Package 'smcfcs'

March 25, 2025

Title Multiple Imputation of Covariates by Substantive Model Compatible Fully Conditional Specification

Version 2.0.0

URL https://github.com/jwb133/smcfcs

Description Implements multiple imputation of missing covariates by Substantive Model Compatible Fully Conditional Specification. This is a modification of the popular FCS/chained equations multiple imputation approach, and allows imputation of missing covariate values from models which are compatible with the user specified substantive model.

Depends R (>= 3.1.2)

License GPL-3

LazyData true

Imports MASS, survival, VGAM, stats, rlang, checkmate, abind, brglm2

Suggests knitr, rmarkdown, mitools, ggplot2, kmi, flexsurv, stringr

VignetteBuilder knitr

RoxygenNote 7.3.2

Encoding UTF-8

NeedsCompilation no

Author Jonathan Bartlett [aut, cre],
Ruth Keogh [aut],
Edouard F. Bonneville [aut],
Lars van der Burg [aut],
Claus Thorn Ekstrøm [ctb]

Maintainer Jonathan Bartlett <jonathan.bartlett1@lshtm.ac.uk>

Repository CRAN

Date/Publication 2025-03-25 05:50:02 UTC

Contents

`ex_cc` *Simulated case cohort data*

Description

A dataset containing simulated case cohort data, where the sub-cohort was a 10% random sample of the full cohort.

Usage

```
ex_cc
```

Format

A data frame with 1571 rows and 7 variables:

t Time to event or censoring

d Indicator of whether event 1 occurred (d=1), or not (d=0)

x Partially observed continuous covariate

z Fully observed covariate

in.subco A binary indicator of whether the subject is in the sub-cohort

id An id variable

entertime The entry time variable to be used in the analysis

ex_coarsening *Simulated example data with a coarsened factor covariate*

Description

A simulated dataset with a factor level covariate subject to coarsening.

Usage

```
ex_coarsening
```

Format

A data frame with 100 rows and 4 variables:

x Factor variable with 3 levels (a,b,c) and some missing values

xobs Variable which contains coarsening information for x

z Fully observed continuous covariate

y Fully observed continuous outcome

ex_compet *Simulated example data with competing risks outcome and partially observed covariates*

Description

A dataset containing simulated competing risks data. There are two competing risks, and some times are also censored.

Usage

```
ex_compet
```

Format

A data frame with 1000 rows and 4 variables:

t Time to event or censoring

d Indicator of whether event 1 occurred (d=1), event 2 occurred (d=2) or individual was censored (d=0)

x1 Partially observed binary covariate, with linear effects on log competing risk hazards

x2 Partially observed normally distributed (conditional on x1) covariate, with linear effects on log competing risk hazards

ex_coxquad — *Simulated example data with time to event outcome and quadratic covariate effects*

Description

A dataset containing simulated data where a time to event outcome depends quadratically on a partially observed covariate.

Usage

```
ex_coxquad
```

Format

A data frame with 1000 rows and 5 variables:

t Time to event or censoring

d Binary indicator of whether event occurred or individual was censored

z Fully observed covariate, with linear effect on outcome (on log hazard scale)

x Partially observed normally distributed covariate, with quadratic effect on outcome (on log hazard scale)

v An auxiliary variable (i.e. not contained in the substantive model)

ex_dtsam — *Simulated discrete time survival data set*

Description

A dataset containing simulated discrete time survival data.

Usage

```
ex_dtsam
```

Format

A data frame with 1000 rows and 8 variables:

x1 A binary variable with missing values

x2 A fully observed continuous variable

failtime The discrete failure/censoring time

d Indicator of failure (=1) or censoring (=0)

ex_finegray — *Simulated example data with competing risks outcome and partially observed covariates*

Description

A dataset containing simulated competing risks data. There are two competing risks, and some times are also censored. Proportionality holds on the subdistribution hazard scale for cause 2, where for dataset 'ex_compet' it instead holds on the cause-specific hazard scale.

Usage

```
ex_finegray
```

Format

A data frame with 1000 rows and 4 variables:

times Time to event or censoring

d Indicator of whether event 1 occurred (d=1), event 2 occurred (d=2) or individual was censored (d=0)

x1 Partially observed binary covariate, with linear effects on log subdistribution hazard of cause 1

x2 Partially observed normally distributed (conditional on x1) covariate, with linear effects on log subdistribution hazard of cause 1

ex_flexsurv — *Simulated example data with time-to-event Weibull outcome and two covariates*

Description

A dataset containing simulated data where the time-to-event outcome is Weibull distributed with two covariates, one of which is partially observed.

