Package 'maic'

May 11, 2021

Type Package

Title Matching-Adjusted Indirect Comparison

Version 0.1.3

Date 2021-05-10

Maintainer Rob Young <rob.young@heor.co.uk>

Depends R (>= 3.0.0)

Description A generalised workflow for generation of subject weights to be used in Matching-Adjusted Indirect Comparison (MAIC) per Signorovitch et al (2012) <doi:10.1016/j.jval.2012.05.004>, Signorovitch et al (2010) <doi:10.2165/11538370-000000000-00000>. In MAIC, unbiased comparison between outcomes of two trials is facilitated by weighting the subject-level outcomes of one trial with weights derived such that the weighted aggregate measures of the prognostic or effect modifying variables are equal to those of the sample in the comparator trial. The functions and classes included in this package wrap and abstract the process demonstrated in the UK National Institute for Health and Care Excellence Decision Support Unit (NICE DSU)'s example (Phillippo et al, (2016) [see URL]), providing a repeatable and easily specifiable workflow for producing multiple comparison variable sets against a variety of target studies, with preprocessing for a number of aggregate target forms (e.g. mean, median, domain limits).

License GPL-3

Encoding UTF-8

URL https://github.com/heorltd/maic,
http://nicedsu.org.uk/technical-support-documents/population-adjusted-indirect-comparisons-maic-and-stc/

Imports Hmisc, matrixStats, weights

RoxygenNote 7.1.1

Suggests testthat

NeedsCompilation no

Author Rob Young [aut, cre]

Repository CRAN

Date/Publication 2021-05-11 06:50:05 UTC

R topics documented:

`createMAICInput` *Construct a MAIC input matrix*

Description

From index patient level data and a set of target baseline characteristics, construct the input matrix to the maic.

Usage

```
createMAICInput(index, target, dictionary, matching.variables, x = FALSE)
```

Arguments

`index`	A matrix or data.frame containing patient-level data
`target`	A list containing target summary data
`dictionary`	A data frame containing the columns "match.id", "target.variable", "index.variable" and "match.type"
`matching.variables`	A character vector indicating the match.id to use
`x`	Return subject level inputs?

Details

The `dictionary` is a data frame containing at least 4 vectors:

- "match.id" - the name of the match, used to refer to it in the matching.variables list
- "target.variable" - the name of the variable in the target values list use to inform the matching. Use dependent on type
- "index.variable" - the name of the variable in the index data frame to match on.
- "match.type" - A string indicating the match type to use. The following values are accepted:
  - minimum - records with index values lower than the target variable will be assigned 0 weight

  - maximum - records with index values greater than the target variable will be assigned 0 weight
  - median - records with index values greater than the target variable will be assigned a value of 1, those lower 0. The target for matching will be a mean of 0.5
  - quantile.X - Generalisation of the median code. records with index values greater than the target variable will be assigned a value of 1, those lower 0. The target for matching will be a mean of 0.X
  - mean - records will match index value directly onto target value
  - proportion - as mean, with index values encoded as 1 = true, 0 = false. If target proportion is exclusive (0 or 1 exactly) then excluded members of the index population shall receive no weighting.
  - sd - a matching on the square of the index value on the sum of the square of the target mean and target standard deviation. The target mean is provided by the "supplementary.target.variable"
  - var - a matching on the square of the index value on the sum of the square of the target mean and the variance specified by the target variable. The target mean is provided by the "supplementary.target.variable"

In addition, the following vector may be necessary:

- "supplementary.target.variable" - The name of the variable in the target values list that provides e.g. the mean for sd and var matching.

and, for estimating some p-values on difference (e.g. in proportion)

- "sample.size.variable" - The name of the variable in the target values list that provides the number of subjects in the sample. Only for reporting.

It is possible to use these match types to match on other variables by pre-processing the input correctly.

Finally, the `matching.variables` is a list or character vector containing `match.ids` to be acted upon in this MAIC.

