

# Package ‘smcfcs’

January 18, 2022

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

**Version** 1.6.1

**URL** <https://github.com/jwb133/smcfcfs>

**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

**Suggests** knitr, rmarkdown, mitools, ggplot2

**VignetteBuilder** knitr

**RoxygenNote** 7.1.2

**Encoding** UTF-8

**NeedsCompilation** no

**Author** Jonathan Bartlett [aut, cre],  
Ruth Keogh [aut],  
Edouard F. Bonneville [aut],  
Claus Thorn Ekstrøm [ctb]

**Maintainer** Jonathan Bartlett <j.w.bartlett@bath.ac.uk>

**Repository** CRAN

**Date/Publication** 2022-01-18 17:10:02 UTC

## R topics documented:

ex_cc . . . . .	2
ex_compet . . . . .	3
ex_coxquad . . . . .	3
ex_dtsam . . . . .	4
ex_lininter . . . . .	4
ex_linquad . . . . .	5
ex_logisticquad . . . . .	5
ex_ncc . . . . .	6
ex_poisson . . . . .	6
plot.smfcfs . . . . .	7
smfcfs . . . . .	8
smfcfs.casecohort . . . . .	12
smfcfs.dtsam . . . . .	14
smfcfs.nestedcc . . . . .	16
smfcfs.parallel . . . . .	19
<b>Index</b>	<b>21</b>

---

ex_cc	<i>Simulated case cohort data</i>
-------	-----------------------------------

---

### 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_compet	<i>Simulated example data with competing risks outcome and partially observed covariates</i>
-----------	--

---

**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	<i>Simulated example data with time to event outcome and quadratic covariate effects</i>
------------	--

---

**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 6 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)
- xsq** The square of **x**, which thus has missing values also
- 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\_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	<i>Simulated example data with continuous outcome and quadratic covariate effects</i>
------------	---

---

**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 5 variables:

**y** Continuous outcome

**z** Fully observed covariate, with linear effect on outcome

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

**xsq** The square of **x**, which thus has missing values also

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

---

ex_logisticquad	<i>Simulated example data with binary outcome and quadratic covariate effects</i>
-----------------	---

---

**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 5 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)

**xsq** The square of **x**, which thus has missing values also

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

---

ex_ncc	<i>Simulated nested case-control data</i>
--------	---

---

**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	<i>Simulated example data with count outcome, modelled using Poisson regression</i>
------------	---

---

**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	<i>Assess convergence of a smcfcs object</i>
-------------	--

---

**Description**

Visualises the contents of smCoeftter. 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 faceted 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, faceted 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	<i>Substantive model compatible fully conditional specification imputation of covariates.</i>
--------	---

---

### 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
)

```

### Arguments

originaldata	The original data frame with missing values.
smtype	A string specifying the type of substantive model. Possible values are "lm", "logistic", "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 used to impute. Possible values are "norm" (normal linear regression), "logreg" (logistic regression), "poisson" (Poisson regression), "pods" (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 the rejection sampling scheme used 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 the measurement error vignette for more details.

## 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"), 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 development of this package was supported by a UK Medical Research Council Fellowship (MR/K02180X/1). Part of its development took place while the author 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 `mi` 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 <j.w.bartlett@bath.ac.uk> <https://thestatsgeek.com> <http://www.missingdata.org.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](https://doi.org/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+xsq",
              method=c("","norm","x^2",""))

#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+xsq))
  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+xsq",
              method=c("", "", "norm", "x^2", ""), 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(0,1,0,0,1)
imps <- smcfcs(ex_linquad, smtype="lm", smformula="y~z+x+xsq",
              method=c("", "", "norm", "x^2", ""), 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+xsq",
              method=c("", "", "norm", "x^2", ""))

#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+xsq",
                method=c("", "", "", "norm", "x^2", ""))

  #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	<i>Substantive model compatible fully conditional imputation of covariates for case cohort studies</i>
-------------------	--

---

## Description

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

## Usage