Usage

```
ex_flexsurv
```

Format

A data frame with 1000 rows and 4 variables:

t Time to event (d=1) or censoring (d=0)

d Event indicator

x Partially observed binary covariate

z Fully observed continuous covariate

ex_lininter — *Simulated example data with continuous outcome and interaction between two partially observed covariates*

Description

A dataset containing simulated data where the outcome depends on both main effects and interaction of two partially observed covariates.

Usage

```
ex_lininter
```

Format

A data frame with 1000 rows and 4 variables:

y Continuous outcome

x1 Partially observed normally distributed covariate

x2 Partially observed binary covariate

ex_linquad — *Simulated example data with continuous outcome and quadratic covariate effects*

Description

A dataset containing simulated data where the outcome depends quadratically on a partially observed covariate.

Usage

```
ex_linquad
```

Format

A data frame with 1000 rows and 4 variables:

y Continuous outcome

z Fully observed covariate, with linear effect on outcome

x Partially observed normally distributed covariate, with quadratic effect on outcome

v An auxiliary variable (i.e. not contained in the substantive model)

ex_logisticquad — *Simulated example data with binary outcome and quadratic covariate effects*

Description

A dataset containing simulated data where the binary outcome depends quadratically on a partially observed covariate.

Usage

```
ex_logisticquad
```

Format

A data frame with 1000 rows and 4 variables:

y Binary outcome

z Fully observed covariate, with linear effect on outcome (on log odds scale)

x Partially observed normally distributed covariate, with quadratic effect on outcome (on log odds scale)

v An auxiliary variable (i.e. not contained in the substantive model)

ex_ncc — *Simulated nested case-control data*

Description

A dataset containing simulated nested case-control data.

Usage

```
ex_ncc
```

Format

A data frame with 728 rows and 8 variables:

t Time to event or censoring

d Indicator of whether event 1 occurred (d=1), or not (d=0)

x Partially observed binary covariate

z Fully observed covariate

id An id variable

numrisk Number of patients at risk at time of case's event

setno The case-control set number

case Binary indicator of case (=1) or control (=0)

ex_poisson *Simulated example data with count outcome, modelled using Poisson regression*

Description

A dataset containing simulated data where the count outcome depends on two covariates, x and z, with missing values in x. The substantive model is Poisson regression.

Usage

```
ex_poisson
```

Format

A data frame with 1000 rows and 3 variables:

y Count outcome

z Fully observed covariate, with linear effect on outcome

x Partially observed normally distributed covariate, with linear effect on outcome

plot.smcfcs *Assess convergence of a smcfcs object*

Description

Visualises the contents of smCoefIter. Specifically, it plots the parameter estimates of the substantive model against the number of iterations from the imputation procedure. This is done for each regression coefficient, and each line corresponds to an imputed dataset.

Usage

```
## S3 method for class 'smcfcs'
plot(x, include = "all", ...)
```

Arguments

x	An object of class 'smcfcs'
include	Character vector of coefficient names for which to return the convergence plot. Default is "all" and returns plots for all coefficients in a facetted manner. Recommendation is to plot first with include = "all", and then select coefficient names to zoom in to. For competing risks, the coefficients are indexed by their cause. E.g. for coefficient of a variable x1 in a model for cause 2, will be labelled "x1-cause2".
...	Additional parameters to pass on to ggplot2::facet_wrap(), eg. nrow = 2

Details

Requires loading of ggplot2 plotting library.

Value

A ggplot2 object, containing the convergence plots, facetted per covariate in the substantive model

Author(s)

Edouard F. Bonneville <e.f.bonneville@lumc.nl>

Examples

```
## Not run:
# Use simulated competing risks example in package
imps <- smcfcs(
  originaldata = ex_compet,
  smtype = "compet",
  smformula = list(
    "Surv(t, d == 1) ~ x1 + x2",
    "Surv(t, d == 2) ~ x1 + x2"
  ),
  method = c("", "", "norm", "norm")
)

plot(imps)
plot(imps, include = c("x1-cause1", "x2-cause2"))

## End(Not run)
```

`smcfcs` *Substantive model compatible fully conditional specification imputation of covariates.*

Description

Multiply imputes missing covariate values using substantive model compatible fully conditional specification.

Usage

```
smcfcs(
  originaldata,
  smtype,
  smformula,
  method,
  predictorMatrix = NULL,
```

```
  m = 5,
  numit = 10,
  rjlimit = 1000,
  noisy = FALSE,
  errorProneMatrix = NULL,
  restrictions = NULL
)
```

