Package ‘superMICE’

May 4, 2022

Type Package

Title SuperLearner Method for MICE

Version 1.1.1

Author Aaron B. Shev

Maintainer Aaron B. Shev <abshev@ucdavis.edu>


Imports stats, mice, SuperLearner

Suggests arm, bartMachine, class, e1071, earth, extraTrees, gbm, glmnet, ipred, KernelKnn, kernlab, LogicReg, MASS, mnet, party, polsplines, randomForest, ranger, rpart, speedglm, spls, xgboost

License GPL-3

Encoding UTF-8

RoxygenNote 7.1.2

NeedsCompilation no

Repository CRAN

Date/Publication 2022-05-04 20:00:02 UTC

R topics documented:

  binarySuperLearner ................................. 2
  continuousSuperLearner ................................ 2
  gaussianKernel ...................................... 3
  jackknifeBandwidthSelection .......................... 4
  jackknifeVariance .................................... 4
  localImputation ..................................... 5
  mice.impute.SuperLearner ......................... 6

Index 9
**binarySuperLearner**

Function to generate imputations using SuperLearner for data with a binary outcome.

**Usage**

```r
binarySuperLearner(y, x, wy, SL.library, ...)
```

**Arguments**

- `y`: Vector of observed values of the variable to be imputed.
- `x`: Numeric matrix of variables to be used as predictors in SuperLearner methods with rows corresponding to values in Y.
- `wy`: Logical vector of length `length(y)`. A TRUE value indicates locations in y for which imputations are created.
- `SL.library`: Either a character vector of prediction algorithms or a list containing character vectors. A list of functions included in the SuperLearner package can be found with `SuperLearner::listWrappers()`.
- `...`: Further arguments passed to SuperLearner.

**Value**

Binary Vector of randomly drawn imputed values.

---

**continuousSuperLearner**

Function to generate imputations using SuperLearner for data with a continuous outcome

**Usage**

```r
continuousSuperLearner(y, x, wy, SL.library, kernel, bw, bw.update, ...)
```

**Description**

Function to generate imputations using SuperLearner for data with a continuous outcome.
**Arguments**

- **y**: Vector of observed and missing/imputed values of the variable to be imputed.
- **x**: Numeric matrix of variables to be used as predictors in SuperLearner models with rows corresponding to observed values of the variable to be imputed and columns corresponding to individual predictor variables.
- **wy**: Logical vector. A TRUE value indicates locations in `y` that are missing or imputed.
- **SL.library**: Either a character vector of prediction algorithms or a list containing character vectors. A list of functions included in the SuperLearner package can be found with `SuperLearner::listWrappers()`.
- **kernel**: one of `gaussian`, `uniform`, or `triangular`. Specifies the kernel to be used in estimating the distribution around a missing value.
- **bw**: NULL or numeric value for bandwidth of kernel function (as standard deviations of the kernel).
- **bw.update**: logical indicating whether bandwidths should be computed every iteration or only on the first iteration. Default is `TRUE`, but `FALSE` may speed up the run time at the cost of accuracy.
- **...**: further arguments passed to `SuperLearner()`.

**Value**

umeric vector of randomly drawn imputed values.

---

**gaussianKernel**

*Kernel functions used for local imputation*

**Description**

Kernel functions used for local imputation

**Usage**

gaussianKernel(x, xcenter, bw = 1, lambda = NULL)

**Arguments**

- **x**: numeric vector of values to weight.
- **xcenter**: numeric value to center the kernel.
- **bw**: bandwidth of the kernel.
- **lambda**: kernel radius, function of `bw`.

**Value**

kernel values for `x` centered at `xcenter`. 

---

**gaussianKernel**

*Kernel functions used for local imputation*


**jackknifeBandwidthSelection**

*Jackknife method for selection bandwidth*

**Description**

Jackknife method for selection bandwidth

**Usage**

```
jackknifeBandwidthSelection(i, bwGrid, preds, y, delta, kernel)
```

**Arguments**

- `i` integer referring to the index of the missing value to be imputed.
- `bwGrid` numeric vector of candidate bandwidth values
- `preds` numeric vector of predicted values for missing observations
- `y` numeric vector of length `n` of observed and imputed values.
- `delta` Binary vector of length length(`y`) indicating missingness. 1 where `y` is observed and 0 where `y` is missing.
- `kernel` one of `gaussian`, `uniform`, or `triangular`. Specifies the kernel to be used in estimating the distribution around a missing value.

**Value**

`bandwidth`

---

**jackknifeVariance**

*Computes jackknife variance*

**Description**

Computes jackknife variance

**Usage**

```
jackknifeVariance(j, kernMatrix, delta, y)
```

**Arguments**

- `j` integer index for deleted observation in the jackknife procedure.
- `kernMatrix` \((n-1)\) by \(m\) matrix of kernel values centered at missing observation \(j\) where \(n\) is the total number of observations and \(m\) is the number of candidate bandwidths.
- `delta` Binary vector of length \(n\) indicating missingness. 1 where `y` is observed and 0 where `y` is missing.
- `y` numeric vector of length \(n\) of observed values and imputed values.
localImputation

Value
returns a single numeric value for the estimate of the jackknife variance.

Description
Function to generate imputations using non-parametric and semi-parametric local imputation methods.

