Package: Amelia (via r-universe)

October 21, 2024

Version 1.8.2

Date 2024-04-10

Title A Program for Missing Data

Depends R (>= 3.0.2), Rcpp (>= 0.11)

Imports foreign, utils, grDevices, graphics, methods, stats, rlang

LinkingTo Rcpp (>= 0.11), RcppArmadillo

Description A tool that ``multiply imputes'' missing data in a single cross-section (such as a survey), from a time series (like variables collected for each year in a country), or from a time-series-cross-sectional data set (such as collected by years for each of several countries). Amelia II implements our bootstrapping-based algorithm that gives essentially the same answers as the standard IP or EMis approaches, is usually considerably faster than existing approaches and can handle many more variables. Unlike Amelia I and other statistically rigorous imputation software, it virtually never crashes (but please let us know if you find to the contrary!). The program also generalizes existing approaches by allowing for trends in time series across observations within a cross-sectional unit, as well as priors that allow experts to incorporate beliefs they have about the values of missing cells in their data. Amelia II also includes useful diagnostics of the fit of multiple imputation models. The program works from the R command line or via a graphical user interface that does not require users to know R.

License GPL (>= 2)

URL https://gking.harvard.edu/amelia

Suggests tcltk, broom, rmarkdown, knitr

VignetteBuilder knitr

RoxygenNote 7.2.3

Encoding UTF-8

LazyData true

NeedsCompilation yes

Author James Honaker [aut], Gary King [aut], Matthew Blackwell [aut, cre] (<https://orcid.org/0000-0002-3689-9527>)

Maintainer Matthew Blackwell <mblackwell@gov.harvard.edu>

Date/Publication 2024-04-23 21:00:13 UTC

Repository https://mattblackwell.r-universe.dev

RemoteUrl https://github.com/cran/Amelia

RemoteRef HEAD

RemoteSha b547e6a99be8bcc72e277fd7777e22fe70684d1f

Contents

`africa` *Economic and Political Indictors in 6 African States*

Description

Data on a few economic and political variables in six African States from 1972-1991. The variables are year, country name, Gross Domestic Product per capita, inflation, trade as a percentage of GDP, a measure of civil liberties and total population. The data is from the Africa Research Program. A few cells are missing.

Usage

```
africa
```

Format

A data frame with 7 variables and 120 observations.

Source

Africa Research Program https://scholar.harvard.edu/rbates/data

---

`amelia` *AMELIA: Multiple Imputation of Incomplete Multivariate Data*

Description

Runs the bootstrap EM algorithm on incomplete data and creates imputed datasets.

Usage

```
amelia(x, ...)

## S3 method for class 'amelia'
amelia(x, m = 5, p2s = 1, frontend = FALSE, ...)

## S3 method for class 'molist'
amelia(x, ...)

## Default S3 method:
amelia(
  x,
  m = 5,
  p2s = 1,
  frontend = FALSE,
  idvars = NULL,
  ts = NULL,
  cs = NULL,
  polytime = NULL,
  splinetime = NULL,
  intercs = FALSE,
  lags = NULL,
  leads = NULL,
  startvals = 0,
  tolerance = 1e-04,
  logs = NULL,
  sqrts = NULL,
  lgstc = NULL,
```

```
  noms = NULL,
  ords = NULL,
  incheck = TRUE,
  collect = FALSE,
  arglist = NULL,
  empri = NULL,
  priors = NULL,
  autopri = 0.05,
  emburn = c(0, 0),
  bounds = NULL,
  max.resample = 100,
  overimp = NULL,
  boot.type = "ordinary",
  parallel = c("no", "multicore", "snow"),
  ncpus = getOption("amelia.ncpus", 1L),
  cl = NULL,
  ...
)
```

Arguments

x	either a matrix, data.frame, a object of class "amelia", or an object of class "molist". The first two will call the default S3 method. The third a convenient way to perform more imputations with the same parameters. The fourth will impute based on the settings from moPrep and any additional arguments.
...	further arguments to be passed.
m	the number of imputed datasets to create.
p2s	an integer value taking either 0 for no screen output, 1 for normal screen printing of iteration numbers, and 2 for detailed screen output. See "Details" for specifics on output when p2s=2.
frontend	a logical value used internally for the GUI.
idvars	a vector of column numbers or column names that indicates identification variables. These will be dropped from the analysis but copied into the imputed datasets.
ts	column number or variable name indicating the variable identifying time in time series data.
cs	column number or variable name indicating the cross section variable.
polytime	integer between 0 and 3 indicating what power of polynomial should be included in the imputation model to account for the effects of time. A setting of 0 would indicate constant levels, 1 would indicate linear time effects, 2 would indicate squared effects, and 3 would indicate cubic time effects.
splinetime	interger value of 0 or greater to control cubic smoothing splines of time. Values between 0 and 3 create a simple polynomial of time (identical to the polytime argument). Values k greater than 3 create a spline with an additional k-3 knot-points.

`intercs` a logical variable indicating if the time effects of `polytime` should vary across the cross-section.

