Package ‘FastImputation’

March 7, 2017

Type Package

Title Learn from Training Data then Quickly Fill in Missing Data

Version 2.0

Date 2017-03-06

Author Stephen R. Haptonstahl

Maintainer Stephen R. Haptonstahl <srh@haptonstahl.org>

Suggests caret

Description TrainFastImputation() uses training data to describe a multivariate normal distribution that the data approximates or can be transformed into approximating and stores this information as an object of class 'FastImputationPatterns'. FastImputation() function uses this 'FastImputationPatterns' object to impute (make a good guess at) missing data in a single line or a whole data frame of data. This approximates the process used by Amelia [http://gking.harvard.edu/amelia/] but is much faster when filling in values for a single line of data.

License GPL (>= 2)

Collate 'FastImputation.R' 'TrainFastImputation.R' 'UnfactorColumns.R' 'BoundNormalizedVariable.R' 'NormalizeBoundedVariable.R' 'LimitToSet.R' 'CovarianceWithMissing.R'

RoxygenNote 6.0.1

Imports methods

NeedsCompilation no

Depends R (>= 2.10)

R topics documented:

BoundNormalizedVariable *Take a normalized variable and transform it back to a bounded variable.*

Description

This takes variables on the real line and constrains them to be on a half-line (constrained above or below) or a segment (constrained both above and below). This is approximately the inverse of `NormalizeBoundedVariable`; this does not completely reverse the effect of `NormalizeBoundedVariable` because `NormalizeBoundedVariable` first forces values away from the bounds, and this information is lost.

Usage

```
BoundNormalizedVariable(x, constraints)
```

Arguments

`x`	A vector, matrix, array, or dataframe with value to be coerced into a range or set.
`constraints`	A list of constraints. See the examples below for formatting details.

Value

An object of the same class as x with the values transformed into the desired half-line or segment.

Author(s)

Stephen R. Haptonstahl <`srh@haptonstahl.org`>

Examples

```
constraints=list(lower=5)           # lower bound when constrining to an interval
constraints=list(upper=10)          # upper bound when constraining to an interval
constraints=list(lower=5, upper=10) # both lower and upper bounds
```

CovarianceWithMissing *Estimate covariance when data is missing*

Description

Ignoring missing values can lead to biased estimates of the covariance. Lounici (2012) gives an unbiased estimator when the data has missing values.

Usage

```
CovarianceWithMissing(x)
```

Arguments

`x`	matrix or data.frame, data with each row an observation and each column a variable.

Value

matrix, unbiased estimate of the covariance.

Author(s)

Stephen R. Haptonstahl <`srh@haptonstahl.org`>

References

High-dimensional covariance matrix estimation with missing observations. Karim Lounici. 2012.

`FastImputation` *Use the pattern learned from the training data to impute (fill in good guesses for) missing values.*

Description

Like Amelia, FastImputation assumes that the columns of the data are multivariate normal or can be transformed into approximately multivariate normal.

Usage

```
FastImputation(x, patterns, verbose = TRUE)
```

Arguments

`x`	Vector, matrix, dataframe, or object that can be coerced into a dataframe, possibly with some missing (`NA`) values.
`patterns`	An object of class 'FastImputationPatterns' generated by `TrainFastImputation`.
`verbose`	If TRUE then the progress in imputing the data will be shown.

Value

x, but with missing values filled in (imputed)

Author(s)

Stephen R. Haptonstahl <`srh@haptonstahl.org`>

References

`http://gking.harvard.edu/amelia/`

See Also

`TrainFastImputation`

Examples

```
data(FI_train)   # provides FItrain dataset
patterns <- TrainFastImputation(
  FI_train,
  constraints=list(list(2, list(lower=0)),          # continuous var with lower bound
                  list(5, list(upper=0)),          # continuous var with only upper bound
                   list(6, list(lower=0, upper=1))   # bounded to a finite interval
                   ),
  idvars=1,  # user ids; also used for any variable not to be imputed
  categorical=9)

data(FI_test)
FI_test          # note there is missing data
## Not run: imputed.data <- FastImputation(FI_test, patterns)
## Not run: imputed.data    # good guesses for missing values are filled in

data(FI_true)
## Not run: imputation.rmse <- sqrt(sum( (imputed.data - FI_true)^2 )/sum(is.na(FI_test)))
## Not run: imputation.rmse

## Not run: library("caret")
## Not run: confusionMatrix(data=imputed.data$V9, reference=FI_true$V9)
```

FI_test *Fraud Imputation Test Data*

Description

Observations of Web financial transactions with some cells missing. Used with FastImputation.

