome : The outcome regression analysis

txInfo : The feasible tx information

optimal : The estimated optimal tx, decision function, and value

Methods For Post-Processing of Regression Analysis

outcome : Retrieve value object returned by outcome regression methods.

coef : Retrieve parameter estimates for all regression steps.

fitObject : Retrieve value object returned by regression methods.

plot : Generate plots for regression analyses.

fittCont :Retrieve the contrasts component of the regression.

fittMain :Retrieve the main effects component of the regression.

Methods For Accessing Main Results

DTRstep : Retrieve description of method used to create object.

estimator : Retrieve the estimated value of the estimated optimal regime for the training data set.

optTx : Retrieve/predict the estimated decision functions and/or optimal tx.

print : Print main results of analysis.

show : Show main results of analysis.

summary : Retrieve summary information.

iter — *Defining the iter Input Variable*

Description

Several of the statistical methods implemented in package **DynTxRegime** allow for an iterative algorithm when completing an outcome regression. This section details how this input is to be defined.

Details

Outcome regression models are specified by the main effects components (`moMain`) and the contrasts component (`moCont`). Assuming that the treatment is denoted as binary A, the full regression model is: moMain + A*moCont. There are two ways to fit this model: (i) in the full model formulation (moMain + A*moCont) or (ii) each component, `moMain` and `moCont`, is fit separately. `iter` specifies if (i) or (ii) should be used.

`iter` >= 1 indicates that `moMain` and `moCont` are to be fit separately using an iterative algorithm. `iter` is the maximum number of iterations. Assume Y = Ymain + Ycont; the iterative algorithm is as follows:

(1) hat(Ycont) = 0;

(2) Ymain = Y - hat(Ycont);

(3) fit Ymain ~ moMain;

(4) set Ycont = Y - hat(Ymain)

(5) fit Ycont ~ A*moCont;

(6) Repeat steps (2) - (5) until convergence or a maximum of iter iterations.

This choice allows the user to specify, for example, a linear main effects component and a non-linear contrasts component.

`iter` <= 0 indicates that the full model formulation is to be used. The components `moMain` and `moCont` will be combined in the package and fit as a single object. Note that if `iter` <= 0, all non-model components of `moMain` and `moCont` must be identical. Specifically, the regression method and any non-default arguments should be identical. By default, the specifications in `moMain` are used.

moPropen — *Defining the moPropen Input Variable*

Description

Several of the statistical methods implemented in package **DynTxRegime** use propensity score modeling. This section details how this input is to be defined.

Details

For input `moPropen`, the method specified to obtain predictions MUST return the prediction on the scale of the probability, i.e., predictions must be in the range (0,1). In addition, `moPropen` differs from standard `"modelObj"` objects in that an additional element may be required in `predict.args`. Recall, `predict.args` is the list of control parameters passed to the prediction method. An additional control parameter, `propen.missing` can be included. `propen.missing` takes value "smallest" or "largest". It will be required if the prediction method returns predictions for only a subset of the treatment data; e.g., predict.glm(). `propen.missing` indicates if it is the smallest or the largest treatment value that is missing from the returned predictions.

For example, fitting a binary treatment (A in {0,1}) using

```
moPropen <- buildModelObj(model = ~1,
                          solver.method = 'glm',
                          solver.args = list('family'='binomial'),
                          predict.method = 'predict.glm',
                          predict.args = list(type='response'))
```

returns only P(A=1). P(A=0) is "missing," and thus

```
moPropen <- buildModelObj(model = ~1,
                          solver.method = 'glm',
                          solver.args = list('family'='binomial'),
                          predict.method = 'predict.glm',
                          predict.args = list(type='response',
                                              propen.missing = 'smallest'))
```

If the dimension of the value returned by the prediction method is less than the number of treatment options and no value is provided in `propen.missing`, it is assumed that the smallest valued treatment option is missing. Here, 'smallest' indicates the lowest value integer if treatment is an integer, or the 'base' level if treatment is a factor.

`optimalClass` *Classification Perspective*

Description

Classification Perspective

Usage

```
optimalClass(
  ...,
  moPropen,
  moMain,
  moCont,
  moClass,
```

```
  data,
  response,
  txName,
  iter = 0L,
  fSet = NULL,
  verbose = TRUE
)
```

Arguments

...	Included to require named inputs
moPropen	An object of class modelObj, which defines the models and R methods to be used to obtain parameter estimates and predictions for the propensity for treatment. See ?moPropen for details.
moMain	An object of class modelObj, which defines the models and R methods to be used to obtain parameter estimates and predictions for for the main effects component of the outcome regression. See ?modelObj for details. NULL is an appropriate value.
moCont	An object of class modelObj, which defines the models and R methods to be used to obtain parameter estimates and predictions for for the contrasts component of the outcome regression. See ?modelObj for details. NULL is an appropriate value.
moClass	An object of class modelObj, which defines the models and R methods to be used to obtain parameter estimates and predictions for the classification. See ?modelObj for details.
data	A data frame of the covariates and tx histories
response	The response vector
txName	An character giving the column header of the column in data that contains the tx covariate.
iter	An integer See ?iter for details
fSet	A function or NULL. This argument allows the user to specify the subset of tx options available to a patient. See ?fSet for details of allowed structure
verbose	A logical If FALSE, screen prints are suppressed.

