Package 'faoswsImputation'

April 6, 2015

Type Package

Title Package to perform ensemble imputation for various FAO domains.

Version 5.0.1

Date 2015-01-29

Author Joshua M. Browning <joshua.browning@fao.org>, Michael C. J. Kao <michael.kao@fao.org>

Maintainer Joshua M. Browning <joshua.browning@fao.org>

Description This package provides all the functions to perform imputation of variables for the FAO production domain.

URL https://svn.fao.org/projects/SWS/RModules/faoswsImputation/

License GPL (>= 3)

Imports lme4 (>= 1.1-7), reshape2 (>= 1.4), earth (>= 3.2-7), forecast (>= 5.5), zoo (>= 1.7-11), RColorBrewer (>= 1.0-5)

Depends faoswsFlag (>= 0.1.0), faoswsUtil (>= 0.1.3), data.table (>= 1.9.2), splines (>= 3.0.2), ggplot2

LazyData yes

ZipData no

VignetteBuilder knitr

Suggests knitr

R topics documented:

`addNoDataModels` *Add Ensemble Weights for Sets of the Data without Valid Observations*

Description

In some cases, countries (or whatever the byKey is) will not have any data available. In these cases, we can impute values for these observations via an ensemble, but we'll need a method for weighting the models in the ensemble.

Usage

```
addNoDataModels(data, weights, imputationParameters)
```

Arguments

data	The data.table object containing the data which will be imputed.
weights	A data.table containing the currently estimated ensemble weights and errors. These are used to compute the global errors and add new weights for models with missing data.
imputationParameters	A list of the parameters for the imputation algorithms. See defaultImputationParameters() for a starting point.

Details

This function computes the weights for these models by looking at the cross-validation error of the global models (or "commodity" level) and uses those errors to determine the final weights.

Value

A data.table of the same format as weights. This function is really just a helper function for getEnsembleWeight, and so the output of the function is specifically designed to be rbinded on to the weights object in that function.

---

allDefaultModels *Returns all default models*

Description

This is a convenience function that returns a list of all the default models.

Usage

```
allDefaultModels()
```

Value

A list of all the default model functions.

---

checkEnsembleModel *Check Ensemble Model*

Description

Check Ensemble Model

Usage

```
checkEnsembleModel(object)
```

Arguments

object	An S4 object.

Value

TRUE if there are no errors, otherwise the error messages.

---

computeEnsemble *Function to combine the ensembles*

Description

Function to combine the ensembles

Usage

```
computeEnsemble(fits, weights)
```

Arguments

fits	A list of fitted values from models.
weights	A weight matrix giving the weights of each model for each observation.

Value

A vector of length nrow(weights), where each value represents the ensemble estimate for that observation.

---

computeEnsembleFit *Function to compute the fits of all the component models*

Description

Function to compute the fits of all the component models

Usage

```
computeEnsembleFit(data, imputationParameters)
```

Arguments

data	The data.table object containing the data.
imputationParameters	A list of the parameters for the imputation algorithms. See defaultImputationParameters() for a starting point.

Value

Returns a list of vectors of the same length as ensembleModels. The ith element of the list represents the fit of the ith model to data.

computeEnsembleWeight *Function to compute the weights of the ensemble models*

Description

Function to compute the weights of the ensemble models

Usage

```
computeEnsembleWeight(data, cvGroup, fits, method = "inverse",
  imputationParameters)
```

Arguments

data	A data.table containing the data.
cvGroup	A vector of the same length as nrow(data). Entries of the vector should be integers from 1 to the number of cross-validation groups (typically 10). This should be randomly assigned, and is usually created by ensembleImpute.
fits	The fitted values from the models.
method	Must be either "inverse" or "stacking". If "inverse", the final ensemble is a weighted average of all the individual models, where the weight of each model is proportional to 1/error from that model. If "stacking", then the weight is assigned via a linear regression (where the independent variable in the regression is the variable being imputed, and each individual model is a dependent variable). The linear regression is restricted, however: no weights may be negative and the weights must sum to one.
imputationParameters	A list of the parameters for the imputation algorithms. See defaultImputationParameters() for a starting point.

computeErrorLOOCV *LOOCV Error Rate*

Description

This function computes the error of different model fits via leave-one-out cross-validation. However, typically this function will be called via computeErrorRate and not directly.