Arguments

originaldata	The original data frame with missing values.
smtype	A string specifying the type of substantive model. Possible values are "lm", "logistic", "brlogistic", "poisson", "weibull", "coxph", "compet".
smformula	The formula of the substantive model. For "weibull" and "coxph" substantive models the left hand side should be of the form "Surv(t,d)". For "compet" substantive models, a list should be passed consisting of the Cox models for each cause of failure (see example).
method	A required vector of strings specifying for each variable either that it does not need to be imputed (""), the type of regression model to be be used to impute. Possible values are "norm" (normal linear regression), "logreg" (logistic regression), "brlogreg" (bias reduced logistic regression), "poisson" (Poisson regression), "podds" (proportional odds regression for ordered categorical variables), "mlogit" (multinomial logistic regression for unordered categorical variables), or a custom expression which defines a passively imputed variable, e.g. "x^2" or "x1*x2". "latnorm" indicates the variable is a latent normal variable which is measured with error. If this is specified for a variable, the "errorProneMatrix" argument should also be used.
predictorMatrix	An optional predictor matrix. If specified, the matrix defines which covariates will be used as predictors in the imputation models (the outcome must not be included). The i'th row of the matrix should consist of 0s and 1s, with a 1 in the j'th column indicating the j'th variable be used as a covariate when imputing the i'th variable. If not specified, when imputing a given variable, the imputation model covariates are the other covariates of the substantive model which are partially observed (but which are not passively imputed) and any fully observed covariates (if present) in the substantive model. Note that the outcome variable is implicitly conditioned on by smcfcs, and should not be specified as a predictor in the predictor matrix.
m	The number of imputed datasets to generate. The default is 5.
numit	The number of iterations to run when generating each imputation. In a (limited) range of simulations good performance was obtained with the default of 10 iterations. However, particularly when the proportion of missingness is large, more iterations may be required for convergence to stationarity.
rjlimit	Specifies the maximum number of attempts which should be made when using rejection sampling to draw from imputation models. If the limit is reached when running a warning will be issued. In this case it is probably advisable to increase the rjlimit until the warning does not appear.

`noisy` logical value (default FALSE) indicating whether output should be noisy, which can be useful for debugging or checking that models being used are as desired.

`errorProneMatrix` An optional matrix which if specified indicates that some variables are measured with classical measurement error. If the i'th variable is measured with error by variables j and k, then the (i,j) and (i,k) entries of this matrix should be 1, with the remainder of entries 0. The i'th element of the method argument should then be specified as `"latnorm"`. See `vignette("coverror", package = "smcfcs")` for more details.

`restrictions` Optional string which specifies restrictions for handling coarsened factor level covariates. This is where for a factor variable for some individuals we do not their value of the variable but we do know it belongs to some subset of the sample space. For further details on how to specify this argument, see `vignette("coarsening", package = "smcfcs")`.

Details

smcfcs imputes missing values of covariates using the Substantive Model Compatible Fully Conditional Specification multiple imputation approach proposed by Bartlett *et al* 2015 (see references).

Imputation is supported for linear regression (`"lm"`), logistic regression (`"logistic"`), bias reduced logistic regression (`"brlogistic"`), Poisson regression (`"poisson"`), Weibull (`"weibull"`) and Cox regression for time to event data (`"coxph"`), and Cox models for competing risks data (`"compet"`). For `"coxph"`, the event indicator should be integer coded with 0 for censoring and 1 for event. For `"compet"`, a Cox model is assumed for each cause specific hazard function, and the event indicator should be integer coded with 0 corresponding to censoring, 1 corresponding to failure from the first cause etc.

The function returns a list. The first element `impDataset` of the list is a list of the imputed datasets. Models (e.g. the substantive model) can be fitted to each and results combined using Rubin's rules using the mitools package, as illustrated in the examples.

The second element `smCoefIter` is a three dimensional array containing the values of the substantive model parameters obtained at the end of each iteration of the algorithm. The array is indexed by: imputation number, parameter number, iteration.

If the substantive model is linear, logistic or Poisson regression, `smcfcs` will automatically impute missing outcomes, if present, using the specified substantive model. However, even in this case, the user should specify `""` in the element of method corresponding to the outcome variable.

The bias reduced methods make use of the `brglm2` package to fit the corresponding glms using Firth's bias reduced approach. These may be particularly useful to use in case of perfect prediction, since the resulting model estimates are always guaranteed to be finite, even in the case of perfect prediction.

The development of this package was supported by the UK Medical Research Council (Fellowship MR/K02180X/1 and grant MR/T023953/1). Part of its development took place while Bartlett was kindly hosted by the University of Michigan's Department of Biostatistics & Institute for Social Research.

The structure of many of the arguments to `smcfcs` are based on those of the excellent `mice` package.