`lags` a vector of numbers or names indicating columns in the data that should have their lags included in the imputation model.

`leads` a vector of numbers or names indicating columns in the data that should have their leads (future values) included in the imputation model.

`startvals` starting values, 0 for the parameter matrix from listwise deletion, 1 for an identity matrix.

`tolerance` the convergence threshold for the EM algorithm.

`logs` a vector of column numbers or column names that refer to variables that require log-linear transformation.

`sqrts` a vector of numbers or names indicating columns in the data that should be transformed by a sqaure root function. Data in this column cannot be less than zero.

`lgstc` a vector of numbers or names indicating columns in the data that should be transformed by a logistic function for proportional data. Data in this column must be between 0 and 1.

`noms` a vector of numbers or names indicating columns in the data that are nominal variables.

`ords` a vector of numbers or names indicating columns in the data that should be treated as ordinal variables.

`incheck` a logical indicating whether or not the inputs to the function should be checked before running `amelia`. This should only be set to `FALSE` if you are extremely confident that your settings are non-problematic and you are trying to save computational time.

`collect` a logical value indicating whether or not the garbage collection frequency should be increased during the imputation model. Only set this to `TRUE` if you are experiencing memory issues as it can significantly slow down the imputation process

`arglist` an object of class "ameliaArgs" from a previous run of Amelia. Including this object will use the arguments from that run.

`empri` number indicating level of the empirical (or ridge) prior. This prior shrinks the covariances of the data, but keeps the means and variances the same for problems of high missingness, small N's or large correlations among the variables. Should be kept small, perhaps 0.5 to 1 percent of the rows of the data; a reasonable upper bound is around 10 percent of the rows of the data.

`priors` a four or five column matrix containing the priors for either individual missing observations or variable-wide missing values. See "Details" for more information.

`autopri` allows the EM chain to increase the empirical prior if the path strays into an nonpositive definite covariance matrix, up to a maximum empirical prior of the value of this argument times `n`, the number of observations. Must be between 0 and 1, and at zero this turns off this feature.

emburn	a numeric vector of length 2, where emburn[1] is a the minimum EM chain length and emburn[2] is the maximum EM chain length. These are ignored if they are less than 1.
bounds	a three column matrix to hold logical bounds on the imputations. Each row of the matrix should be of the form c(column.number, lower.bound,upper.bound) See Details below.
max.resample	an integer that specifies how many times Amelia should redraw the imputed values when trying to meet the logical constraints of bounds. After this value, imputed values are set to the bounds.
overimp	a two-column matrix describing which cells are to be overimputed. Each row of the matrix should be a c(row,column) pair. Each of these cells will be treated as missing and replaced with draws from the imputation model.
boot.type	choice of bootstrap, currently restricted to either "ordinary" for the usual non-parametric bootstrap and "none" for no bootstrap.
parallel	the type of parallel operation to be used (if any). If missing, the default is taken from the option "amelia.parallel" (and if that is not set, "no").
ncpus	integer: the number of processes to be used in parallel operation: typically one would choose the number of available CPUs.
cl	an optional **parallel** or **snow** cluster for use if parallel = "snow". If not supplied, a cluster on the local machine is created for the duration of the amelia call.

Details

Multiple imputation is a method for analyzing incomplete multivariate data. This function will take an incomplete dataset in either data frame or matrix form and return m imputed datatsets with no missing values. The algorithm first creates a bootstrapped version of the original data, estimates the sufficient statistics (with priors if specified) by EM on this bootstrapped sample, and then imputes the missing values of the original data using the estimated sufficient statistics. It repeats this process m times to produce the m complete datasets where the observed values are the same and the unobserved values are drawn from their posterior distributions.

The function will start a "fresh" run of the algorithm if x is either a incomplete matrix or data.frame. In this method, all of the options will be user-defined or set to their default. If x is the output of a previous Amelia run (that is, an object of class "amelia"), then Amelia will run with the options used in that previous run. This is a convenient way to run more imputations of the same model.

You can provide Amelia with informational priors about the missing observations in your data. To specify priors, pass a four or five column matrix to the priors argument with each row specifying a different priors as such:

```
one.prior <- c(row, column, mean,standard deviation)
```

or,

```
one.prior <- c(row, column, minimum, maximum, confidence).
```

So, in the first and second column of the priors matrix should be the row and column number of the prior being set. In the other columns should either be the mean and standard deviation of the prior, or a minimum, maximum and confidence level for the prior. You must specify your priors all as

distributions or all as confidence ranges. Note that ranges are converted to distributions, so setting a confidence of 1 will generate an error.

Setting a priors for the missing values of an entire variable is done in the same manner as above, but inputing a `0` for the row instead of the row number. If priors are set for both the entire variable and an individual observation, the individual prior takes precedence.

In addition to priors, Amelia allows for logical bounds on variables. The `bounds` argument should be a matrix with 3 columns, with each row referring to a logical bound on a variable. The first column should be the column number of the variable to be bounded, the second column should be the lower bounds for that variable, and the third column should be the upper bound for that variable. As Amelia enacts these bounds by resampling, particularly poor bounds will end up resampling forever. Amelia will stop resampling after `max.resample` attempts and simply set the imputation to the relevant bound.