Value

an object of class OptimalClass

References

Baqun Zhang, Anastasios A. Tsiatis, Marie Davidian, Min Zhang and Eric B. Laber. "Estimating optimal tx regimes from a classification perspective." Stat 2012; 1: 103-114.

Note that this method is a single decision point, binary treatment method. For multiple decision points, can be called repeatedly.

See Also

Other statistical methods: bowl(), earl(), iqLearn, optimalSeq(), owl(), qLearn(), rwl()

Other single decision point methods: earl(), optimalSeq(), owl(), qLearn(), rwl()

Other multiple decision point methods: bowl(), iqLearn, optimalSeq(), qLearn()

Examples

```
# Load and process data set
data(bmiData)

# define the negative 12 month change in BMI from baseline
y12 <- -100*(bmiData[,6L] - bmiData[,4L])/bmiData[,4L]

# Define the propensity for treatment model and methods.
moPropen <- buildModelObj(model =  ~ 1,
                          solver.method = 'glm',
                          solver.args = list('family'='binomial'),
                          predict.method = 'predict.glm',
                          predict.args = list(type='response'))

# classification model
library(rpart)
moClass <- buildModelObj(model = ~parentBMI+month4BMI+race+gender,
                         solver.method = 'rpart',
                         solver.args = list(method="class"),
                         predict.args = list(type='class'))

#### Second-Stage Analysis using IPW
fitSS_IPW <- optimalClass(moPropen = moPropen,
                          moClass = moClass,
                          data = bmiData, response = y12,  txName = 'A2')

# outcome model
moMain <- buildModelObj(model = ~parentBMI+month4BMI,
                        solver.method = 'lm')

moCont <- buildModelObj(model = ~race + parentBMI+month4BMI,
                        solver.method = 'lm')

#### Second-Stage Analysis using AIPW
fitSS_AIPW <- optimalClass(moPropen = moPropen,
                           moMain = moMain, moCont = moCont,
                           moClass = moClass,
                           data = bmiData, response = y12,  txName = 'A2')

##Available methods

  # Retrieve the classification regression object
  classif(object = fitSS_AIPW)
```

```
  # Coefficients of the outcome regression objects
  coef(object = fitSS_AIPW)

  # Description of method used to obtain object
  DTRstep(object = fitSS_AIPW)

  # Estimated value of the optimal treatment regime for training set
  estimator(x = fitSS_AIPW)

  # Value object returned by outcome regression method
  fitObject(object = fitSS_AIPW)

  # Estimated optimal treatment and decision functions for training data
  optTx(x = fitSS_AIPW)

  # Estimated optimal treatment and decision functions for new data
  optTx(x = fitSS_AIPW, newdata = bmiData)

  # Value object returned by outcome regression method
  outcome(object = fitSS_AIPW)
  outcome(object = fitSS_IPW)

  # Plots if defined by outcome regression method
  dev.new()
  par(mfrow = c(2,4))

  plot(x = fitSS_AIPW)
  plot(x = fitSS_AIPW, suppress = TRUE)

  # Retrieve the value object returned by propensity regression method
  propen(object = fitSS_AIPW)

  # Show main results of method
  show(object = fitSS_AIPW)

  # Show summary results of method
  summary(object = fitSS_AIPW)

#### First-stage Analysis using AIPW

 # Define the propensity for treatment model and methods.
 moPropen <- buildModelObj(model =  ~ 1,
                           solver.method = 'glm',
                           solver.args = list('family'='binomial'),
                           predict.method = 'predict.glm',
                           predict.args = list(type='response'))

# classification model
moClass <- buildModelObj(model = ~parentBMI+baselineBMI+race+gender,
                         solver.method = 'rpart',
                         solver.args = list(method="class"),
                         predict.args = list(type='class'))
```

```
# outcome model
moMain <- buildModelObj(model = ~parentBMI+baselineBMI,
                        solver.method = 'lm')

moCont <- buildModelObj(model = ~race + parentBMI+baselineBMI,
                        solver.method = 'lm')

fitFS_AIPW <- optimalClass(moPropen = moPropen,
                           moMain = moMain, moCont = moCont,
                           moClass = moClass,
                           data = bmiData, response = fitSS_AIPW,
                           txName = 'A1')

##Available methods for fitFS_AIPW are as shown above for fitSS_AIPW
```

`OptimalClass-class` *Class* `OptimalClass`

Description

Class `OptimalClass` contains results for a single decision point when estimates are obtained from the classification perspective. Objects of this class are returned by optimalClass().

Slots

`step` Step in the algorithm.

`analysis` Analysis results.

Methods For Post-Processing of Regression Analysis

outcome : Retrieve value object returned by outcome regression methods.

propen : Retrieve value object returned by propensity regression methods.

classif : Retrieve value object returned by classification regression methods.

coef : Retrieve parameter estimates for all regression steps.

fitObject : Retrieve value object returned by regression methods.

plot : Generate plots for regression analyses.

Methods For Accessing Main Results

DTRstep : Retrieve description of method used to create object.

estimator : Retrieve the estimated value of the estimated optimal regime for the training data set.

optTx : Retrieve/predict the estimated decision functions and/or optimal tx.

print : Print main results of analysis.

show : Show main results of analysis.

summary : Retrieve summary information.

`OptimalClassObj-class` *Class* `OptimalClassObj`

Description

Class `OptimalClassObj` contains results for a single decision point when estimates are obtained from the classification perspective. Objects of this class are returned by optimalClass().

Slots

`class` Results of the classification step.