Value

A list containing:

`impDatasets` a list containing the imputed datasets

`smCoefIter` a three dimension matrix containing the substantive model parameter values. The matrix is indexed by [imputation,parameter number,iteration]

Author(s)

Jonathan Bartlett <jonathan.bartlett1@lshtm.ac.uk>

References

Bartlett JW, Seaman SR, White IR, Carpenter JR. Multiple imputation of covariates by fully conditional specification: accommodating the substantive model. Statistical Methods in Medical Research 2015; 24(4): 462-487. doi:10.1177/0962280214521348

Examples

```
#set random number seed to make results reproducible
set.seed(123)

#linear substantive model with quadratic covariate effect
imps <- smcfcs(ex_linquad, smtype="lm", smformula="y~z+x+I(x^2)",
               method=c("","","norm",""))

#if mitools is installed, fit substantive model to imputed datasets
#and combine results using Rubin's rules
if (requireNamespace("mitools", quietly = TRUE)) {
  library(mitools)
  impobj <- imputationList(imps$impDatasets)
  models <- with(impobj, lm(y~z+x+I(x^2)))
  summary(MIcombine(models))
}

#the following examples are not run when the package is compiled on CRAN
#(to keep computation time down), but they can be run by package users
## Not run:
  #examining convergence, using 100 iterations, setting m=1
  imps <- smcfcs(ex_linquad, smtype="lm", smformula="y~z+x+I(x^2)",
                 method=c("","","norm",""),m=1,numit=100)
  #convergence plot from first imputation for third coefficient of substantive model
  plot(imps$smCoefIter[1,3,])

 #include auxiliary variable assuming it is conditionally independent of Y (which it is here)
  predMatrix <- array(0, dim=c(ncol(ex_linquad),ncol(ex_linquad)))
  predMatrix[3,] <- c(0,1,0,1)
  imps <- smcfcs(ex_linquad, smtype="lm", smformula="y~z+x+I(x^2)",
                 method=c("","","norm",""),predictorMatrix=predMatrix)

  #impute missing x1 and x2, where they interact in substantive model
  imps <- smcfcs(ex_lininter, smtype="lm", smformula="y~x1+x2+x1*x2",
```

```
               method=c("","norm","logreg"))

#logistic regression substantive model, with quadratic covariate effects
imps <- smcfcs(ex_logisticquad, smtype="logistic", smformula="y~z+x+I(x^2)",
               method=c("","","norm",""))

#Poisson regression substantive model
imps <- smcfcs(ex_poisson, smtype="poisson", smformula="y~x+z",
               method=c("","norm",""))
if (requireNamespace("mitools", quietly = TRUE)) {
  library(mitools)
  impobj <- imputationList(imps$impDatasets)
  models <- with(impobj, glm(y~x+z,family=poisson))
  summary(MIcombine(models))
}

#Cox regression substantive model, with only main covariate effects
if (requireNamespace("survival", quietly = TRUE)) {
  imps <- smcfcs(ex_coxquad, smtype="coxph", smformula="Surv(t,d)~z+x+I(x^2)",
                 method=c("","","","norm",""))

  #competing risks substantive model, with only main covariate effects
  imps <- smcfcs(ex_compet, smtype="compet",
                 smformula=c("Surv(t,d==1)~x1+x2", "Surv(t,d==2)~x1+x2"),
                 method=c("","","logreg","norm"))
}

#if mitools is installed, fit model for first competing risk
if (requireNamespace("mitools", quietly = TRUE)) {
  library(mitools)
  impobj <- imputationList(imps$impDatasets)
  models <- with(impobj, coxph(Surv(t,d==1)~x1+x2))
  summary(MIcombine(models))
}

#discrete time survival analysis example
M <- 5
imps <- smcfcs(ex_dtsam, "dtsam", "Surv(failtime,d)~x1+x2",
               method=c("logreg","", "", ""),m=M)
#fit dtsam model to each dataset manually, since we need
#to expand to person-period data form first
ests <- vector(mode = "list", length = M)
vars <- vector(mode = "list", length = M)
for (i in 1:M) {
  longData <- survSplit(Surv(failtime,d)~x1+x2, data=imps$impDatasets[[i]],
                        cut=unique(ex_dtsam$failtime[ex_dtsam$d==1]))
  mod <- glm(d~-1+factor(tstart)+x1+x2, family="binomial", data=longData)
  ests[[i]] <- coef(mod)
  vars[[i]] <- diag(vcov(mod))
}
summary(MIcombine(ests,vars))
```

```
## End(Not run)
```

`smcfcs.casecohort` *Substantive model compatible fully conditional specification imputation of covariates for case cohort studies*

Description

Multiply imputes missing covariate values using substantive model compatible fully conditional specification for case cohort studies.