Usage

```
computeErrorLOOCV(data, model, cvGroup, imputationParameters)
```

Arguments

data	A data.table containing the data.
model	The model fit to x, should be of class ensembleModel.
cvGroup	A vector of the same length as nrow(data). Entries of the vector should be integers from 1 to the number of cross-validation groups (typically 10). This should be randomly assigned, and is usually created by ensembleImpute.
imputationParameters	A list of the parameters for the imputation algorithms. See defaultImputationParameters() for a starting point.

---

computeErrorRate *Function to compute the error of different model fits*

Description

Function to compute the error of different model fits

Usage

```
computeErrorRate(data, fit, imputationParameters)
```

Arguments

data	A data.table containing the data.
fit	The fitted value from the model.
imputationParameters	A list of the parameters for the imputation algorithms. See defaultImputationParameters() for a starting point.

Value

A vector of (the absolute value of the) errors corresponding to the non-missing entries in data's "value" column.

---

computeLoocvFits *Function to compute the fitted values from Leave One Out Cross Validation*

Description

Function to compute the fitted values from Leave One Out Cross Validation

Usage

```
computeLoocvFits(data, cvGroup, imputationParameters)
```

Arguments

data	A data.table containing the data.
cvGroup	A vector of the same length as nrow(data). Entries of the vector should be integers from 1 to the number of cross-validation groups (typically 10). This should be randomly assigned, and is usually created by ensembleImpute.
imputationParameters	A list of the parameters for the imputation algorithms. See defaultImputationParameters() for a starting point.

Value

A list of the same length as ensembleModels which contains the fitted values corresponding to the loocv procedure. The ith element of the list corresponds to the fitting of the ith element of ensembleModels.

---

defaultArima *The default ARIMA model for the ensemble model.*

Description

The default ARIMA model for the ensemble model.

Usage

```
defaultArima(x)
```

Arguments

x	A numeric vector to be imputed.

---

defaultExp *The default exponential model for the ensemble model.*

Description

The default exponential model for the ensemble model.

Usage

```
defaultExp(x)
```

Arguments

x	A numeric vector to be imputed.

defaultGlobalMean *Global Mean Model for Imputation*

Description

This function imputes missing values through a global mean.

Usage

```
defaultGlobalMean(data, imputationParameters)
```

Arguments

data	The data.table object containing the data.
imputationParameters	A list of the parameters for the imputation algorithms. See defaultImputationParameters() for a starting point.

Value

Returns a vector of the estimated/imputed values. If a value existed in the original data, then an NA is returned in that location. Otherwise, the mean of all available observations is returned (computed across the entire dataset).

defaultGlobalMedian *Global Median Model for Imputation*

Description

This function imputes missing values through a global median.

Usage

```
defaultGlobalMedian(data, imputationParameters)
```

Arguments

data	The data.table object containing the data.
imputationParameters	A list of the parameters for the imputation algorithms. See defaultImputationParameters() for a starting point.

Value

Returns a vector of the estimated/imputed values. If a value existed in the original data, then an NA is returned in that location. Otherwise, the mean of all available observations is returned (computed across the entire dataset).

`defaultImputationParameters` *Default Imputation Parameters*

Description

This function can be used to generate the input parameters for the ensemble imputation code. This is a good way to get a list of the required parameters and then modify parameters to match your particular configuration.

Usage

```
defaultImputationParameters(variable)
```

Arguments

`variable`	Should be one of "production", "yield", or "seed". These are currently the three variables for which this imputation package has been used. You may also set this to NULL and then manually assign values to the variable, imputationValueColumn, imputationFlagColumn, and imputationMethodColumn elements of this list.