`outcome` Results of the outcome regression step.

`propen` Results of the propensity step.

`optimal` Estimated optimal tx and value

`Call` Unevaluated call.

Methods For Post-Processing of Regression Analysis

outcome : Retrieve value object returned by outcome regression methods.

propen : Retrieve value object returned by propensity regression methods.

classif : Retrieve value object returned by classification regression methods.

coef : Retrieve parameter estimates for all regression steps.

fitObject : Retrieve value object returned by regression methods.

plot : Generate plots for regression analyses.

Methods For Accessing Main Results

DTRstep : Retrieve description of method used to create object.

estimator : Retrieve the estimated value of the estimated optimal regime for the training data set.

optTx : Retrieve/predict the estimated decision functions and/or optimal tx.

print : Print main results of analysis.

show : Show main results of analysis.

summary : Retrieve summary information.

OptimalInfo-class *Class* OptimalInfo

Description

Class OptimalInfo stores the estimated optimal tx, decision functions, and estimated value.

Slots

optimalTx a vector of the estimated optimal tx

estimatedValue a vector of the estimated value

decisionFunc a vector or matrix containing the values used to determine @optimalTx (if applicable)

optimalSeq *Missing or Coarsened Data Perspective - Genetic Algorithm*

Description

Missing or Coarsened Data Perspective - Genetic Algorithm

Usage

```
optimalSeq(
  ...,
  moPropen,
  moMain,
  moCont,
  data,
  response,
  txName,
  regimes,
  fSet = NULL,
  refit = FALSE,
  iter = 0L,
  verbose = TRUE
)
```

Arguments

...	Additional arguments required by rgenoud. At a minimum this should include Domains, pop.size and starting.values. See ?rgenoud for more information.

moPropen	An object of class modelObj, a list of objects of class modelObj, or a list of object of class modelObjSubset, which define the models and R methods to be used to obtain parameter estimates and predictions for the propensity for treatment. See ?moPropen for details.
moMain	An object of class modelObj, a list of objects of class modelObj, or a list of object of class modelObjSubset, which define the models and R methods to be used to obtain parameter estimates and predictions for the main effects component of the outcome regression. See ?modelObj and/or ?modelObjSubset for details. NULL is an acceptable input if IPWE is desired or there is no main effects component of the outcome regression model.
moCont	An object of class modelObj, a list of objects of class modelObj, or a list of object of class modelObjSubset, which define the models and R methods to be used to obtain parameter estimates and predictions for the contrasts component of the outcome regression. See ?modelObj and/or ?modelObjSubset for details. NULL is an acceptable input if IPWE is desired or there is no contrast component of the outcome regression model.
data	A data frame of the covariates and tx history
response	The response vector
txName	A vector of characters. The column headers of *data* that correspond to the tx covariate for each decision point. The ordering should be sequential, i.e., the 1st element gives column name for the 1st decision point tx, the 2nd gives column name for the 2nd decision point tx, etc.
regimes	A function or a list of functions. For each decision point, a function defining the tx rule. For example, if the tx rule is : I(eta_1 < x1), regimes is defined as regimes <- function(a,data) {as.numeric(a < data$x1)} THE LAST ARGUMENT IS ALWAYS TAKEN TO BE THE DATA.FRAME
fSet	A function or a list of functions. This argument allows the user to specify the subset of tx options available to a patient or the subset of patients that will be modeled uniquely. see ?fSet for details
refit	No longer used
iter	An integer. See ?iter for details
verbose	A logical. If FALSE, screen prints are suppressed.

Value

An object inheriting from class OptimalSeq

References

Baqun Zhang, Anastasios A. Tsiatis, Eric B. Laber & Marie Davidian, "A Robust Method for Estimating Optimal Treatment Regimes", Biometrics, 68, 1010-1018.

Baqun Zhang, Anastasios A. Tsiatis, Eric B. Laber & Marie Davidian, "Robust estimation of optimal treatment regimes for sequential treatment decisions", Biometrika (2013) pp.1-14.

See Also

Other statistical methods: bowl(), earl(), iqLearn, optimalClass(), owl(), qLearn(), rwl()

Other single decision point methods: earl(), optimalClass(), owl(), qLearn(), rwl()

Other multiple decision point methods: bowl(), iqLearn, optimalClass(), qLearn()