Usage

```
smcfcs.casecohort(originaldata, smformula, method, sampfrac, in.subco, ...)
```

Arguments

`originaldata`	The case-cohort data set (NOT a full cohort data set with a case-cohort substudy within it)
`smformula`	A formula of the form "Surv(entertime,t,d)~x", where d is the event (d=1) or censoring (d=0) indicator, t is the event or censoring time and entertime is equal to the time origin (typically 0) for individuals in the subcohort and is equal to (t-0.001) for cases outside the subcohort [this sets cases outside the subcohort to enter follow-up just before their event time. The value 0.001 may need to be modified depending on the time scale.]
`method`	A required vector of strings specifying for each variable either that it does not need to be imputed (""), the type of regression model to be be used to impute. Possible values are `"norm"` (normal linear regression), `"logreg"` (logistic regression), `"brlogreg"` (bias reduced logistic regression), `"poisson"` (Poisson regression), `"podds"` (proportional odds regression for ordered categorical variables), `"mlogit"` (multinomial logistic regression for unordered categorical variables), or a custom expression which defines a passively imputed variable, e.g. `"x^2"` or `"x1*x2"`. `"latnorm"` indicates the variable is a latent normal variable which is measured with error. If this is specified for a variable, the `"errorProneMatrix"` argument should also be used.
`sampfrac`	The proportion of individuals from the underlying full cohort who are in the subcohort
`in.subco`	The name of a column in the dataset with 0/1s that indicates whether the subject is in the subcohort
`...`	Additional arguments to pass on to smcfcs

Details

This version of `smcfcs` is designed for use with case cohort studies but where the analyst does not wish to, or cannot (due to not having the necessary data) impute the full cohort. The function's arguments are the same as for the main smcfcs function, except for `smformula`, `in.subco`, and `sampfrac` - see above for details on how these should be specified.

Author(s)

Ruth Keogh <ruth.keogh@lshtm.ac.uk>

Jonathan Bartlett <jonathan.bartlett1@lshtm.ac.uk>

Examples

```
#the following example is not run when the package is compiled on CRAN
#(to keep computation time down), but it can be run by package users
## Not run:
  #as per the documentation for ex_cc, the sampling fraction is 10%
  imps <- smcfcs.casecohort(ex_cc, smformula="Surv(entertime, t, d)~x+z", sampfrac=0.1,
                            in.subco="in.subco", method=c("", "", "norm", "", "", "", ""))
  library(mitools)
  impobj <- imputationList(imps$impDatasets)
  models <- with(impobj, coxph(Surv(entertime,t,d)~x+z+cluster(id)))
  summary(MIcombine(models))

## End(Not run)
```

`smcfcs.dtsam` *Substantive model compatible fully conditional specification imputation of covariates for discrete time survival analysis*

Description

Multiply imputes missing covariate values using substantive model compatible fully conditional specification for discrete time survival analysis.

Usage

```
smcfcs.dtsam(originaldata, smformula, method, timeEffects = "factor", ...)
```

Arguments

`originaldata`	The data in wide form (i.e. one row per subject)
`smformula`	A formula of the form "Surv(t,d)~x1+x2+x3", where t is the discrete time variable, d is the binary event indicator, and the covariates should not include time. The time variable should be an integer coded numeric variable taking values from 1 up to the final time period.
`method`	A required vector of strings specifying for each variable either that it does not need to be imputed (""), the type of regression model to be be used to impute. Possible values are `"norm"` (normal linear regression), `"logreg"` (logistic regression), `"brlogreg"` (bias reduced logistic regression), `"poisson"` (Poisson regression), `"podds"` (proportional odds regression for ordered categorical variables), `"mlogit"` (multinomial logistic regression for unordered categorical variables), or a custom expression which defines a passively imputed variable, e.g. `"x^2"` or `"x1*x2"`. `"latnorm"` indicates the variable is a latent normal

variable which is measured with error. If this is specified for a variable, the "errorProneMatrix" argument should also be used.

timeEffects — Specifies how the effect of time is modelled. timeEffects="factor" (the default) models time as a factor variable. timeEffects="linear" and timeEffects="quad" specify that time be modelled as a continuous linear or quadratic effect on the log odds scale respectively.

... — Additional arguments to pass on to smcfcs

Details

For this substantive model type, like for the other substantive model types, smcfcs expects the originaldata to have one row per subject. Variables indicating the discrete time of failure/censoring and the event indicator should be passed in smformula, as described.

The default is to model the effect of time as a factor. This will not work in datasets where there is not at least one observed event in each time period. In such cases you must specify a simpler parametric model for the effect of time. At the moment you can specify either a linear or quadratic effect of time (on the log odds scale).