Usage

```
FI_test
```

Format

A data frame with 10 variables and 10000 observations.

1. `cust.id`: Internal customer identification number
2. `order.id`: Unique identification number for this transaction (row)
3. `is.fraud`: 1 if the transaction is fraudulent, 0 otherwise
4. `customer.age.yrs`: Customer age in years; may be a decimal
5. `spent.days.0to2`: Amount spent in dollars by customer between 0 and 2 days before the current transaction
6. `spent.days.3to10`: Amount spent in dollars by customer between 3 and 10 days before the current transaction
7. `spent.days.11to30`: Amount spent in dollars by customer between 11 and 30 days before the current transaction
8. `geo.ip.fraud.rate`: Fraction between 0 and 1 of transactions from that geographic location (identified by IP address) that have been fraudulent

9. `account.age.days`: Integer number of days the customer has had the account
10. `days.to.first.purchase` Integer number of days between account creation and the first purchase by the customer

Author(s)

Stephen R. Haptonstahl `<srh@haptonstahl.org>`

Source

This is simulated data generated to be similar to real data.

`FI_train` *Fraud Training Data*

Description

Complete observations of Web financial transactions. Used with TrainFastImputation to prepare for imputing individual transactions as they come in.

Usage

```
FI_train
```

Format

A data frame with 10 variables and 10000 observations.

1. `cust.id`: Internal customer identification number
2. `order.id`: Unique identification number for this transaction (row)
3. `is.fraud`: 1 if the transaction is fraudulent, 0 otherwise
4. `customer.age.yrs`: Customer age in years; may be a decimal
5. `spent.days.0to2`: Amount spent in dollars by customer between 0 and 2 days before the current transaction
6. `spent.days.3to10`: Amount spent in dollars by customer between 3 and 10 days before the current transaction
7. `spent.days.11to30`: Amount spent in dollars by customer between 11 and 30 days before the current transaction
8. `geo.ip.fraud.rate`: Fraction between 0 and 1 of transactions from that geographic location (identified by IP address) that have been fraudulent
9. `account.age.days`: Integer number of days the customer has had the account
10. `days.to.first.purchase` Integer number of days between account creation and the first purchase by the customer

Author(s)

Stephen R. Haptonstahl `<srh@haptonstahl.org>`

Source

This is simulated data generated to be similar to real data.

`FI_true` *Fraud "True" Data*

Description

Complete observations of Web financial transactions. Used to gauge the accuracy of imputation of FItest.

Usage

```
FI_true
```

Format

A data frame with 10 variables and 10000 observations.

1. `cust.id`: Internal customer identification number
2. `order.id`: Unique identification number for this transaction (row)
3. `is.fraud`: 1 if the transaction is fraudulent, 0 otherwise
4. `customer.age.yrs`: Customer age in years; may be a decimal
5. `spent.days.0to2`: Amount spent in dollars by customer between 0 and 2 days before the current transaction
6. `spent.days.3to10`: Amount spent in dollars by customer between 3 and 10 days before the current transaction
7. `spent.days.11to30`: Amount spent in dollars by customer between 11 and 30 days before the current transaction
8. `geo.ip.fraud.rate`: Fraction between 0 and 1 of transactions from that geographic location (identified by IP address) that have been fraudulent
9. `account.age.days`: Integer number of days the customer has had the account
10. `days.to.first.purchase` Integer number of days between account creation and the first purchase by the customer

Author(s)

Stephen R. Haptonstahl <`srh@haptonstahl.org`>

Source

This is simulated data generated to be similar to real data.

