Package ‘CoImp’

August 8, 2016

Title Copula Based Imputation Method
Date 2016-07-31
Version 0.3-1
Author Francesca Marta Lilja Di Lascio, Simone Giannerini
Depends R (= 3.1.0), methods, copula
Imports nnet, gtools, locfit
Description Copula based imputation method. A semiparametric imputation procedure for missing multivariate data based on conditional copula specifications.
Maintainer Francesca Marta Lilja Di Lascio <marta.dilascio@unibz.it>
License GPL (>= 2)
NeedsCompilation no
Repository CRAN
Date/Publication 2016-08-08 16:51:32

R topics documented:

CoImp ................................................................. 2
CoImp-class ......................................................... 5
MAR ................................................................. 8
MAR-class ......................................................... 9
MCAR ............................................................... 10
MCAR-class ....................................................... 12
PerfMeasure ..................................................... 13
PerfMeasure-class ............................................... 15
Index ......................................................... 17
Description

Imputation method based on conditional copula functions.

Usage

CoImp(X, n.marg = ncol(X), x.up = NULL, x.lo = NULL, q.up = NULL,
q.lo = NULL, type.data = "continuous", smoothing = rep(0.5, n.marg),
plot = TRUE, model = list(normalCopula(0.5, dim=n.marg, dispstr="ex"),
claytonCopula(10, dim=n.marg), gumbelCopula(10, dim=n.marg),
frankCopula(10, dim=n.marg)), ...)

Arguments

x
a data matrix with missing values. Missing values should be denoted with NA.
n.marg
the number of variables in X.

x.up
a vector of length n.marg with the upper value of each margin used in the Hit or Miss method.

x.lo
a vector of length n.marg with the lower value of each margin used in the Hit or Miss method.

q.up
a vector of length n.marg with the probability of the quantile function used to define x.up for each margin.

q.lo
a vector of length n.marg with the probability of the quantile function used to define x.lo for each margin.

type.data
the nature of the variables in X: discrete or continuous.

smoothing
values for the nearest neighbour component of the smoothing parameter of the lp function.

plot
logical: if TRUE plots the estimated marginal densities and a bar plot of the percentages of missing and available data for each margin.

model
a list of copula models to be used for the imputation, see the Details section. This should be one of normal, frank, clayton and gumbel.

... further parameters for fitCopula, lp and further graphical arguments.

Details

CoImp is an imputation method based on conditional copula functions that allows to impute missing observations according to the multivariate dependence structure of the generating process without any assumptions on the margins. This method can be used independently from the dimension and the kind (monotone or non monotone) of the missing patterns.

Brief description of the approach:
1. estimate both the margins and the copula model on available data by means of the semi-parametric sequential two-step inference for margins;
2. derive conditional density functions of the missing variables given non-missing ones through the corresponding conditional copulas obtained by using the Bayes’ rule;
3. impute missing values by drawing observations from the conditional density functions derived at the previous step. The Monte Carlo method used is the Hit or Miss.

The estimation approach for the copula fit is semiparametric: a range of nonparametric margins and parametric copula models can be selected by the user.

Value

An object of S4 class "CoImp", which is a list with the following elements:

- **Missing.data.matrix**: the original missing data matrix to be imputed.
- **Perc.miss**: the matrix of the percentage of missing and available data.
- **Estimated.Model**: the estimated copula model on the available data.
- **Estimation.Method**: the estimation method used for the copula **Estimated.Model**.
- **Index.matrix.NA**: matrix indices of the missing data.
- **Smooth.param**: the smoothing parameter alpha selected on the basis of the AIC.
- **Imputed.data.matrix**: the imputed data matrix.
- **Estimated.Model.Imp**: the estimated copula model on the imputed data matrix.
- **Estimation.Method.Imp**: the estimation method used for the copula **Estimated.Model.Imp**.