Details

Below is a description of the parameters:

- yearValue: The column name for the year variable in data.
- byKey: The column name for the variable representing the splitting group. Usually, this is the country variable.
- ensembleModels: A list of objects, all of type ensembleModel, which will be applied to the data.
- restrictWeights: Should the maximum weight of one model in the ensemble be restricted?
- maximumWeights: If restrictWeights == TRUE, then this value (between 0.5 and 1) gives the largest value of a weight for a particular ensemble.
- plotImputation: Should the results of the imputation be plotted?
- errorType: Should "raw" errors be used or "loocv" (leave-one-out cross-validation)? In general, "loocv" should be preferred, but "raw" is faster.
- errorFunction: A custom error function may be specified. The default is mean-squared error. This should be a function of a single vector numeric argument, and the return value should be a numeric vector of length 1.
- groupCount: How many cross-validation groups should be used for the ensemble models?
- missingFlag: How are missing values specified in the database? Usually, this is "M".
- imputationFlag: What observation flag should be assigned to imputed values?
- newMethodFlag: What method flag should be assigned to imputed values?
- flagTable: A table of the observation flags and their corresponding weights.
- variable: The name of the variable being imputed, either "seed", "yield", or "production".
- imputationValueColumn: The column name of the value to be imputed.

- imputationFlagColumn: The column name of the observation flag for the imputed variable.
- imputationMethodColumn: The column name of the method flag for the imputed variable.
- newImputationColumn: The column name of a new column to append to the dataset which will store the imputed values. The data will be contained in a column with the Value_ prefix appended to this name, and the flags will prepend "flagObservationStatus_" and "flagMethod_". If this parameter is "", the original data columns are overwritten with the imputed data.
- estimateNoData: This logical value indicates if imputation should be performed for countries with no available observations. For example, a hierarchical regression model may be fit to the data via defaultMixedModel (with an updated formula) or something similar. Then, this model could be used to estimate for countries with no available data provided that data is available for some higher hierarchy. The default, though, is FALSE: one must set this option if it is desired to be used.

Value

Returns a list of the default parameters used in the ensemble imputation algorithms.

`defaultLm` *The default linear regression model for the ensemble model.*

Description

The default linear regression model for the ensemble model.

Usage

```
defaultLm(x)
```

Arguments

`x`	A numeric vector to be imputed.

`defaultLoess` *The default LOESS model for the ensemble model.*

Description

The default LOESS model for the ensemble model.

Usage

```
defaultLoess(x)
```

Arguments

`x`	A numeric vector to be imputed.

defaultLogistic *Default Logistic Model for the Ensemble.*

Description

The model fit is $Production = A/(1 + exp(-B(time - C)))$, and fitting is done via non-linear least squares (see ?nls). If this fit fails to converge, than A is fixed to the maximum Production value and the model is fit via glm().

Usage

```
defaultLogistic(x)
```

Arguments

x	A numeric vector to be imputed.

Value

A numeric vector with the estimated logistic regression model.

Note

If the midpoint of the logistic regression model is outside the range of the data, then a vector of NA's is returned (to prevent poor extrapolation).

defaultMars *The default MARS model for the ensemble model.*

Description

The default MARS model for the ensemble model.

Usage

```
defaultMars(x)
```

Arguments

x	A numeric vector to be imputed.

defaultMean *The default mean model for the ensemble model.*

Description

The default mean model for the ensemble model.

Usage

```
defaultMean(x)
```

Arguments

x	A numeric vector to be imputed.

defaultMedian *The default median model for the ensemble model.*

Description

The default median model for the ensemble model.

Usage

```
defaultMedian(x)
```

Arguments

x	A numeric vector to be imputed.

defaultMixedModel *Mixed Model for Imputation*

Description

This function imputes missing values with a linear mixed model.