Author(s)

Jonathan Bartlett <jonathan.bartlett1@lshtm.ac.uk>

Examples

```
#the following example is not run when the package is compiled on CRAN
#(to keep computation time down), but it can be run by package users
## Not run:
  #discrete time survival analysis example
  M <- 5
  imps <- smcfcs.dtsam(ex_dtsam, "Surv(failtime,d)~x1+x2",
                 method=c("logreg","", "", ""),m=M)
  #fit dtsam model to each dataset manually, since we need
  #to expand to person-period data form first
  ests <- vector(mode = "list", length = M)
  vars <- vector(mode = "list", length = M)
  for (i in 1:M) {
    longData <- survSplit(Surv(failtime,d)~x1+x2, data=imps$impDatasets[[i]],
                          cut=unique(ex_dtsam$failtime[ex_dtsam$d==1]))
    mod <- glm(d~-1+factor(tstart)+x1+x2, family="binomial", data=longData)
    ests[[i]] <- coef(mod)
    vars[[i]] <- diag(vcov(mod))
  }
  library(mitools)
  summary(MIcombine(ests,vars))

## End(Not run)
```

`smcfcs.finegray` *Substantive model compatible fully conditional specification imputation of covariates for a Fine-Gray model*

Description

Multiply imputes missing covariate values using substantive model compatible fully conditional specification for competing risks outcomes, when the substantive model is a Fine-Gray model for the subdistribution hazard of one event.

Usage

```
smcfcs.finegray(
  originaldata,
  smformula,
  method,
  cause = 1,
  m = 5,
  numit = 10,
  rjlimit = 5000,
  kmi_args = list(formula = ~1, bootstrap = FALSE, nboot = 10),
  ...
)
```

Arguments

`originaldata`	The original data frame with missing values.
`smformula`	The formula of the substantive model, given as a string. Needs to be of the form "Surv(t, d) ~ x1 + x2", where t is a vector of competing event times, and d is a (numeric) competing event indicator, where 0 must designate a censored observation.
`method`	A required vector of strings specifying for each variable either that it does not need to be imputed (""), the type of regression model to be be used to impute. Possible values are `"norm"` (normal linear regression), `"logreg"` (logistic regression), `"brlogreg"` (bias reduced logistic regression), `"poisson"` (Poisson regression), `"podds"` (proportional odds regression for ordered categorical variables), `"mlogit"` (multinomial logistic regression for unordered categorical variables), or a custom expression which defines a passively imputed variable, e.g. `"x^2"` or `"x1*x2"`. `"latnorm"` indicates the variable is a latent normal variable which is measured with error. If this is specified for a variable, the `"errorProneMatrix"` argument should also be used.
`cause`	Numeric, designating the competing event of interest (default is ‘cause = 1‘).
`m`	The number of imputed datasets to generate. The default is 5.
`numit`	The number of iterations to run when generating each imputation. In a (limited) range of simulations good performance was obtained with the default of 10 iterations. However, particularly when the proportion of missingness is large, more iterations may be required for convergence to stationarity.

`rjlimit`	Specifies the maximum number of attempts which should be made when using rejection sampling to draw from imputation models. If the limit is reached when running a warning will be issued. In this case it is probably advisable to increase the `rjlimit` until the warning does not appear.
`kmi_args`	List, containing arguments to be passed on to kmi. The "formula" element is a formula where the right-hand side specifies the covariates used for multiply imputing the potential censoring times for individual's failing from competing events. The default is 'formula = ~ 1', which uses marginal Kaplan-Meier estimator of the censoring distribution.
`...`	Additional arguments to pass on to smcfcs

Details

In the presence of random right censoring, the function first multiply imputes the potential censoring times for those failing from competing events using kmi, and thereafter uses smcfcs to impute the missing covariates. See Bonneville *et al.* 2024 for further details on the methodology.

The function does not (yet) support parallel computation.

Value

An object of type "smcfcs", as would usually be returned from smcfcs.

Author(s)

Edouard F. Bonneville <e.f.bonneville@lumc.nl>

References

Bonneville EF, Beyersmann J, Keogh RH, Bartlett JW, Morris TP, Polverelli N, de Wreede LC, Putter H. Multiple imputation of missing covariates when using the Fine–Gray model. 2024. Submitted.

Examples

```
## Not run:
library(survival)
library(kmi)

imps <- smcfcs.finegray(
  originaldata = ex_finegray,
  smformula = "Surv(times, d) ~ x1 + x2",
  method = c("", "", "logreg", "norm"),
  cause = 1,
  kmi_args = list("formula" = ~ 1)
)

if (requireNamespace("mitools", quietly = TRUE)) {
  library(mitools)
  impobj <- imputationList(imps$impDatasets)
  # Important: use Surv(newtimes, newevent) ~ ... when pooling
```

```
  # (respectively: subdistribution time and indicator for cause of interest)
  models <- with(impobj, coxph(Surv(newtimes, newevent) ~ x1 + x2))
  summary(MIcombine(models))
}

## End(Not run)
```

`smcfcs.flexsurv` *Substantive model compatible fully conditional specification imputation of covariates and event times using flexible parametric survival models*

Description

Multiply imputes missing covariate values and event times using substantive model compatible fully conditional specification with a Royston-Parmar flexible parametric survival model.