```

smcfcs.casecohort(
  originaldata,
  smformula,
  sampfrac,
  in.subco,
  method,
  predictorMatrix = NULL,
  m = 5,
  numit = 10,
  rjlimit = 1000,
  noisy = FALSE,
  errorProneMatrix = NULL
)

```

## 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.]
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
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 used to impute. Possible values are "norm" (normal linear regression), "logreg" (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 the rejection sampling scheme used 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 the measurement error vignette for more details.

**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 <j.w.bartlett@bath.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,
  timeEffects = "factor",
  method,
  predictorMatrix = NULL,
  m = 5,
  numit = 10,
  rjlimit = 1000,
  noisy = FALSE,
  errorProneMatrix = NULL
)
```

**Arguments**

<code>originaldata</code>	The data in wide form (i.e. one row per subject)
<code>smformula</code>	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.
<code>timeEffects</code>	Specifies how the effect of time is modelled. <code>timeEffects="factor"</code> (the default) models time as a factor variable. <code>timeEffects="linear"</code> and <code>timeEffects="quad"</code> specify that time be modelled as a continuous linear or quadratic effect on the log odds scale respectively.
<code>method</code>	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 used to impute. Possible values are "norm" (normal linear regression), "logreg" (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.
<code>predictorMatrix</code>	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 the rejection sampling scheme used by smcfcs, and should not be specified as a predictor in the predictor matrix.
<code>m</code>	The number of imputed datasets to generate. The default is 5.
<code>numit</code>	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.
<code>rjlimit</code>	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 <code>rjlimit</code> until the warning does not appear.
<code>noisy</code>	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.
<code>errorProneMatrix</code>	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 the measurement error vignette for more details.

## Details

For this substantive model type, like for the other substantive model types, smcfcs expects the original data 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 <j.w.bartlett@bath.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.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,
  set,
  event,
  nrisk,
  method,
  predictorMatrix = NULL,
  m = 5,
  numit = 10,
  rjlimit = 1000,
  noisy = FALSE,
  errorProneMatrix = NULL
)

```

**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).
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).
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 used to impute. Possible values are "norm" (normal linear regression), "logreg" (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 the rejection sampling scheme used by smcfcs, and should not be specified as a predictor in the predictor matrix.
<code>m</code>	The number of imputed datasets to generate. The default is 5.
<code>numit</code>	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.
<code>rjlimit</code>	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 <code>rjlimit</code> until the warning does not appear.
<code>noisy</code>	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.
<code>errorProneMatrix</code>	An optional matrix which if specified indicates that some variables are measured with classical measurement error. If the <i>i</i> 'th variable is measured with error by variables <i>j</i> and <i>k</i> , then the ( <i>i,j</i> ) and ( <i>i,k</i> ) entries of this matrix should be 1, with the remainder of entries 0. The <i>i</i> 'th element of the method argument should then be specified as "latnorm". See the measurement error vignette for more details.

## 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 <j.w.bartlett@bath.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,
  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.dtsam’, ‘smcfcs.nestedcc’. Defaults to ‘smcfcs’.
seed	A required 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 than 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 <a href="#">parlmice</a> .
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 <a href="#">smcfcs</a> , <a href="#">smcfcs.casecohort</a> , <a href="#">smcfcs.dtsam</a> , or <a href="#">smcfcs.nestedcc</a> .

## 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 <j.w.bartlett@bath.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

## \* datasets

- ex\_cc, [2](#)
- ex\_compet, [3](#)
- ex\_coxquad, [3](#)
- ex\_dtsam, [4](#)
- ex\_lininter, [4](#)
- ex\_linquad, [5](#)
- ex\_logisticquad, [5](#)
- ex\_ncc, [6](#)
- ex\_poisson, [6](#)

- ex\_cc, [2](#)
- ex\_compet, [3](#)
- ex\_coxquad, [3](#)
- ex\_dtsam, [4](#)
- ex\_lininter, [4](#)
- ex\_linquad, [5](#)
- ex\_logisticquad, [5](#)
- ex\_ncc, [6](#)
- ex\_poisson, [6](#)

- parlmice, [19](#)
- plot.smcfcs, [7](#)

- smcfcs, [8](#), [19](#), [20](#)
- smcfcs.casecohort, [12](#), [19](#)
- smcfcs.dtsam, [14](#), [19](#)
- smcfcs.nestedcc, [16](#), [19](#)
- smcfcs.parallel, [19](#)