Author(s)

Francesca Marta Lilja Di Lascio <marta.dilascio@unibz.it>,
Simone Giannerini <simone.giannerini@unibo.it>

References


Examples

```r
## generate data from a 4-variate Frank copula with different margins
set.seed(11)
n.marg <- 4
theta <- 5
copula <- copula::frankCopula(theta, dim = n.marg)
mymvdc <- mvdc(copula, c("norm", "gamma", "beta", "gamma"), list(list(mean=7, sd=2),
               list(shape=3, rate=2), list(shape1=4, shape2=1), list(shape=4, rate=3)))
n <- 20
x.samp <- copula::rmvdc(n, mymvdc)

# randomly introduce univariate and multivariate missing
perc.mis <- 0.5
set.seed(11)
miss.row <- sample(1:n, perc.mis*n, replace=TRUE)
miss.col <- sample(1:n.marg, perc.mis*n, replace=TRUE)
miss <- cbind(miss.row, miss.col)
x.samp.miss <- replace(x.samp, miss, NA)

# impute missing values
imp <- coimp(x.samp.miss, n.marg=n.marg, smoothing = rep(0.6, n.marg), plot=TRUE, type.data="continuous");

# methods show and plot
show(imp)
plot(imp)

## generate data from a 3-variate Clayton copula with different bounded margins
set.seed(11)
n.marg <- 3
theta <- 5
copula <- copula::claytonCopula(theta, dim = n.marg)
mymvdc <- mvdc(copula, c("beta", "beta", "beta"), list(list(shape=4, shape2=1),
               list(shape1=.5, shape2=.5), list(shape1=2, shape2=3)))
n <- 100
x.samp <- copula::rmvdc(n, mymvdc)

# randomly introduce univariate and multivariate missing
perc.mis <- 0.2
```

CoImp
set.seed(11)
miss.row <- sample(1:n, perc.miss*n, replace=TRUE)
miss.col <- sample(1:n.marg, perc.miss*n, replace=TRUE)
miss <- cbind(miss.row, miss.col)
x.samp.miss <- replace(x.samp, miss, NA)

# impute missing values
imp <- CoImp(x.samp.miss, n.marg=n.marg, smoothing = c(0.45,0.2,0.5), plot=TRUE,
             q.lo=rep(0.1,n.marg), q.up=rep(0.9,n.marg));

# methods show and plot
show(imp)
plot(imp)

---

**CoImp-class**

Class "CoImp"

**Description**

A class for CoImp and its extensions

**Objects from the Class**

Objects can be created by calls of the form new("CoImp", ...).

**Slots**

- **Missing.data.matrix**: Object of class "matrix". Original missing data matrix to be imputed.
- **Perc.miss**: Object of class "matrix". Missing and available data percentage for each variable.
- **Estimated.Model**: Object of class "list". The list contains:
  - **model**: the copula model selected and estimated on the complete cases.
  - **dimension**: the dimension of the model.
  - **parameter**: the estimated dependence parameter of the model.
  - **number**: the index of the estimated model in the list of models given in input.
- **Estimation.Method**: Object of class "character". The estimation method used for the copula model in Estimated.Model. Allowed methods are in fitCopula.
- **Index.matrix.NA**: Object of class "matrix". Matrix of row and column indexes of missing data.
- **Smooth.param**: Object of class "numeric". The values of the nearest neighbor component of the smoothing parameter of the lp function.

**Imputed.data.matrix** Object of class "matrix". The imputed data matrix.
Estimated.Model.Imp Object of class "list". The list contains:
model: the copula model selected and estimated on the imputed cases.
dimension: the dimension of the model.
parameter: the estimated dependence parameter of the model.
number: the index of the estimated model in the list of models given in input.


Methods

plot signature(x = "CoImp", y = "missing"): ...
show signature(object = "CoImp"): ...

Author(s)
Francesca Marta Lilja Di Lascio <marta.dilascio@unibz.it>,
Simone Giannerini <simone.giannerini@unibo.it>

References


See Also

See Also CoImp, lp and copula.

Examples

showClass("CoImp")
**MAR**

*Generation of multivariate missing at random (MAR) data*

---

**Description**

Introduction of artificial missing at random (MAR) data in a given data set. Missing values are multivariate and have generic pattern.

**Usage**

```
MAR(db.complete, perc.miss = 0.3, setseed = 13, ...)
```

**Arguments**

- `db.complete` the complete data matrix.
- `perc.miss` the percentage of missing value to be generated.
- `setseed` the seed for the generation of the missing values.
- `...` further parameters for `fitCopula`.

**Details**

MAR introduce artificial missing at random values in a given complete data set. Missing values are univariate and multivariate and have generic pattern.