Examples

```
# Load and process data set
data(bmiData)

# define the negative 12 month change in BMI from baseline
y12 <- -100*(bmiData[,6L] - bmiData[,4L])/bmiData[,4L]

# Define the propensity for treatment model and methods.
# Will use constant model for both decision points
moPropen <- buildModelObj(model =  ~ 1,
                          solver.method = 'glm',
                          solver.args = list('family'='binomial'),
                          predict.method = 'predict.glm',
                          predict.args = list(type='response'))
moPropen <- list(moPropen, moPropen)

# outcome model second stage
moMain2 <- buildModelObj(model = ~parentBMI+month4BMI,
                         solver.method = 'lm')

moCont2 <- buildModelObj(model = ~race + parentBMI+month4BMI,
                         solver.method = 'lm')

# outcome model first stage
moMain1 <- buildModelObj(model = ~parentBMI+baselineBMI,
                         solver.method = 'lm')

moCont1 <- buildModelObj(model = ~race + parentBMI+baselineBMI,
                         solver.method = 'lm')

moMain <- list(moMain1, moMain2)
moCont <- list(moCont1, moCont2)

# regime function second stage
regime2 <- function(eta1, eta2, data) {
             tst <- {data$parentBMI > eta1} & {data$month4BMI > eta2}
             rec <- rep('MR', nrow(x = data))
             rec[!tst] <- 'CD'
             return( rec )
           }

# regime function first stage
regime1 <- function(eta1, eta2, data) {
             tst <- {data$parentBMI > eta1} & {data$baselineBMI > eta2}
```

```
              rec <- rep('MR', nrow(x = data))
              rec[!tst] <- 'CD'
              return( rec )
            }

regimes <- list(regime1, regime2)

#### Analysis using AIPW
## Not run:
fit_AIPW <- optimalSeq(moPropen = moPropen,
                       moMain = moMain, moCont = moCont,
                       regimes = regimes,
                       data = bmiData, response = y12,  txName = c('A1','A2'),
                       Domains = cbind(rep(0,4),rep(100,4)),
                       pop.size = 100, starting.values = rep(25,4))

##Available methods

  # Coefficients of the regression objects
  coef(object = fit_AIPW)

  # Description of method used to obtain object
  DTRstep(object = fit_AIPW)

  # Estimated value of the optimal treatment regime for training set
  estimator(x = fit_AIPW)

  # Value object returned by regression methods
  fitObject(object = fit_AIPW)

  # Retrieve the results of genetic algorithm
  genetic(object = fit_AIPW)

  # Estimated optimal treatment and decision functions for training data
  optTx(x = fit_AIPW)

  # Estimated optimal treatment and decision functions for new data
  optTx(x = fit_AIPW, newdata = bmiData)

  # Value object returned by outcome regression method
  outcome(object = fit_AIPW)

  # Plots if defined by regression methods
  dev.new()
  par(mfrow = c(2,4))

  plot(x = fit_AIPW)
  plot(x = fit_AIPW, suppress = TRUE)

  # Retrieve the value object returned by propensity regression method
  propen(object = fit_AIPW)

  # Show main results of method
```

```
  show(object = fit_AIPW)

  # Show summary results of method
  summary(object = fit_AIPW)

## End(Not run)
#### Single Decision Point Analysis using IPW

# Define the propensity for treatment model and methods.
moPropen <- buildModelObj(model =  ~ 1,
                          solver.method = 'glm',
                          solver.args = list('family'='binomial'),
                          predict.method = 'predict.glm',
                          predict.args = list(type='response'))

# regime function second stage
regime <- function(eta1, eta2, data) {
            tst <- {data$parentBMI > eta1} & {data$month4BMI > eta2}
            rec <- rep('MR', nrow(x = data))
            rec[!tst] <- 'CD'
            return( rec )
          }
## Not run:
fit_IPW <- optimalSeq(moPropen = moPropen,
                      regimes = regime,
                      data = bmiData, response = y12,  txName = 'A2',
                      Domains = cbind(rep(0,2),rep(100,2)),
                      pop.size = 100, starting.values = rep(25,2))

##Available methods

  # Coefficients of the regression objects
  coef(object = fit_IPW)

  # Description of method used to obtain object
  DTRstep(object = fit_IPW)

  # Estimated value of the optimal treatment regime for training set
  estimator(x = fit_IPW)

  # Value object returned by regression method
  fitObject(object = fit_IPW)

  # Retrieve the results of genetic algorithm
  genetic(object = fit_IPW)

  # Estimated optimal treatment and decision functions for training data
  optTx(x = fit_IPW)

  # Estimated optimal treatment and decision functions for new data
  optTx(x = fit_IPW, newdata = bmiData)

  # Value object returned by outcome regression method
```

```
  outcome(object = fit_IPW)

  # Plots if defined by outcome regression method
  dev.new()
  par(mfrow = c(2,4))

  plot(x = fit_IPW)
  plot(x = fit_IPW, suppress = TRUE)

  # Retrieve the value object returned by propensity regression method
  propen(object = fit_IPW)

  # Show main results of method
  show(object = fit_IPW)

  # Show summary results of method
  summary(object = fit_IPW)

## End(Not run)
```

`OptimalSeq-class` *Class* `OptimalSeq`

Description

Class `OptimalSeq` contains the results for the estimated optimal tx and value when estimated from a coarsened or missing data perspective.

Slots

`genetic` A list containing the results from the genetic algorithm

`propen` Results of the propensity regression step

`outcome` Results of the outcome regression step

`regime` Results for the regime.

`optimal` Results for the estimated optimal tx and value

`Call` The unevaluated call.

Methods For Post-Processing of Regression Analysis

outcome : Retrieve value object returned by outcome regression methods.

propen : Retrieve value object returned by propensity regression methods.

coef : Retrieve parameter estimates for all regression steps.

fitObject : Retrieve value object returned by regression methods.

plot : Generate plots for regression analyses.

Methods For Accessing Main Results

regimeCoef : Retrieve the estimated regime parameters.

DTRstep : Retrieve description of method used to create object.

estimator : Retrieve the estimated value of the estimated optimal regime for the training data set.

optTx : Retrieve/predict the estimated decision functions and/or optimal tx.

print : Print main results of analysis.

show : Show main results of analysis.

summary : Retrieve summary information.

OptimalSeqCoarsened-class *Class Contains Results for the Coarsened Data IPW/AIPW Method*

Description

Methods for multiple decision point analyses. Class inherits directly from `OptimalSeq` and all methods defined for objects of class `OptimaSeq` are defined for this class.