Usage

```
smcfcs.flexsurv(
  originaldata,
  smformula,
  method,
  k = 2,
  imputeTimes = FALSE,
  censtime = NULL,
  originalKnots = TRUE,
  ...
)
```

Arguments

`originaldata`	The original data frame with missing values.
`smformula`	A formula of the form "Surv(t,d)~x+z"
`method`	A required vector of strings specifying for each variable either that it does not need to be imputed (""), the type of regression model to be be used to impute. Possible values are `"norm"` (normal linear regression), `"logreg"` (logistic regression), `"brlogreg"` (bias reduced logistic regression), `"poisson"` (Poisson regression), `"podds"` (proportional odds regression for ordered categorical variables), `"mlogit"` (multinomial logistic regression for unordered categorical variables), or a custom expression which defines a passively imputed variable, e.g. `"x^2"` or `"x1*x2"`. `"latnorm"` indicates the variable is a latent normal variable which is measured with error. If this is specified for a variable, the `"errorProneMatrix"` argument should also be used.
`k`	Number of knots to use in the flexible parametric survival model

imputeTimes	If set to TRUE, smcfcs.flexsurv will impute censored survival times, as well as any missing covariates
censtime	Value(s) to use for censoring of imputed event times. If a vector, it should be of length equal to the number of original censored individuals
originalKnots	If imputing censored event times, setting originalKnots=TRUE means the automatically chosen knot locations from the model fitted to the observed times are used throughout. If FALSE, knots are chosen automatically at each iteration by flexsurvspline based on the current observed+imputed event times, according to the chosen value of k.
...	Additional arguments to pass on to smcfcs

Details

This version of smcfcs is for time-to-event outcomes which are modelled using a flexible parametric proportional hazards survival model, as proposed by Royston and Parmar (2002). The model is fitted using the flexsurvspline function in the **flexsurv** package. Specifically it fits models using the hazard scale. The flexibility of the model can be changed by modifying the k argument, which specifies the number of knots.

If desired, smcfcs.flexsurv can be used to impute event times for individuals who are originally censored, by specifying imputeTimes=TRUE. In the resulting imputed datasets every individual will have an event time and the event indicator will be one for all. Alternatively, you can impute censored times, but setting a larger potential censoring time, which is either a common value used for all or a vector of times, by using the censtime argument. If some individuals have their time-to-event outcome completely missing and you want to impute this, they should have a time of zero and the event indicator set to zero.

smcfcs.flexsurv will not let you impute using norm, latnorm or poisson methods for variables that are allowed to have time-varying effects, because the usual rejection sampling bound used by smcfcs is not valid in this setting.

flexsurvspline sometimes fails during model fitting. If/when this occurs, smcfcs.flexsurv takes a posterior draw based on the model fit from the preceding iteration, and a warning is printed at the end of the smcfcs.flexsurv run detailing how many times it occurred.

Author(s)

Jonathan Bartlett <jonathan.bartlett1@lshtm.ac.uk>

References

Royston P, Parmar MKB. Flexible parametric proportional-hazards and proportional-odds models for censored survival data, with application to prognostic modelling and estimation of treatment effects. Statistics in Medicine 2002; 21(15): 2175-2197. doi:10.1002/sim.1203

Examples

```
#the following example is not run when the package is compiled on CRAN
#(to keep computation time down), but it can be run by package users
## Not run:
```

```
set.seed(63213)
imps <- smcfcs.flexsurv(ex_flexsurv,
                        k=2,
                        smformula="Surv(t,d)~x+z",
                        method=c("","","logreg",""))
library(mitools)
impobj <- imputationList(imps$impDatasets)
models <- with(impobj, flexsurvspline(Surv(t,d)~x+z, k=2))
summary(MIcombine(models))

# now impute event times as well as missing covariates
imps <- smcfcs.flexsurv(ex_flexsurv,
                        k=2,
                        smformula="Surv(t,d)~x+z",
                        method=c("","","logreg",""),
                        imputeTimes=TRUE)

# now impute event times as well as missing covariates,
# but setting max observed event time to 2
imps <- smcfcs.flexsurv(ex_flexsurv,
                        k=2,
                        smformula="Surv(t,d)~x+z",
                        method=c("","","logreg",""),
                        imputeTimes=TRUE,
                        censtime=2)


## End(Not run)
```

`smcfcs.nestedcc` *Substantive model compatible fully conditional specification imputation of covariates for nested case control studies*

Description

Multiply imputes missing covariate values using substantive model compatible fully conditional specification for nested case control studies.

