Package ‘FastImputation’

October 12, 2022

Type Package

Title Learn from Training Data then Quickly Fill in Missing Data

Version 2.1

Date 2022-02-08

Author Stephen R. Haptonstahl

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

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' <https://gking.harvard.edu/amelia> but is much faster when filling in values for a single line of data.

License GPL (>= 2)

Depends R (>= 4.0)


RoxygenNote 7.1.2

Imports methods, Matrix

Suggests testthat, caret, e1071

NeedsCompilation no

Encoding UTF-8

Repository CRAN

Date/Publication 2022-02-09 07:40:15 UTC
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 constraining 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**

```r
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**


---

**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**

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

**Arguments**

- **x**
  
  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

https://gking.harvard.edu/amelia

See Also

TrainFastImputation

Examples

data(FI_train)  # provides FItrain dataset
patterns <- TrainFastImputation(
  FI_train,
  constraints=list(list("bounded_below_2", list(lower=0)),
                  list("bounded_above_5", list(upper=0)),
                  list("bounded_above_and_below_6", list(lower=0, upper=1))
    ),
  idvars="user_id_1",
  categorical="categorical_9"
)

data(FI_test)
FI_test      # note there is missing data
imputed_data <- FastImputation(FI_test, patterns)
imputed_data  # good guesses for missing values are filled in

data(FI_true)
continuous_cells_imputed <- is.na(FI_test[,2:8])
continuous_imputed_values <- imputed_data[,2:8][continuous_cells_imputed]
continuous_true_values <- FI_true[,2:8][continuous_cells_imputed]
rmse <- sqrt(median((continuous_imputed_values-continuous_true_values)^2))
median_relative_error <- median( abs((continuous_imputed_values - continuous_true_values) /
                              continuous_true_values) )

imputed_data_column_means <- FI_test[,2:8]
for(j in 1:ncol(imputed_data_column_means)) {
  imputed_data_column_means[is.na(imputed_data_column_means[,j]),j] <-
  mean(imputed_data_column_means[,j], na.rm=TRUE)
}
cont_imputed_vals_col_means <- imputed_data_column_means[continuous_cells_imputed]
rmse_column_means <- sqrt(median((cont_imputed_vals_col_means-continuous_true_values)^2))
rmse_column_means  # much larger error than using FastImputation
median_relative_error_col_means <- median( abs((cont_imputed_vals_col_means -
                               continuous_true_values) / continuous_true_values) )
median_relative_error_col_means  # larger error than using FastImputation

# Let's look at the accuracy of the imputation of the categorical variable
# using library("caret")
categorical_rows_imputed <- which(is.na(FI_test$categorical_9))
confusionMatrix(data=imputed_data$categorical_9[categorical_rows_imputed],
                reference=FI_true$categorical_9[categorical_rows_imputed])
# Compare to imputing with the modal value
stat_mode <- function(x) {
    unique_values <- unique(x)
    unique_values <- unique_values[!is.na(unique_values)]
    unique_values[which.max(tabulate(match(x, unique_values)))]
}
categorical_rows_imputed_col_mode <- rep(stat_mode(FI_test$categorical_9),
                                         length(categorical_rows_imputed))
confusionMatrix(data=categorical_rows_imputed_col_mode,
                reference=FI_true$categorical_9[categorical_rows_imputed])
# less accurate than using FastImputation

---

**FI_test**  

**Imputation Test Data**

**Description**

Smaller simulated dataset drawn from the same distribution as FI_train and FI_true. This dataset is entirely the same as FI_true except this one has 5% of its values missing. Used with FastImputation.

**Usage**

data(FI_test)

**Format**

A data frame with 9 variables and 250 observations.

- **user_id_1**  Sequential user ids
- **bounded_below_2**  Multivariate normal, transformed using \( \exp(x) \)
- **unbounded_3**  Multivariate normal
- **unbounded_4**  Multivariate normal
- **bounded_above_5**  Multivariate normal, transformed using \(-\exp(x)\)