OptimalSeqMissing-class *Class Contains Results for the Missing Data IPW/AIPW Method*

Description

Methods for single decision point analyses. Class inherits directly from `OptimalSeq` and all methods defined for objects of class `OptimaSeq` are defined for this class.

---

optimObj *Extract Optimization Results*

Description

Retrieves the value object returned by the optimization method for weighted learning methods.

Usage

```
optimObj(object, ...)

## S4 method for signature 'OWL'
optimObj(object, ...)

## S4 method for signature 'RWL'
optimObj(object, ...)

## S4 method for signature 'BOWL'
optimObj(object, ...)

## S4 method for signature 'EARL'
optimObj(object, ...)
```

Arguments

object	A value object returned by a statistical method of DynTxRegime that uses optimization to estimate regime parameters.
...	Ignored.

---

optTx *Extract or Estimate the Optimal Tx and Decision Functions*

Description

If newdata is provided, the results of the statistical method are used to estimate the decision functions and/or optimal tx. If newdata is missing, the estimated decision functions and/or optimal tx obtained for the original training data are returned.

Usage

```
optTx(x, newdata, ...)

## S4 method for signature 'IQLearnFS,data.frame'
optTx(x, newdata, ..., y = NULL, z = NULL, dens = NULL)

## S4 method for signature 'IQLearnFS,missing'
optTx(x, newdata, ..., y = NULL, z = NULL, dens = NULL)
```

Arguments

x	a DynTxRegime Object.
newdata	Optional data.frame if estimates for new patients are desired.
...	Optional additional input.
y	Object of class IQLearnFS

z	Object of class IQLearnFS
dens	one of {norm, nonpar}

Details

Methods are defined for all statistical methods implemented in DynTxRegime.

---

outcome *Retrieve Outcome Regression Analysis*

Description

For statistical methods that require an outcome regression analysis, the value object returned by the modeling function(s) is retrieved.

Usage

```
outcome(object, ...)
```

Arguments

object	A value object returned by a statistical method of DynTxRegime.
...	Ignored.

Details

Methods are defined for all statistical methods implemented in DynTxRegime that use outcome regression.

---

owl *Outcome Weighted Learning*

Description

Outcome Weighted Learning

Usage

```
owl(
  ...,
  moPropen,
  data,
  reward,
  txName,
  regime,
  response,
  lambdas = 2,
  cvFolds = 0L,
  kernel = "linear",
  kparam = NULL,
  surrogate = "hinge",
  verbose = 2L
)
```

Arguments

...	Used primarily to require named input. However, inputs for the optimization methods can be sent through the ellipsis. If surrogate is hinge, the optimization method is kernlab::ipop(). For all other surrogates, stats::optim() is used.
moPropen	An object of class modelObj, which defines the model and R methods to be used to obtain parameter estimates and predictions for the propensity for treatment. See ?moPropen for details.
data	A data frame of the covariates and tx histories
reward	The response vector
txName	A character object. The column header of *data* that corresponds to the tx covariate
regime	A formula object or a character vector. The covariates to be included in classification
response	A numeric vector. The reward. Allows for naming convention followed in most DynTxRegime methods.
lambdas	A numeric object or a numeric vector object giving the penalty tuning parameter. If more than 1 is provided, the finite set of values to be considered in the cross-validation algorithm
cvFolds	If cross-validation is to be used to select the tuning parameters, the number of folds.
kernel	A character object. must be one of {"linear", "poly", "radial"}
kparam	A numeric object of NULL. If kernel = linear, kparam is ignored. If kernel = poly, kparam is the degree of the polynomial If kernel = radial, kparam is the inverse bandwidth of the kernel. If a vector of bandwidth parameters is given, cross-validation will be used to select the parameter
surrogate	The surrogate 0-1 loss function must be one of logit, exp, hinge, sqhinge, huber

verbose — An integer or logical. If 0, no screen prints are generated. If 1, screen prints are generated with the exception of optimization results obtained in iterative algorithm. If 2, all screen prints are generated.

Value

an OWL object

References

Yingqi Zhao, Donglin Zeng, A. John Rush, Michael R. Kosorok (2012) Estimated individualized treatment rules using outcome weighted learning. Journal of the American Statistical Association, 107(409): 1106-1118. PMCID: 3636816

See Also

Other statistical methods: `bowl()`, `earl()`, `iqLearn`, `optimalClass()`, `optimalSeq()`, `qLearn()`, `rwl()`

Other weighted learning methods: `bowl()`, `earl()`, `rwl()`

Other single decision point methods: `earl()`, `optimalClass()`, `optimalSeq()`, `qLearn()`, `rwl()`

Examples

```
# Load and process data set
data(bmiData)

# define the negative 12 month change in BMI from baseline
y12 <- -100*(bmiData[,6L] - bmiData[,4L])/bmiData[,4L]

# propensity model
moPropen <- buildModelObj(model = ~parentBMI+month4BMI,
                          solver.method = 'glm',
                          solver.args = list('family'='binomial'),
                          predict.method = 'predict.glm',
                          predict.args = list(type='response'))

fitOWL <- owl(moPropen = moPropen,
              data = bmiData, reward = y12,  txName = 'A2',
              regime = ~ parentBMI + month4BMI,
              surrogate = 'hinge', kernel = 'linear', kparam = NULL)

##Available methods

  # Coefficients of the propensity score regression
  coef(fitOWL)

  # Description of method used to obtain object
  DTRstep(fitOWL)

  # Estimated value of the optimal treatment regime for training set
```

```
estimator(fitOWL)

# Value object returned by propensity score regression method
fitObject(fitOWL)

# Summary of optimization routine
optimObj(fitOWL)

# Estimated optimal treatment for training data
optTx(fitOWL)

# Estimated optimal treatment for new data
optTx(fitOWL, bmiData)

# Plots if defined by propensity regression method
dev.new()
par(mfrow = c(2,4))

plot(fitOWL)
plot(fitOWL, suppress = TRUE)

# Value object returned by propensity score regression method
propen(fitOWL)

# Parameter estimates for decision function
regimeCoef(fitOWL)

# Show main results of method
show(fitOWL)

# Show summary results of method
summary(fitOWL)
```