Value

An object of class `maic.input`

Examples

```
target <- c("Air.Flow" = 60,
            "Water.Temp" = 21,
            "Prop.Acid.Conc.LT.90" = 0.7,
            "min.air.flow" = 55)

stackloss$match.conc.lt.90 <-
  ifelse(stackloss$Acid.Conc. < 90, 1, 0)

dict <- data.frame(
  "match.id" =
    c("airflow", "watertemp",
```

```
      "acidconc", "min.airflow"),
  "target.variable" =
    c("Air.Flow", "Water.Temp",
      "Prop.Acid.Conc.LT.90", "min.air.flow"),
  "index.variable" =
    c("Air.Flow", "Water.Temp",
      "match.conc.lt.90", "Air.Flow"),
  "match.type" =
    c("mean", "mean", "proportion", "min"),
  stringsAsFactors = FALSE)

ipmat <- createMAICInput(
  index = stackloss,
  target = target,
  dictionary = dict,
  matching.variables =
    c("airflow", "watertemp",
      "acidconc", "min.airflow"))

wts <- maicWeight(ipmat)

rcv <- reportCovariates(
  stackloss, target, dict,
  matching.variables =
    c("airflow", "watertemp",
      "acidconc", "min.airflow"),
  wts)
```

`maicMatching` *calculate MAIC weights*

Description

From index patient level data and a set of target baseline characteristics, calculate MAIC weights.

Usage

```
maicMatching(
  index,
  target,
  dictionary,
  matching.variables,
  reporting.variables = NULL
)
```

Arguments

`index`	A matrix or data.frame containing patient-level data
`target`	A list containing target summary data

`dictionary` A data frame containing the columns "match.id", "target.variable", "index.variable" and "match.type"

`matching.variables` A character vector indicating the match.id to use

`reporting.variables` A optional character vector of matches to report upon (defaults to `matching.variables`)

Details

The `dictionary` is a data frame containing at least 4 vectors:

- "match.id" - the name of the match, used to refer to it in the matching.variables list
- "target.variable" - the name of the variable in the target values list use to inform the matching. Use dependent on type
- "index.variable" - the name of the variable in the index data frame to match on.
- "match.type" - A string indicating the match type to use. The following values are accepted:
  - minimum - records with index values lower than the target variable will be discarded
  - maximum - records with index values greater than the target variable will be discarded
  - median - records with index values greater than the target variable will be assigned a value of 1, those lower 0. The target for matching will be a mean of 0.5
  - quantile.X - Generalisation of the median code. records with index values greater than the target variable will be assigned a value of 1, those lower 0. The target for matching will be a mean of 0.X
  - mean - records will match index value directly onto target value
  - proportion - as mean, with index values encoded as 1 = true, 0 = false. If target proportion is exclusive (0 or 1 exactly) then excluded members of the index population shall receive no weighting.
  - sd - a matching on the square of the index value on the sum of the square of the target mean and target standard deviation. The target mean is provided by the "supplementary.target.variable"
  - var - a matching on the square of the index value on the sum of the square of the target mean and the variance specified by the target variable. The target mean is provided by the "supplementary.target.variable"

In addition, the following vector may be necessary:

- "supplementary.target.variable" - The name of the variable in the target values list that provides e.g. the mean for sd and var matching.

It is possible to use these match types to match on other variables, e.g. variance, by pre-processing the input correctly.

Finally, the `matching.variables` is a list or character vector containing `match.ids` to be acted upon in this MAIC.

Value

An object of class `MaicAnalysis`, with components `weights` and `aggregate`, containing the weights vector and the covariate aggregate data respectively

Examples

```
target <- c("Air.Flow" = 60,
            "Water.Temp" = 21,
            "Prop.Acid.Conc.LT.90" = 0.7,
            "min.air.flow" = 55)

stackloss$match.conc.lt.90 <-
  ifelse(stackloss$Acid.Conc. < 90, 1, 0)

dict <- data.frame(
  "match.id" =
    c("airflow", "watertemp",
      "acidconc", "min.airflow"),
  "target.variable" =
    c("Air.Flow", "Water.Temp",
      "Prop.Acid.Conc.LT.90", "min.air.flow"),
  "index.variable" =
    c("Air.Flow", "Water.Temp",
      "match.conc.lt.90", "Air.Flow"),
  "match.type" =
    c("mean", "mean", "proportion", "min"),
  stringsAsFactors = FALSE)

weightObj <- maicMatching(
  index = stackloss,
  target = target,
  dictionary = dict,
  matching.variables =
    c("airflow", "watertemp",
      "acidconc", "min.airflow"))
```