**Value**

An object of S4 class "MAR", which is a list with the following element:

- `perc.record.missing`
  Object of class "numeric". A percentage value.
- `db.missing`
  Object of class "matrix". A data set with artificial multivariate MAR.

**Author(s)**

Francesca Marta Lilja Di Lascio <marta.dilascio@unibz.it>,
Simone Giannerini <simone.giannerini@unibo.it>

**References**


Examples

```r
# generate data from a 4-variate Gumbel copula with different margins
set.seed(11)
n.marg <- 4
theta <- 5
copula <- frankCopula(theta, dim = n.marg)
mmvdc <- mvdc(copula, c("norm", "gamma", "beta","gamma"), list(list(mean=7, sd=2),
list(shape=3, rate=2), list(shape=4, shape2=1), list(shape=4, rate=3)))
n <- 50
x.samp <- rmvdc(n, mmvdc)

# apply MAR by introducing 30% of missing data
mar <- MAR(db.complete = x.samp, perc.miss = 0.3, setseed = 11)
mar
```

MAR-class

### Class "MAR"

**Description**

A class for MAR and its extensions

**Objects from the Class**

Objects can be created by calls of the form `new("MAR", ...)`.  

**Slots**

- `perc.record.missing`: Object of class "numeric". A percentage value.  
- `db.missing`: Object of class "matrix". A data set with artificial multivariate MAR with generic pattern.
Methods

show signature(object = "MAR"): ...

Author(s)

Francesca Marta Lilja Di Lascio <marta.dilascio@unibz.it>,
Simone Giannerini <simone.giannerini@unibo.it>

References


See Also

See Also CoImp, lp and copula.

Examples

showClass("MAR")

---

**MCAR**

*Generation of multivariate MCAR data*

Description

Introduction of artificial missing completely at random (MCAR) data in a given data set. Missing values are multivariate and have generic pattern.

Usage

MCAR(db.complete, perc.miss = 0.3, setseed = 13, ...)

---
Arguments

- `db.complete`: the complete data matrix.
- `perc.miss`: the percentage of missing value to be generated.
- `setseed`: the seed for the generation of the missing values.
- `...`: further parameters for `fitCopula`.

Details

MCAR introduce artificial missing completely at random values in a given complete data set. Missing values are multivariate and have generic pattern.

Value

An object of S4 class "MCAR", which is a list with the following element:

- `db.missing`: Object of class "matrix". A data set with artificial multivariate MCAR.

Author(s)

Francesca Marta Lilja Di Lascio <marta.dilascio@unibz.it>, Simone Giannerini <simone.giannerini@unibo.it>

References


Examples

```r
# generate data from a 4-variate Gumbel copula with different margins
set.seed(11)
n.marg <- 4
theta <- 5
copula <- frankCopula(theta, dim = n.marg)
mymvdc <- mvdc(copula, c("norm", "gamma", "beta","gamma"), list(list(mean=7, sd=2),
```

...
list(shape=3, rate=2), list(shape=1, shape2=1), list(shape=4, rate=3))
n <- 50
x.samp <- rMvdc(n, mymvdc)

# apply MCAR by introducing 30% of missing data
mcar <- MCAR(db.complete = x.samp, perc.miss = 0.3, setseed = 11)
mcar

_____ MCAR-class ______

Class "MCAR"

Description

A class for MCAR and its extensions

Objects from the Class

Objects can be created by calls of the form new("MCAR", ...).

Slots

db.missing: Object of class "matrix". A data set with artificial multivariate MCAR.

Methods

show signature(object = "MCAR"): ...

Author(s)

Francesca Marta Lilja Di Lascio <marta.dilascio@unibz.it>,
Simone Giannerini <simone.giannerini@unibo.it>

References

PerfMeasure


See Also

See Also CoImp, lp and copula.

Examples

showClass("MCAR")

---

PerfMeasure

*Performance measures for evaluating the goodness of an imputed database*

Description

Set of measures useful to evaluate the goodness of the used imputation method.

Usage

PerfMeasure(db.complete, db.imputed, db.missing, n.marg = 2, model = list(normalCopula(0.5, dim=n.marg, dispstr="ex"), claytonCopula(10, dim=n.marg), gumbelCopula(10, dim=n.marg), frankCopula(10, dim=n.marg)), ...)