`OWL-class` *Class* `OWL`

Description

Class `OWL` contains results for an OWL analysis.

Slots

`analysis` Contains a Learning or LearningMulti object.

`analysis@txInfo` Feasible tx information.

`analysis@propen` Propensity regression analysis.

`analysis@outcome` Outcome regression analysis.

`analysis@cvInfo` Cross-validation analysis if single regime.

`analysis@optim` Optimization analysis if single regime.

`analysis@optimResult` list of cross-validation and optimization results if multiple regimes. optimResult[[i]]@cvInfo and optimResult[[i]]@optim.

`analysis@optimal` Estimated optimal Tx and value.

`analysis@call` Unevaluated call to statistical method.

Methods For Post-Processing of Regression Analysis

propen : Retrieve value object returned by propensity regression methods.

coef : Retrieve parameter estimates for all regression steps.

fitObject : Retrieve value object returned by regression methods.

plot : Generate plots for regression analyses.

Methods For Post-Processing of Optimization Analysis

cvInfo : Retrieve cross-validation results.

optimObj : Retrieve value object returned by optimization method(s).

regimeCoef : Retrieve estimated parameters for optimal tx regime.

Methods For Accessing Main Results

DTRstep : Retrieve description of method used to create object.

estimator : Retrieve the estimated value of the estimated optimal regime for the training data set.

optTx : Retrieve/predict the estimated decision functions and/or optimal tx.

print : Print main results of analysis.

show : Show main results of analysis.

summary : Retrieve summary information.

---

plot *Generates Plots as Defined by Modeling Functions*

Description

Calls plot() method for all regression steps of a statistical method

Arguments

`x`	Value object returned by a statistical method
`y`	Ignored
`suppress`	T/F indicating if titles should be concatenated with information indicating the specific regression step
`...`	Optional additional inputs

Details

Methods are defined for all statistical methods implemented in DynTxRegime.

propen *Retrieve Propensity Regression Analysis*

Description

For statistical methods that require a propensity regression analysis, the value object returned by the modeling function(s) is retrieved.

Usage

```
propen(object, ...)
```

Arguments

object	A value object returned by a statistical method of DynTxRegime.
...	Ignored.

Details

Methods are defined for all statistical methods implemented in DynTxRegime that use propensity regression.

qLearn *A Step of the Q-Learning Algorithm*

Description

Performs a single step of the Q-Learning algorithm. If an object of class `QLearn` is passed through input response, it is assumed that the `QLearn` object is the value object returned from the preceding step of the Q-Learning algorithm, and the value fit by the regression is taken from the `QLearn` object. If a vector is passed through input response, it is assumed that the call if for the first step in the Q-Learning algorithm, and models are fit using the provided response.

Usage

```
qLearn(
  ...,
  moMain,
  moCont,
  data,
  response,
  txName,
  fSet = NULL,
  iter = 0L,
  verbose = TRUE
)
```

Arguments

...	ignored. Provided to require named inputs.
moMain	An object of class modelObj or a list of objects of class modelObjSubset, which define the models and R methods to be used to obtain parameter estimates and predictions for the main effects component of the outcome regression. See ?modelObj and/or ?modelObjSubset for details. NULL is an acceptable value if moCont is defined.
moCont	An object of class modelObj or a list of objects of class modelObjSubset, which define the models and R methods to be used to obtain parameter estimates and predictions for the contrasts component of the outcome regression. See ?modelObj and/or ?modelObjSubset for details. NULL is an acceptable value if moMain is defined.
data	A data frame of covariates and treatment history.
response	A response vector or object of class QLearn from a previous Q-Learning step.
txName	A character string giving column header of treatment variable in data
fSet	NULL or a function. This argument allows the user to specify the subset of treatment options available to a patient. See ?fSet for details of allowed structure
iter	An integer. See ?iter for details
verbose	A logical. If TRUE, screen prints are generated.