Usage

```
defaultMixedModel(data, df = 1, weights = NULL, modelFormula = NULL,
  imputationParameters)
```

Arguments

data	The data.table object containing the data.
df	The number of degrees of freedom for the spline.
weights	The weights for the observations.
modelFormula	Formula specifying how the dependent variable (value) depends on the other columns of data. Should be a valid mixed model formula, as it will be passed to lmer (R's mixed model function).
imputationParameters	A list of the parameters for the imputation algorithms. See defaultImputationParameters() for a starting point.

Details

The default functionality of this model is to fit a linear mixed model to the data. time and intercept are assumed to be fixed effects, and the random effects are specified by the byKey parameter of imputationParameters. So, for example, byKey may reference the variable containing country data. In that case, a model is fit which assumes a linear relationship between production (or whatever dependent variable the user has specified) and time. However, the intercept and slope of this fit varies from country to country, and so country is considered a random effect.

Moreover, the model fit is not a simple linear regression, but rather a spline regression (using the bs function from the **splines** package). The fit of this model will therefore depend on the number of degrees of freedom of this spline model (and, if the degrees of freedom is 1, then the simple linear regression model is used).

Value

Returns a vector of the estimated/imputed values. If a value existed in the original data, then an NA is returned in that location.

---

defaultMixedModelOptimized

Optimized Mixed Model for Imputation

Description

This function imputes missing values with a linear mixed model.

Usage

```
defaultMixedModelOptimized(data, maxdf = 5, weights = NULL, modelFormula,
  imputationParameters)
```

Arguments

data	The data.table object containing the data.
maxdf	The maximum degrees of freedom for the spline.
weights	The weights for the observations.

`modelFormula` Formula specifying how the dependent variable (value) depends on the other columns of data. Should be a valid mixed model formula, as it will be passed to lmer (R's mixed model function).

`imputationParameters`
A list of the parameters for the imputation algorithms. See defaultImputationParameters() for a starting point.

Details

The default functionality of this model is to fit a linear mixed model to the data. time and intercept are assumed to be fixed effects, and the random effects are specified by the byKey parameter of imputationParameters. So, for example, byKey may reference the variable containing country data. In that case, a model is fit which assumes a linear relationship between production (or whatever dependent variable the user has specified) and time. However, the intercept and slope of this fit varies from country to country, and so country is considered a random effect.

Moreover, the model fit is not a simple linear regression, but rather a spline regression (using the bs function from the **splines** package). The fit of this model will therefore depend on the number of degrees of freedom of this spline model (and, if the degrees of freedom is 1, then the simple linear regression model is used).

Value

Returns a vector of the estimated/imputed values. If a value existed in the original data, then an NA is returned in that location.

`defaultNaive` *The default naive model for the ensemble model.*

Description

The naive model is simply linear interpolation with last observation carried forward and backward.

Usage

```
defaultNaive(x)
```

Arguments

`x` A numeric vector to be imputed.

defaultSpline *The default spline model for the ensemble model.*

Description

The default spline model for the ensemble model.

Usage

```
defaultSpline(x)
```

Arguments

x	A numeric vector to be imputed.

ensembleImpute *Function to perform ensemble imputation*

Description

This is an implementation of the ensemble imputation methodology developed for the FAO production domain.

Usage

```
ensembleImpute(data, imputationParameters)
```

Arguments

data	A data.table containing the data.
imputationParameters	A list of the parameters for the imputation algorithms. See defaultImputationParameters() for a starting point.

ensembleModel-class *Ensemble Model Class*

Description

`model`: The function defining how the model is fit to the data. The function should take one or two arguments: data and imputationParameters (see extendSimpleModel) if level == "country" but only data if level == "countryCommodity".

`extrapolationRange`: How many time steps outside of the data is this model valid for? Should be a positive integer (or Inf). Defaults to 0.