Arguments

db.complete the complete data matrix.
db.imputed the imputed data matrix.
db.missing the data matrix with NA data.
n.marg the number of variables in db.complete.
model a list of copula models to be used for the imputation. See the Details section. This should be one of normal, frank, clayton and gumbel.
... further parameters for fitCopula.

Details

PerfMeasure computes some measures useful for evaluating the goodness of the used imputation method. PerfMeasure requires in input the imputed, the complete and the missing data matrix and gives in output five different measures of performance. See below for details.
Value
An object of S4 class "PerfMeasure", which is a list with the following elements:

- **MARE** Object of class "numeric". The mean (on the replications performed) of the absolute relative error between the imputed and the corresponding original value.

- **RB** Object of class "numeric". The relative bias of the estimator for the dependence parameter.

- **RRMSE** Object of class "numeric". The relative root mean squared error of the estimator for the dependence parameter.

- **TID** Object of class "vector". Upper and lower tail dependence indexes for bivariate copulas. Original function is in `tailIndex`.

Author(s)
Francesca Marta Lilja Di Lascio <marta.dilascio@unibz.it>,
Simone Giannerini <simone.giannerini@unibo.it>

References


Examples

```r
# generate data from a 4-variate Gumbel copula with different margins

set.seed(11)
n.marg <- 4
theta <- 5
copula <- frankCopula(theta, dim = n.marg)
mymvdc <- mvdc(copula, c("norm", "gamma", "beta","gamma"), list(list(mean=7, sd=2),
               list(shape=3, rate=2), list(shape=4, shape2=1), list(shape=4, rate=3)))
n <- 20
x.samp <- rMvdc(n, mymvdc)
```
PerfMeasure-class

# randomly introduce univariate and multivariate missing
percenmis <- 0.5
set.seed(11)
miss.row <- sample(1:n, percenmis*n, replace=TRUE)
miss.col <- sample(1:n.marg, percenmis*n, replace=TRUE)
miss <- cbind(miss.row, miss.col)
ox.samp.miss <- replace(x.samp, miss, NA)

# impute missing values
imp <- CoImp(x.samp.miss, n.marg=n.marg, smoothing=rep(0.6, n.marg), plot=TRUE,
              type.data="continuous");
imp

# apply PerfMeasure to the imputed data set
pm <- PerfMeasure(db.complete=x.samp, db.missing=x.samp.miss,
                   db.imputed=imp@"Imputed.data.matrix", n.marg=4)

pm
str(pm)

---

Class "PerfMeasure"

Description

A class for PerfMeasure and its extensions

Objects from the Class

Objects can be created by calls of the form new("PerfMeasure", ...).

Slots

MARE: Object of class "numeric". The mean (on the replications performed) of the absolute relative error between the imputed and the corresponding original value.
RB: Object of class "numeric". The relative bias of the estimator for the dependence parameter.
RRMSE: Object of class "numeric". The relative root mean squared error of the estimator for the dependence parameter.
TID: Object of class "vector". Upper and lower tail dependence indexes for bivariate copulas.

Original function is in tailIndex.

Methods

show signature(object = "PerfMeasure"): ...
Author(s)
Francesca Marta Lilja Di Lascio <marta.dilascio@unibz.it>,
Simone Giannerini <simone.giannerini@unibo.it>

References

See Also
See Also CoImp, lp and copula.

Examples
showClass("PerfMeasure")
Index

*Topic **classes**
  CoImp-class, 5
  MAR-class, 9
  MCAR-class, 12
  PerfMeasure-class, 15

*Topic **copula**
  CoImp, 2
  MAR, 8
  MCAR, 10
  PerfMeasure, 13

*Topic **imputation**
  CoImp, 2
  MAR, 8
  MCAR, 10
  PerfMeasure, 13

*Topic **multivariate**
  CoImp, 2
  MAR, 8
  MCAR, 10
  PerfMeasure, 13

CoImp, 2, 7, 10, 13, 16
CoImp-class, 5
copula, 7, 10, 13, 16

fitCopula, 2, 5, 7, 8, 11, 13

lp, 2, 7, 10, 13, 16

MAR, 8
MAR-class, 9
MCAR, 10
MCAR-class, 12

PerfMeasure, 13
PerfMeasure-class, 15
plot, CoImp-missing-method
  (CoImp-class), 5

show, CoImp-method (CoImp-class), 5

tailIndex, 14, 15