Value

An object of class QLearn-class

See Also

Other statistical methods: bowl(), earl(), iqLearn, optimalClass(), optimalSeq(), owl(), rwl()

Other multiple decision point methods: bowl(), iqLearn, optimalClass(), optimalSeq()

Other single decision point methods: earl(), optimalClass(), optimalSeq(), owl(), rwl()

Examples

```
# Load and process data set
data(bmiData)

# define the negative 12 month change in BMI from baseline
y12 <- -100*(bmiData[,6L] - bmiData[,4L])/bmiData[,4L]

# outcome model
moMain <- buildModelObj(model = ~parentBMI+month4BMI,
                        solver.method = 'lm')

moCont <- buildModelObj(model = ~race + parentBMI+month4BMI,
                        solver.method = 'lm')
```

```
#### Second-Stage Analysis
fitSS <- qLearn(moMain = moMain, moCont = moCont,
                data = bmiData, response = y12,  txName = 'A2')

##Available methods

  # Coefficients of the outcome regression objects
  coef(fitSS)

  # Description of method used to obtain object
  DTRstep(fitSS)

  # Estimated value of the optimal treatment regime for training set
  estimator(fitSS)

  # Value object returned by outcome regression method
  fitObject(fitSS)

  # Estimated optimal treatment and decision functions for training data
  optTx(fitSS)

  # Estimated optimal treatment and decision functions for new data
  optTx(fitSS, bmiData)

  # Value object returned by outcome regression method
  outcome(fitSS)

  # Plots if defined by outcome regression method
  dev.new()
  par(mfrow = c(2,4))

  plot(fitSS)
  plot(fitSS, suppress = TRUE)

  # Show main results of method
  show(fitSS)

  # Show summary results of method
  summary(fitSS)

#### First-stage Analysis

# outcome model
moMain <- buildModelObj(model = ~parentBMI+baselineBMI,
                        solver.method = 'lm')

moCont <- buildModelObj(model = ~race + parentBMI+baselineBMI,
                        solver.method = 'lm')

fitFS <- qLearn(moMain = moMain, moCont = moCont,
                data = bmiData, response = fitSS,  txName = 'A1')

##Available methods for fitFS are as shown above for fitSS
```

QLearn-class *Class* QLearn

Description

Class QLearn contains the results for a Q-Learning step

Slots

step An integer indicating the step of the Q-Learning algorithm.

outcome The outcome regression analysis

txInfo The feasible tx information

optimal The estimated optimal tx, decision function, and value

Methods For Post-Processing of Regression Analysis

outcome : Retrieve value object returned by outcome regression methods.

coef : Retrieve parameter estimates for all regression steps.

fitObject : Retrieve value object returned by regression methods.

plot : Generate plots for regression analyses.

Methods For Accessing Main Results

DTRstep : Retrieve description of method used to create object.

estimator : Retrieve the estimated value of the estimated optimal regime for the training data set.

optTx : Retrieve/predict the estimated decision functions and/or optimal tx.

print : Print main results of analysis.

show : Show main results of analysis.

summary : Retrieve summary information.

QLearnObj-class *Class* QLearnObj

Description

Class QLearnObj contains the results for a Q-Learning step

Slots

outcome The outcome regression analysis

txInfo The feasible tx information

optimal The estimated optimal tx, decision function, and value

Methods For Post-Processing of Regression Analysis

outcome : Retrieve value object returned by outcome regression methods.

coef : Retrieve parameter estimates for all regression steps.

fitObject : Retrieve value object returned by regression methods.

plot : Generate plots for regression analyses.

Methods For Accessing Main Results

DTRstep : Retrieve description of method used to create object.

estimator : Retrieve the estimated value of the estimated optimal regime for the training data set.

optTx : Retrieve/predict the estimated decision functions and/or optimal tx.

print : Print main results of analysis.

show : Show main results of analysis.

summary : Retrieve summary information.

regimeCoef *Extract Regime Parameters*

Description

Extract the estimated regime parameters.

Usage

```
regimeCoef(object, ...)
```

Arguments

object	A value object returned by a statistical method of DynTxRegime.
...	Ignored.

Details

Methods are defined for all statistical methods implemented in DynTxRegime that use a non-regression based regime. Specifically, OptimalSeq, OWL, BOWL, RWL, and EARL.

---

residuals *Extract Model Residuals*

Description

Retrieve residuals from an interactive Q-Learning step.

Usage

```
residuals(object, ...)

## S4 method for signature 'IQLearnFS_C'
residuals(object, ...)

## S4 method for signature 'IQLearnFS_VHet'
residuals(object, ...)
```

Arguments

object	A value object returned by iqLearnC() or iqLearnVar()
...	Ignored.

---

rwl *Residual Weighted Learning*

Description

Residual Weighted Learning

Usage

```
rwl(
  ...,
  moPropen,
  moMain,
  data,
  reward,
  txName,
  regime,
  response,
  fSet = NULL,
  lambdas = 2,
  cvFolds = 0L,
  kernel = "linear",
  kparam = NULL,
  responseType = "continuous",
  verbose = 2L
)
```

Arguments

...	Used primarily to require named input. However, inputs for the optimization methods can be sent through the ellipsis. The optimization method is stats::optim().
moPropen	An object of class modelObj or modelObjSubset, which defines the model and R methods to be used to obtain parameter estimates and predictions for the propensity for treatment. See ?moPropen for details.
moMain	An object of class modelObj or modelObjSubset, which defines the model and R methods to be used to obtain parameter estimates and predictions for the main effects of the outcome. See ?modelObj for details.
data	A data frame of the covariates and tx histories
reward	The response vector
txName	A character object. The column header of *data* that corresponds to the tx covariate
regime	A formula object or a list of formula objects. The covariates to be included in classification. If a list is provided, this specifies that there is an underlying subset structure – fSet must then be defined.
response	A numeric vector. The reward. Allows for naming convention followed in most DynTxRegime methods.
fSet	A function or NULL defining subset structure
lambdas	A numeric object or a numeric vector object giving the penalty tuning parameter. If more than 1 is provided, the finite set of values to be considered in the cross-validation algorithm
cvFolds	If cross-validation is to be used to select the tuning parameters, the number of folds.
kernel	A character object. must be one of {"linear", "poly", "radial"}

kparam	A numeric object of NULL. If kernel = linear, kparam is ignored. If kernel = poly, kparam is the degree of the polynomial If kernel = radial, kparam is the inverse bandwidth of the kernel. If a vector of bandwidth parameters is given, cross-validation will be used to select the parameter
responseType	A character indicating if response is continuous, binary or count data.
verbose	An integer or logical. If 0, no screen prints are generated. If 1, screen prints are generated with the exception of optimization results obtained in iterative algorithm. If 2, all screen prints are generated.