`level`: The level at which this model is applied. Currently, must be one of "countryCommodity" or "commodity". This defines if this model operates on each country-commodity pair individually ("countryCommodity" level) or all countries for a fixed commodity ("commodity" level) at once. Defaults to "countryCommodity".

ensureImputationInputs *Ensure Imputation Inputs*

Description

Ensure Imputation Inputs

Usage

```
ensureImputationInputs(data, imputationParameters)
```

Arguments

data	The data.frame containing the data to be imputed.
imputationParameters	A list of the imputation parameters, such as one generated by defaultImputationParameters().

Value

No value is returned, but errors are thrown if the input parameters are not as expected.

extendSimpleModel *Extend Simple Model*

Description

This function takes any model which is designed to run on a simple time series (i.e. at the commodity-country level) and applies it to each country in turn, thus making it a commodity level model.

Usage

```
extendSimpleModel(data, model, imputationParameters)
```

Arguments

data	The data.table object containing the data.
model	The model to be applied to each individual time series. Typically, this will be a function such as one from allDefaultModels().
imputationParameters	A list of the parameters for the imputation algorithms. See defaultImputationParameters() for a starting point.

Value

Returns a vector of the estimated/imputed values. If a value existed in the original data, then an NA is returned in that location.

getDefaultRange *Default range for ensemble models*

Description

This function returns the default extrapolation range values for each model in the input list of models.

Usage

```
getDefaultRange(ensembleModel)
```

Arguments

ensembleModel	A list of ensemble models to return ranges for.

Value

A numeric vector of the default ranges for each of the ensemble models.

getInverseWeights *Get Ensemble Weights via Inverse Errors*

Description

Weights used to construct the final ensemble from the individual models are computed by comparing the errors. For each model, the errors are computed (simply comparing observed values in data to fitted values in fits) and then the inverse (i.e. 1/) these errors are used as the weights (after rescaling to 1). For example, if three models have errors of 1, 2, and 3 then the inverse errors are 1, 1/2, and 1/3. We rescale these weights to add to 1 and get 6/11, 3/11, and 2/11.

Usage

```
getInverseWeights(data, fits, imputationParameters)
```

Arguments

data	The data object containing the observations to impute.
fits	A list of the fitted values. These may be estimated via leave one out cross-validation or directly.
imputationParameters	A list of the parameters for the imputation algorithms. See defaultImputationParameters() for a starting point.

Value

A data.table containing the weight for each model within each byKey group, as well as a few other (currently unused) statistics.

getObservedExtrapolationRange *Get Observed Extrapolation Range*

Description

This function takes a vector of data (with posisbly missing values) and returns a vector of the same length. If the corresponding value of x is not missing or is an interpolation, the returned value is 0. If the corresponding value of x is missing and is an extrapolation, then the returned value is the "distance" (in terms of number of observations) to the closest non-missing value. See examples.

Usage

```
getObservedExtrapolationRange(x)
```

Arguments

x	The vector of data.

Value

A vector of the same length as x.

Examples

```
x = c(NA, NA, NA, 1:7, NA, NA)
getObservedExtrapolationRange(x)
```

getStackingWeights *Get Ensemble Weights via Stacking*

Description

Weights used to construct the final ensemble from the individual models are computed via stacking.

Usage

```
getStackingWeights(data, fits, imputationParameters)
```

Arguments

data	The data object containing the observations to impute.
fits	A list of the fitted values. These may be estimated via leave one out cross-validation or directly.
imputationParameters	A list of the parameters for the imputation algorithms. See defaultImputationParameters() for a starting point.

Details

Traditional stacking proceeds as follows: For each individual byKey (usually country) a set of weights is chosen for the ensemble. The weights are constrained to be positive and to sum to 1, and the final ensemble is constructed by sum(w_i*model_i). The weights should be chosen in a way such that better models are given more importance. Thus, the following criteria is minimized:

sum(errorFunction(|y - w_i*model_i|))

i.e. the errorFunction applied to the difference between the observed values and the ensemble estimate. The errorFunction is typically just x^2, but could be a more complex function.