Usage

```
smcfcs.nestedcc(originaldata, smformula, method, set, event, nrisk, ...)
```

Arguments

`originaldata`	The nested case-control data set (NOT a full cohort data set with a case-cohort substudy within it)
`smformula`	A formula of the form "Surv(t,case)~x+strata(set)", where case is case-control indicator, t is the event or censoring time. Note that t could be set to the case's event time for the matched controls in a given set. The right hand side should include the case control set as a strata term (see example).

method	A required vector of strings specifying for each variable either that it does not need to be imputed (""), the type of regression model to be be used to impute. Possible values are "norm" (normal linear regression), "logreg" (logistic regression), "brlogreg" (bias reduced logistic regression), "poisson" (Poisson regression), "podds" (proportional odds regression for ordered categorical variables), "mlogit" (multinomial logistic regression for unordered categorical variables), or a custom expression which defines a passively imputed variable, e.g. "x^2" or "x1*x2". "latnorm" indicates the variable is a latent normal variable which is measured with error. If this is specified for a variable, the "errorProneMatrix" argument should also be used.
set	variable identifying matched sets in nested case-control study
event	variable which indicates who is a case/control in the nested case-control sample. Note that this is distinct from d.
nrisk	variable which is the number at risk (in the underlying full cohort) at the event time for the case in each matched set (i.e. nrisk is the same for all individuals in a matched set).
...	Additional arguments to pass on to smcfcs

Details

This version of smcfcs is designed for use with nested case control studies. The function's arguments are the same as for the main smcfcs function, except for smformula, set, event and nrisk - see above for details on how these should be specified.

Author(s)

Ruth Keogh <ruth.keogh@lshtm.ac.uk>

Jonathan Bartlett <jonathan.bartlett1@lshtm.ac.uk>

Examples

```
#the following example is not run when the package is compiled on CRAN
#(to keep computation time down), but it can be run by package users
## Not run:
  predictorMatrix <- matrix(0,nrow=dim(ex_ncc)[2],ncol=dim(ex_ncc)[2])
  predictorMatrix[which(colnames(ex_ncc)=="x"),c(which(colnames(ex_ncc)=="z"))] <- 1

  imps <- smcfcs.nestedcc(originaldata=ex_ncc,set="setno",nrisk="numrisk",event="d",
                          smformula="Surv(t,case)~x+z+strata(setno)",
                          method=c("", "", "logreg", "", "", "", "", ""),
                          predictorMatrix=predictorMatrix)
  library(mitools)
  impobj <- imputationList(imps$impDatasets)
  models <- with(impobj, clogit(case~x+z+strata(setno)))
  summary(MIcombine(models))

## End(Not run)
```

`smcfcs.parallel` *Parallel substantive model compatible imputation*

Description

Runs substantive model compatible imputation using parallel cores

Usage

```
smcfcs.parallel(
  smcfcs_func = "smcfcs",
  seed = NULL,
  m = 5,
  n_cores = parallel::detectCores() - 1,
  cl_type = "PSOCK",
  outfile = "",
  ...
)
```

Arguments

`smcfcs_func`	Specifies which base smcfcs function to call. Possible values are ‘smcfcs‘, ‘smcfcs.casecohort‘, ‘smcfcs.dtasam‘, ‘smcfcs.nestedcc‘. Defaults to ‘smcfcs‘.
`seed`	Optional seed, set as ‘set.seed‘ when ‘n_cores = 1‘, or as ‘parallel::clusterSetRNGStream‘ when ‘n_cores > 1‘.
`m`	Number of imputed datasets to generate.
`n_cores`	Number of cores over which to split the ‘m‘ imputations. If ‘n_cores‘ is not divisible exactly by ‘m‘, one of the cores will perform more/less imputations that the rest such that the final result still contains ‘m‘ imputed datasets.
`cl_type`	Either "PSOCK" or "FORK". If running on a Windows system "PSOCK" is recommended, otherwise for Linux/Mac machines "FORK" tends to offer faster computation - see parlmice.
`outfile`	Optional character path to location for output from the workers. Useful to diagnose rejection sampling warnings. File path must be formulated as "path/to/filename.txt".
`...`	Additional arguments to pass on to smcfcs, smcfcs.casecohort, smcfcs.dtsam, or smcfcs.nestedcc.

Details

This function can be used to call one of the substantive model compatible imputation methods using parallel cores, to reduce computation time. You must specify the arguments required for the standard smcfcs call, and then specify your the arguments for how to use parallel cores.

Value

An object of type "smcfcs", as would usually be returned from smcfcs.

Author(s)

Edouard F. Bonneville <e.f.bonneville@lumc.nl>

Jonathan Bartlett <jonathan.bartlett1@lshtm.ac.uk>

Examples

```
## Not run:
# Detect number of cores
parallel::detectCores()

imps <- smcfcs.parallel(
  smcfcs_func = "smcfcs",
  seed = 2021,
  n_cores = 2,
  originaldata = smcfcs::ex_compet,
  m = 10,
  smtype = "compet",
  smformula = list(
    "Surv(t, d == 1) ~ x1 + x2",
    "Surv(t, d == 2) ~ x1 + x2"
  ),
  method = c("", "", "norm", "norm")
)

## End(Not run)
```

Index