Usage
localImputation(
  i,
  preds,
  y,
  delta,
  bw = NULL,
  kernel = c("gaussian", "uniform", "triangular")
)

Arguments

i integer referring to the index of the missing value to be imputed.
preds numeric vector of predictions of missing values from SuperLearner.
y numeric vector for variable to be imputed.
delta binary vector of length length(y) indicating missingness. 1 where y is observed and 0 where y is missing.
bw NULL or numeric value for bandwidth of kernel function (as standard deviations of the kernel).
kernel one of gaussian, uniform, or triangular. Specifies the kernel to be used in estimating the distribution around a missing value.

Value
numeric vector of randomly drawn imputed values.
mice.impute.SuperLearner

SuperLearner method for mice package.

Description

Method for the mice package that uses SuperLearner as the predictive algorithm. Model fitting is done using the SuperLearner package.

Usage

mice.impute.SuperLearner(
  y,
  ry,
  x,
  wy = NULL,
  SL.library,
  kernel = c("gaussian", "uniform", "triangular"),
  bw = c(0.1, 0.2, 0.25, 0.3, 0.5, 1, 2.5, 5, 10, 20),
  bw.update = TRUE,
  ...
)

Arguments

y Vector to be imputed
ry Logical vector of length length(y) indicating the the subset y[ry] of elements in y to which the imputation model is fitted. The ry generally distinguishes the observed (TRUE) and missing values (FALSE) in y.
x Numeric design matrix with length(y) rows with predictors for y. Matrix x may have no missing values.
wy Logical vector of length length(y). A TRUE value indicates locations in y for which imputations are created.
SL.library For SuperLearner: Either a character vector of prediction algorithms or list containing character vectors as specified by the SuperLearner package. See details below.
kernel One of "gaussian", "uniform", "triangular". Kernel function used to compute weights.
bw NULL or numeric value for bandwidth of kernel function (as standard deviations of the kernel).
bw.update logical indicating whether bandwidths should be computed every iteration or only on the first iteration. Default is TRUE, but FALSE may speed up the run time at the cost of accuracy.
... Further arguments passed to SuperLearner.
mice.impute.SuperLearner

Details

mice.impute.SuperLearner() is a method for use with the mice() function that implements the ensemble predictive model, SuperLearner (van der Laan, 2011), into the mice (van Buuren, 2011) multiple imputation procedure. This function is never called directly, instead a user that wishes to use SuperLearner in MICE simply needs to set the argument method = “SuperLearner” in the call to mice(). Arguments for the SuperLearner() function are passed from mice as extra arguments in the mice() call.

All MICE methods randomly generate imputed values for a number of data sets. The approach of SuperMICE is to estimate parameters for a normal distribution centered at the point estimate for an imputed value predicted by a SuperLearner model. The point estimates are obtained by fitting a selection of different predictive models on complete cases and determining an optimal weighted average of candidate models to predict the missing cases. SuperMICE uses the implementation of SuperLearner found in the SuperLearner package. The models to be used with SuperLearner() are supplied by the user as a character vector. For a full list of available methods see listWrappers().

SuperLearner models do not produce standard errors for estimates, so instead we use a kernel based estimate of local variance around each point estimate as the variance parameter in the normal distribution used to randomly sample values. The kernel can be set by the user with the kernel argument as either a gaussian kernel, uniform kernel, or triangular kernel. The user must also supply a list of candidate bandwidths in the bw argument as a numeric vector. For more information on the variance and bandwidth selection see Laqueur, et. al (2021). In every iteration the mice procedure, the optimal bandwidth is reselected. This may be changed to select the bandwidth only on the first iteration to speed up the total run time of the imputation by changing bw.update to FALSE; however this may bias your results. Note that this only applies to continuous response variables. In the binary case the variance is a function of the SuperLearner estimate.

Value

Vector with imputed data, same type as y, and of length sum(wy)

References


See Also

mice(), SuperLearner()

Examples

#Multiple imputation with missingness on a continuous variable.
# Randomly generated data with missingness in x2. The probability of x2 being missing increases with with value of x1.

```r
n <- 20
pmissing <- 0.10
x1 <- runif(n, min = -3, max = 3)
x2 <- x1^2 + rnorm(n, mean = 0, sd = 1)
error <- rnorm(n, mean = 0, sd = 1)
y <- x1 + x2 + error
f <- ecdf(x1)
x2 <- ifelse(runif(x2) < (f(x1) * 2 * pmissing), NA, x2)
dat <- data.frame(y, x1, x2)

# Create vector of SuperLearner method names
# Note: see SuperLearner::listWrappers() for a full list of methods available.
SL.lib <- c("SL.mean", "SL.glm")

# Run mice().
# Note 1: m >= 30 and maxit >= 10 are recommended outside of this toy example
# Note 2: a denser bandwidth grid is recommended, see default for bw argument for example.
imp.SL <- mice::mice(dat, m = 2, maxit = 2,
                     method = "SuperLearner",
                     print = TRUE, SL.library = SL.lib,
                     kernel = "gaussian",
                     bw = c(0.25, 1, 5))
```
Index

binarySuperLearner, 2
continuousSuperLearner, 2
gaussianKernel, 3
jackknifeBandwidthSelection, 4
jackknifeVariance, 4
listWrappers(), 7
localImputation, 5
mice(), 7
mice.impute.SuperLearner, 6
SuperLearner, 7
SuperLearner(), 7