However, this is roughly equivalent to regression, with a constraint added. In some cases, however, our datasets will be so sparse that we won't be able to perform this optimization (only four observations and 7 valid models, for example, will not have a unique solution). Thus, we instead use a LASSO regression for computing the stacking weights.

Note: if errorType is not "loocv", then stacking will be problematic: much higher weights will be given to flexible models (such as loess or splines) without valid reason.

Value

A data.table containing the weight for each model within each byKey group, as well as a few other (currently unused) statistics.

---

getWeightMatrix *Get Weight Matrix*

Description

Initially, a weight is computed for each model and byKey. However, some models are not valid for some observations (as certain models are limited in how far they can extrapolate outside the range of the data). Thus, the final weight for each ensemble model at each observation will depend on that models performance for that byKey group as well as if that model is valid at that point.

Usage

```
getWeightMatrix(data, w, imputationParameters)
```

Arguments

`data`	The data.table containing the data.
`w`	The weights data.table, typically as produced in computeEnsembleWeight. There should be three columns: byKey, model, and weight. Weight gives the model weight for model within the byKey group, and exactly one row should exist for each byKey/model pair.
`imputationParameters`	A list of the parameters for the imputation algorithms. See defaultImputationParameters() for a starting point.

Details

This function creates a weight matrix to use in constructing the final ensemble. If F is a nxk matrix (n = number of observations, k = number of models) containing the fitted models, then this function constructs W, another nxk matrix of weights. The final ensemble estimate for observation i can be computed by sum(F[i,]*W[i,]).

Value

A matrix of weights that can be multiplied by the fitted models to give the imputed values. Rows corresponding to non-missing values in data have values of NA.

---

imputeVariable *Function to impute production or yield*

Description

This is a wrapper of the ensemble imputation for the production domain.

Usage

```
imputeVariable(data, imputationParameters)
```

Arguments

data	The data.table object containing the data.
imputationParameters	A list of the parameters for the imputation algorithms. See defaultImputationParameters() for a starting point.

---

logisticGlm *Helper function for defaultLogistic*

Description

This function fits a logistic model to the data via logistic regression (see ?glm and the binomial family). To ensure all x values are between 0 and 1, the values are all divided by the maximum x value. Thus, this model assumes the asymptote is the largest observed x value. Time is the independent variable and is the vector 1:length(x).

Usage

```
logisticGlm(x)
```

Arguments

x	The dependent variable.

Value

A numeric vector of the same length as x but with the model estimates at each point in time. Note: if the logistic model has it's midpoint outside the range of the data, this function will return a vector of NA's (as the original defaultLogistic function had this behavior to prevent poor fitting).

See Also

Other logistic.functions: logisticNlsIntercept; logisticNls

logisticNls *Helper function for defaultLogistic*

Description

This function fits a logistic model to the data via non-linear least squares (?nls). The function fit is: x = B / (1 + exp(-C*(time-D))) where time is the independent variable (1 to length(x)) and x is the dependent variable.

Usage

```
logisticNls(x)
```

Arguments

x	The dependent variable.

Value

A numeric vector of the same length as x but with the model estimates at each point in time. Note: if the logistic model has it's midpoint outside the range of the data, this function will return a vector of NA's (as the original defaultLogistic function had this behavior to prevent poor fitting).

See Also

Other logistic.functions: logisticGlm; logisticNlsIntercept

logisticNlsIntercept *Helper function for defaultLogistic*

Description

This function fits a logistic model to the data via non-linear least squares (?nls). The function fit is: x = A + B / (1 + exp(-C*(time-D))) where time is the independent variable (1 to length(x)) and x is the dependent variable.

Usage

```
logisticNlsIntercept(x)
```

Arguments

x	The dependent variable.

Value

A numeric vector of the same length as x but with the model estimates at each point in time. Note: if the logistic model has it's midpoint outside the range of the data, this function will return a vector of NA's (as the original defaultLogistic function had this behavior to prevent poor fitting).