`LimitToSet` *Coerce numeric values into a given set.*

Description

Given some values `x` and a set of values `set`, each value in `x` is changed to the value in `set` that is closest.

Usage

```
LimitToSet(x, set)
```

Arguments

`x`	A vector, matrix, array, or dataframe with value to be coerced into a range or set.
`set`	A list of values that x will be forced to take on.

Value

An object of the same class as `x` with values replaced as needed to satisfy the constraints.

Author(s)

Stephen R. Haptonstahl <`srh@haptonstahl.org`>

Examples

```
x <- runif(100, min=0, max=10)
y <- LimitToSet(x, set=c(1:10))
plot(x, y)
```

`NormalizeBoundedVariable` *Take a variable bounded above/below/both and return an unbounded (normalized) variable.*

Description

This transforms bounded variables so that they are not bounded. First variables are coerced away from the boundaries. by a distance of `tol`. The natural log is used for variables bounded either above or below but not both. The inverse of the standard normal cumulative distribution function (the quantile function) is used for variables bounded above and below.

Usage

```
NormalizeBoundedVariable(x, constraints, tol = stats::pnorm(-5),
  trim = TRUE)
```

Arguments

`x` A vector, matrix, array, or dataframe with value to be coerced into a range or set.

`constraints` A list of constraints. See the examples below for formatting details.

`tol` Variables will be forced to be at least this far away from the boundaries.

`trim` If TRUE values in x < lower and values in x > upper will be set to lower and upper, respectively, before normalizing.

Value

An object of the same class as `x` with the values transformed so that they spread out over any part of the real line.

A variable `x` that is bounded below by `lower` is transformed to `log(x - lower)`.

A variable `x` that is bounded above by `upper` is transformed to `log(upper - x)`.

A variable `x` that is bounded below by `lower` and above by `upper` is transformed to `qnorm((x-lower)/(upper - lower)`

Author(s)

Stephen R. Haptonstahl <srh@haptonstahl.org>

Examples

```
constraints=list(lower=5)           # lower bound when constrining to an interval
constraints=list(upper=10)          # upper bound when constraining to an interval
constraints=list(lower=5, upper=10) # both lower and upper bounds
```

`TrainFastImputation` *Learn from the training data so that later you can fill in missing data*

Description

Like Amelia, FastImputation assumes that the columns of the data are multivariate normal or can be transformed into approximately multivariate normal.

Usage

```
TrainFastImputation(x, constraints = list(), idvars, categorical)
```

Arguments

`x` Dataframe containing training data. Can have incomplete rows.

`constraints` A list of constraints. See the examples below for formatting details.

`idvars` A vector of column numbers or column names to be ignored in the imputation process.

`categorical` A vector of column numbers or column names of varaibles with a (small) set of possible values.

Value

An object of class 'FastImputationPatterns' that contains information needed later to impute on a single row.

Author(s)

Stephen R. Haptonstahl <srh@haptonstahl.org>

References

http://gking.harvard.edu/amelia/

See Also

FastImputation

Examples

```
data(FI_train)   # provides FI_train dataset

patterns_with_constraints <- TrainFastImputation(
  FI_train,
  constraints=list(list(2, list(lower=0)),           # continuous var with lower bound
                 list(5, list(upper=0)),          # continuous var with only upper bound
                   list(6, list(lower=0, upper=1))   # bounded to a finite interval
                   ),
  idvars=1,  # user ids; also used for any variable not to be imputed
  categorical=9)
```

UnfactorColumns *Convert columns of a dataframe from factors to character or numeric.*

Description

Convert columns of a dataframe from factors to character or numeric.

Usage

```
UnfactorColumns(x)
```

Arguments

`x`	A dataframe

Value

A dataframe containing the same data but any `factor` columns have been replaced with numeric or character columns.

Author(s)

Stephen R. Haptonstahl <srh@haptonstahl.org>

Index