Value

an RWL object

References

Xin Zhou, Nicole Mayer-Hamblett, Umer Khan, and Michael R Kosorok (2017) Residual weighted learning for estimating individualized treatment rules. Journal of the American Statistical Association, 112, 169–187.

See Also

Other statistical methods: `bowl()`, `earl()`, `iqLearn`, `optimalClass()`, `optimalSeq()`, `owl()`, `qLearn()`

Other weighted learning methods: `bowl()`, `earl()`, `owl()`

Other single decision point methods: `earl()`, `optimalClass()`, `optimalSeq()`, `owl()`, `qLearn()`

Examples

```
## Not run:
# Load and process data set
data(bmiData)

# define the negative 12 month change in BMI from baseline
y12 <- -100*(bmiData[,6L] - bmiData[,4L])/bmiData[,4L]

# propensity model
moPropen <- buildModelObj(model = ~parentBMI+month4BMI,
                          solver.method = 'glm',
                          solver.args = list('family'='binomial'),
                          predict.method = 'predict.glm',
                          predict.args = list(type='response'))

# outcome model
moMain <- buildModelObj(model = ~parentBMI+month4BMI,
                        solver.method = 'lm')

fitRWL <- rwl(moPropen = moPropen, moMain = moMain,
              data = bmiData, reward = y12,  txName = 'A2',
              regime = ~ parentBMI + month4BMI,
              kernel = 'radial', kparam = 1.5)
```

```
##Available methods

  # Coefficients of the regression objects
  coef(fitRWL)

  # Description of method used to obtain object
  DTRstep(fitRWL)

  # Estimated value of the optimal treatment regime for training set
  estimator(fitRWL)

  # Value object returned by regression methods
  fitObject(fitRWL)

  # Summary of optimization routine
  optimObj(fitRWL)

  # Estimated optimal treatment for training data
  optTx(fitRWL)

  # Estimated optimal treatment for new data
  optTx(fitRWL, bmiData)

  # Value object returned by outcome regression method
  outcome(fitRWL)

  # Plots if defined by regression methods
  dev.new()
  par(mfrow = c(2,4))

  plot(fitRWL)
  plot(fitRWL, suppress = TRUE)

  # Value object returned by propensity score regression method
  propen(fitRWL)

  # Parameter estimates for decision function
  regimeCoef(fitRWL)

  # Show main results of method
  show(fitRWL)

  # Show summary results of method
  summary(fitRWL)

## End(Not run)
```

RWL-class *Class* RWL

Description

Class RWL contains results for an RWL analysis.

Slots

`responseType` character indicating type of response

`residuals` vector of outcome residuals

`beta` vector of regime parameters

`analysis` Contains a Learning or LearningMulti object

`analysis@txInfo` Feasible tx information

`analysis@propen` Propensity regression analysis

`analysis@outcome` Outcome regression analysis

`analysis@cvInfo` Cross-validation analysis if single regime

`analysis@optim` Optimization analysis if single regime

`analysis@optimResult` list of cross-validation and optimization results if multiple regimes. optimResult[[i]]@cvInfo and optimResult[[i]]@optim

`analysis@optimal` Estimated optimal Tx and value

`analysis@Call` Unevaluated Call

Methods For Post-Processing of Regression Analysis

outcome : Retrieve value object returned by outcome regression methods.

propen : Retrieve value object returned by propensity regression methods.

coef : Retrieve parameter estimates for all regression steps.

fitObject : Retrieve value object returned by regression methods.

plot : Generate plots for regression analyses.

Methods For Post-Processing of Optimization Analysis

cvInfo : Retrieve cross-validation results.

optimObj : Retrieve value object returned by optimization method(s).

regimeCoef : Retrieve estimated parameters for optimal tx regime.

Methods For Accessing Main Results

DTRstep : Retrieve description of method used to create object.

estimator : Retrieve the estimated value of the estimated optimal regime for the training data set.

optTx : Retrieve/predict the estimated decision functions and/or optimal tx.

print : Print main results of analysis.

show : Show main results of analysis.

summary : Retrieve summary information.

sd *Standard Deviation*

Description

Retrieve the standard deviation of the residuals for the first-stage contrasts regression in the interactive Q-Learning algorithm.

Usage

```
sd(x, na.rm=FALSE)

  ## S4 method for IQLearnFS_C
  sd(x, na.rm=FALSE)
```

Arguments

x	An object of class IQLearnFS_C
na.rm	logical. Should missing values be removed?

summary *Result Summaries*

Description

Returns a list of the primary results, including regression results, optimization results, estimated tx and value, etc.

Usage

```
summary(object, ...)
```

Arguments

object	Value object returned by a statistical method
...	Optional additional inputs

Details

Methods are defined for all statistical methods implemented in DynTxRegime.

The exact structure of the returned list will vary depending on the statistical method.

Index