See Also

Other logistic.functions: logisticGlm; logisticNls

`makeCvGroup` *Make Cross-Validation Groups*

Description

Creates a vector of cross-validation groups to be used for leave-one-out cross-validation in later models.

Usage

```
makeCvGroup(data, imputationParameters)
```

Arguments

`data`	A data.table object containing the data.
`imputationParameters`	A list of the parameters for the imputation algorithms. See defaultImputationParameters() for a starting point.

Value

A numeric vector taking values in 1:groupCount, or NA if the corresponding observation is missing. All groups are represented within each individual timeseries (defined by byKey) when possible.

`mixedModelFixedDf` *Mixed Model for Imputation with Fixed DF*

Description

This function imputes missing values through a linear mixed model. It differs from defaultMixedModel in that it only fits a model with one value for the degree of freedom (defaultMixedModel currently fits multiple models and uses bootstrapping to pick the optimal one). However, if models are eventually combined into an ensemble using leave-one-out cross-validation, such bootstrapping shouldn't be necessary (but it may be reasonable to include several mixed models with varying degrees of freedom).

Usage

```
mixedModelFixedDf(data, df = 1, weights = NULL, modelFormula,
  imputationParameters)
```

Arguments

data	The data.table object containing the data.
df	The degrees of freedom for the spline.
weights	The weights for the observation, if NULL each observation has the same weight.
modelFormula	Formula specifying how the dependent variable (value) depends on the other columns of data. Should be a valid mixed model formula, as it will be passed to lmer (R's mixed model function). If missing, a spline on the year will be used.
imputationParameters	A list of the parameters for the imputation algorithms. See defaultImputationParameters() for a starting point.

Details

Note: this function will return the same result as defaultMixedModel if a modelFormula is specified (to both functions).

Value

Returns a vector of the estimated/imputed values. If a value existed in the original data, then an NA is returned in that location.

---

okrapd *Example data for the documentations.*

Description

The data containing global okra production, area harvested and yield from the year 1995 to 2013.

Usage

```
data(okrapd)
```

Format

A data.table object with 912 rows and 14 variables

---

plotEnsemble *Plot Ensemble and Model Fits*

Description

This function plots each of the individual models in the ensemble as well as the initial data and the final ensemble. This function is not meant to be called directly by the user but instead is a helper function called by ensembleImpute (if the argument plot = TRUE).

Usage

```
plotEnsemble(data, modelFits, modelWeights, ensemble, imputationParameters,
  returnFormat = "faceted")
```

Arguments

data	The data.table containing the data being imputed.
modelFits	A list of length equal to the number of models. Each element of the list should be a numeric vector of length nrow(data). This object is usually created by computeEnsembleFit.
modelWeights	A matrix of dimension nrow(data) x length(modelFits). Each element corresponds to the weight for model/column j and for observation/row i. Note: modelFits and modelWeights need not have the same ordering, but they should have the same names as this is how they are matched.
ensemble	A numeric vector containing either the original data (if it was not missing) or the imputed value (if the original data was missing).
imputationParameters	A list of the parameters for the imputation algorithms. See defaultImputationParameters() for a starting point.
returnFormat	In what format should the plots be returned? If "faceted", a single plot showing with all countries is returned. If "individual", a list with ggplot objects is returned. If "prompt", the first country's plot is displayed and then the user must press ENTER to cycle through the plots.

Value

If returnType = "faceted" or "prompt", then no value is returned (but a plot is generated). However, if returnType = prompt, then a list of ggplot objects is returned.

---

plotEnsembleOld *Plot Ensemble and Model Fits (Old)*

Description

This function plots each of the individual models in the ensemble as well as the initial data and the final ensemble. This function is not meant to be called directly by the user but instead is a helper function called by ensembleImpute (if the argument plotImputation = TRUE). This function differs from plotEnsemble in that it uses the original plotting code (base graphics instead of ggplot2).