- **bounded_above_and_below_6**  Multivariate normal, transformed using \( \text{pnorm}(x) \)
- **unbounded_7**  Multivariate normal
- **unbounded_8**  Multivariate normal
- **categorical_9**  "A" if the first of three multivariate normal draws is greatest; "B" if the second is greatest; "C" if the third is greatest
Author(s)
Stephen R. Haptonstahl <srh@haptonstahl.org>

Source
All columns start as multivariate normal draws. Columns 2, 5, and 6 are transformed. Column 9 is the result of three multivariate normal columns being interpreted as one-hot encoding of a three-valued categorical variable.

---

FI_train
Imputation Training Data

Description
Larger simulated dataset drawn from the same distribution as FI_test and FI_true and used to train the imputation algorithm. 5% of the values are missing. Used with TrainFastImputation.

Usage
data(FI_train)

Format
A data frame with 9 variables and 10000 observations.

  user_id_1  Sequential user ids
  bounded_below_2  Multivariate normal, transformed using exp(x)
  unbounded_3  Multivariate normal
  unbounded_4  Multivariate normal
  bounded_above_5  Multivariate normal, transformed using -exp(x)
  bounded_above_and_below_6  Multivariate normal, transformed using pnorm(x)
  unbounded_7  Multivariate normal
  unbounded_8  Multivariate normal
  categorical_9  "A" if the first of three multivariate normal draws is greatest; "B" if the second is greatest; "C" if the third is greatest

Author(s)
Stephen R. Haptonstahl <srh@haptonstahl.org>

Source
All columns start as multivariate normal draws. Columns 2, 5, and 6 are transformed. Column 9 is the result of three multivariate normal columns being interpreted as one-hot encoding of a three-valued categorical variable.
Description

Smaller simulated dataset drawn from the same distribution as FI_train and FI_test. This dataset is entirely the same as FI_test except FI_test has 5% of its values missing. Used to evaluate the quality of the values imputed in FI_test.

Usage

data(FI_true)

Format

A data frame with 9 variables and 250 observations.

user_id_1  Sequential user ids  
bounded_below_2  Multivariate normal, transformed using exp(x)  
unbounded_3  Multivariate normal  
unbounded_4  Multivariate normal  
bounded_above_5  Multivariate normal, transformed using -exp(x)  
bounded_above_and_below_6  Multivariate normal, transformed using pnorm(x)  
unbounded_7  Multivariate normal  
unbounded_8  Multivariate normal  
categorical_9  "A" if the first of three multivariate normal draws is greatest; "B" if the second is greatest; "C" if the third is greatest

Author(s)

Stephen R. Haptonstahl <srh@haptonstahl.org>

Source

All columns start as multivariate normal draws. Columns 2, 5, and 6 are transformed. Column 9 is the result of three multivariate normal columns being interpreted as one-hot encoding of a three-valued categorical variable.
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 \( \log(x - \text{lower}) \).

A variable x that is bounded above by upper is transformed to \( \log(\text{upper} - x) \).

A variable x that is bounded below by lower and above by upper is transformed to \( \text{qnorm}((x - \text{lower})/(\text{upper} - \text{lower})) \).

Author(s)

Stephen R. Haptonstahl <srh@haptonstahl.org>

Examples

constraints=list(lower=5)  # lower bound when constraining 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 variables 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

https://gking.harvard.edu/amelia

See Also

FastImputation

Examples

data(FI_train)  # provides FI_train dataset

patterns_with_constraints <- TrainFastImputation(
  FI_train,
  constraints=list(list("bounded_below_2", list(lower=0)),
                  list("bounded_above_5", list(upper=0)),
                  list("bounded_above_and_below_6", list(lower=0, upper=1)))
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

* datasets
  FI_test, 5
  FI_train, 6
  FI_true, 7

BoundNormalizedVariable, 2

CovarianceWithMissing, 3

FastImputation, 3, 9
FI_test, 5
FI_train, 6
FI_true, 7

NormalizeBoundedVariable, 8

TrainFastImputation, 4, 9

UnfactorColumns, 10