`maicWeight` *Calculate MAIC weights*

Description

This function calculates the weights to apply to records for Matching-Adjusted Indirect Comparison (MAIC), from either a raw input matrix or a `maic.input` object

Usage

```
maicWeight(x, opt = TRUE, keep.x = TRUE, ...)
```

Arguments

`x`	Either a `maic.input` object or a MAIC input matrix
`opt`	return the optim object as attribute
`keep.x`	return the input matrix as an attribute
`...`	Optional arguments to `optim`

Value

A numeric vector of weights corresponding to the rows in the input matrix

Examples

```
target <- c("Air.Flow" = 60,
            "Water.Temp" = 21,
            "Prop.Acid.Conc.LT.90" = 0.7,
            "min.air.flow" = 55)

stackloss$match.conc.lt.90 <-
  ifelse(stackloss$Acid.Conc. < 90, 1, 0)

dict <- data.frame(
  "match.id" =
    c("airflow", "watertemp",
      "acidconc", "min.airflow"),
  "target.variable" =
    c("Air.Flow", "Water.Temp",
      "Prop.Acid.Conc.LT.90", "min.air.flow"),
  "index.variable" =
    c("Air.Flow", "Water.Temp",
      "match.conc.lt.90", "Air.Flow"),
  "match.type" =
    c("mean", "mean", "proportion", "min"),
  stringsAsFactors = FALSE)

ipmat <- createMAICInput(
  index = stackloss,
  target = target,
  dictionary = dict,
  matching.variables =
    c("airflow", "watertemp",
      "acidconc", "min.airflow"))

wts <- maicWeight(ipmat)

rcv <- reportCovariates(
  stackloss, target, dict,
  matching.variables =
    c("airflow", "watertemp",
      "acidconc", "min.airflow"),
  wts)
```

`reportCovariates` *Calculate the rebalanced covariates*

Description

This function calculates the raw, target and achieved covariates given a set of weights. Note that for mean values, bootstrapped standard errors are used and so downstream values (such as p-values for difference) may differ from run to run if the random number stream is not consistent

Usage

```
reportCovariates(
  index,
  target,
  dictionary,
  matching.variables,
  weights,
  tidy = TRUE,
  var.method = c("ML", "unbiased")
)
```

Arguments

index	A matrix or data.frame containing patient-level data
target	A list containing target summary data
dictionary	A data frame containing the columns "match.id", "target.variable", "index.variable" and "match.type"
matching.variables	A character vector indicating the match.id to use
weights	A numeric vector with weights corresponding to the index data rows
tidy	A boolean - return as a data frame (otherwise list)
var.method	Estimator type passed through to wtd.var. Defaults to ML, as Bessel's correction not used in weights generation.

Value

An object of class maic.covariates

Examples

```
target <- c("Air.Flow" = 60,
            "Water.Temp" = 21,
            "Prop.Acid.Conc.LT.90" = 0.7,
            "min.air.flow" = 55)

stackloss$match.conc.lt.90 <-
  ifelse(stackloss$Acid.Conc. < 90, 1, 0)

dict <- data.frame(
  "match.id" =
    c("airflow", "watertemp",
      "acidconc", "min.airflow"),
```

```
  "target.variable" =
    c("Air.Flow", "Water.Temp",
      "Prop.Acid.Conc.LT.90", "min.air.flow"),
  "index.variable" =
    c("Air.Flow", "Water.Temp",
      "match.conc.lt.90", "Air.Flow"),
  "match.type" =
    c("mean", "mean", "proportion", "min"),
  stringsAsFactors = FALSE)

ipmat <- createMAICInput(
  index = stackloss,
  target = target,
  dictionary = dict,
  matching.variables =
    c("airflow", "watertemp",
      "acidconc", "min.airflow"))

wts <- maicWeight(ipmat)

rcv <- reportCovariates(
  stackloss, target, dict,
  matching.variables =
    c("airflow", "watertemp",
      "acidconc", "min.airflow"),
  wts)
```

Index