Usage

```
plotEnsembleOld(data, modelFits, modelWeights, ensemble)
```

Arguments

data	The data.table containing the data being imputed.
modelFits	A list of length equal to the number of models. Each element of the list should be a numeric vector of length nrow(data). This object is usually created by computeEnsembleFit.
modelWeights	A matrix of dimension nrow(data) x length(modelFits). Each element corresponds to the weight for model/column j and for observation/row i. Note: modelFits and modelWeights need not have the same ordering, but they should have the same names as this is how they are matched.

ensemble	A numeric vector containing either the original data (if it was not missing) or the imputed value (if the original data was missing).
value	The column name of data containing the variable being imputed.
byKey	The column name of data containing the grouping variable.
yearValue	The column name of data containing the time variable.

Value

No value is returned, but a plot is generated.

---

reassignGlobalVariable *Reassign Global Variable*

Description

Parameters used in running the imputation or processing algorithms are assigned to the global environment. To protect these parameters from being accidentally overwritten, we call lockBinding. This prevents reassignment of these variables, but occassionally these variables do need to be reassigned. This function allows this reassignment by temporarily unlocking the variable, reassigning the value, and relocking it. This function should be used with caution: values that have already been ensured can be changed and not checked, which could be disastrous.

Usage

```
reassignGlobalVariable(var, value)
```

Arguments

var	The name of the variable to be reassigned.
value	The value to be reassigned to var.

Value

No value is returned, instead this function changes the global variable var.

---

sampleEqually *Helper function for makeCvGroup*

Description

This function is designed to ensure that creation of cross-validation groups are well-balanced. Suppose there are k groups and n observations. If n is a multiple of k, each group is represented exactly n/k times. Otherwise, each group is represented either floor(n/k) or floor(n/k)+1 times.

Usage

```
sampleEqually(n, k)
```

Arguments

n	The total number of values to be sampled.
k	The values to be sampled from (1:k).

Value

A vector of length n that contains values in 1:k.

---

weightMatrixToVector *Convert the weights matrix to a vector*

Description

Initially, a weight is computed for each model. Then, the weight matrix is created, and this contains those weights if they are valid at the current location (i.e. if that particular model is allowed to extrapolate to the current point). It may be useful to convert the weight matrix back to a vector (in particular, for plotting the weights of each base learner).

Usage

```
weightMatrixToVector(weightMatrix)
```

Arguments

weightMatrix	The weights matrix, as returned by computeEnsembleWeights.

Details

This function works by finding which row of the weights matrix has the fewest number of 0's. This row will correspond to an interpolation or non-missing row, and thus all models will be valid at this point. Since all models are valid here, the weights at this point will correspond to the original weight vector, and so we just return that row.

Value

A vector of weights that is the original weight vector prior to being extended to a matrix.

---

weightVectorToMatrix *Convert the weights vector to a matrix*

Description

The default weights vector assumes that all models are valid for all imputation points. However, some imputations may be extrapolations that are well outside the range of the data, and in this case not all models will be valid. Thus, this function takes the weights vector and adjusts it for such observations.

Usage

```
weightVectorToMatrix(x, w, modelExtrapolationRange)
```

Arguments

x	A numeric vector to be imputed.
w	The vector of weights, computed based on the errors of the models.
modelExtrapolationRange	A numeric vector specifying the valid range of extrapolation for each model. This vector must be the same length as w, as it's i-th element gives the extrapolation range for the i-th weight. Note: care should be taken to ensure the weight vector's i-th element and the modelExtrapolationRange vector's i-th element both correspond to the same model in the ensemble.

Details

Note that the extrapolation range was not a parameter in previous versions of the code. Thus, if that element is not present, the model assumes the range is infinite. This ensures backwards-compatibility.

Value

A matrix of weights of dimension Txm, where T=number of time steps and m=number of models. The (i,j) element of this matrix, then, is the weight that the j-th model should receive at the i-th time step.

Index