If each imputation is taking a long time to converge, you can increase the empirical prior, `empri`. This value has the effect of smoothing out the likelihood surface so that the EM algorithm can more easily find the maximum. It should be kept as low as possible and only used if needed.

Amelia assumes the data is distributed multivariate normal. There are a number of variables that can break this assumption. Usually, though, a transformation can make any variable roughly continuous and unbounded. We have included a number of commonly needed transformations for data. Note that the data will not be transformed in the output datasets and the transformation is simply useful for climbing the likelihood.

Amelia can run its imputations in parallel using the methods of the **parallel** package. The `parallel` argument names the parallel backend that Amelia should use. Users on Windows systems must use the `"snow"` option and users on Unix-like systems should use `"multicore"`. The `multicore` backend sets itself up automatically, but the `snow` backend requires more setup. You can pass a predefined cluster from the `parallel::makePSOCKcluster` function to the `cl` argument. Without this cluster, Amelia will attempt to create a reasonable default cluster and stop it once computation is complete. When using the parallel backend, users can set the number of CPUs to use with the `ncpus` argument. The defaults for these two arguments can be set with the options `"amelia.parallel"` and `"amelia.ncpus"`.

Please refer to the Amelia manual for more information on the function or the options.

Value

An instance of S3 class "amelia" with the following objects:

`imputations`	a list of length `m` with an imputed dataset in each entry. The class (matrix or data.frame) of these entries will match `x`.
`m`	an integer indicating the number of imputations run.
`missMatrix`	a matrix identical in size to the original dataset with 1 indicating a missing observation and a 0 indicating an observed observation.
`theta`	An array with dimensions $(p+1)$ by $(p+1)$ by m (where p is the number of variables in the imputations model) holding the converged parameters for each of the `m` EM chains.
`mu`	A p by m matrix of of the posterior modes for the complete-data means in each of the EM chains.

covMatrices	An array with dimensions (p) by (p) by m where the first two dimensions hold the posterior modes of the covariance matrix of the complete data for each of the EM chains.
code	a integer indicating the exit code of the Amelia run.
message	an exit message for the Amelia run
iterHist	a list of iteration histories for each EM chain. See documentation for details.
arguments	a instance of the class "ameliaArgs" which holds the arguments used in the Amelia run.
overvalues	a vector of values removed for overimputation. Used to reformulate the original data from the imputations.

Note that the `theta`, `mu` and `covMatrcies` objects refers to the data as seen by the EM algorithm and is thusly centered, scaled, stacked, tranformed and rearranged. See the manual for details and how to access this information.

Methods (by class)

- `amelia(amelia)`: Run additional imputations for Amelia output
- `amelia(molist)`: Perform multiple overimputation from moPrep
- `amelia(default)`: Run core Amelia algorithm

Author(s)

James Honaker

Gary King

Matt Blackwell

References

Honaker, J., King, G., Blackwell, M. (2011). Amelia II: A Program for Missing Data. *Journal of Statistical Software*, **45(7)**, 1–47. doi:10.18637/jss.v045.i07

See Also

For imputation diagnostics, `missmap`, `compare.density`, `overimpute` and `disperse`. For time series plots, `tscsPlot`. Also: `plot.amelia`, `write.amelia`, and `ameliabind`

Examples

```
data(africa)
a.out <- amelia(x = africa, cs = "country", ts = "year", logs = "gdp_pc")
summary(a.out)
plot(a.out)
```

ameliabind *Combine multiple runs of Amelia*

Description

Combines multiple runs of amelia with the same arguments and data into one amelia object.

Usage

```
ameliabind(...)
```

Arguments

...	one or more objects of class amelia with the same arguments and created from the same data.

Details

ameliabind will combine multiple runs of amelia into one object so that you can utilize diagnostics and modelling on all the imputations together. This function is useful for combining multiple runs of amelia run on parallel machines.

Note that ameliabind only checks that they arguments and the missingness matrix are identical. Thus, it could be fooled by two datasets that are identical up to a transformation of one variable.

Value

An object of class amelia.

See Also

amelia

Examples

```
data(africa)
a1.out <- amelia(x = africa, cs = "country", ts = "year", logs = "gdp_pc")
a2.out <- amelia(x = africa, cs = "country", ts = "year", logs = "gdp_pc")
all.out <- ameliabind(a1.out, a2.out)
summary(all.out)
plot(all.out)
```

ameliagui *Interactive GUI for Amelia*

Description

Brings up the AmeliaView graphical interface, which allows users to load datasets, manage options and run Amelia from a traditional windowed environment.

Usage

```
AmeliaView()
```

AmeliaView *Interactive GUI for Amelia*

Description

Brings up the AmeliaView graphical interface, which allows users to load datasets, manage options and run Amelia from a traditional windowed environment.

Usage

```
AmeliaView()
```

Details

Requires the tcltk package.

combine.output *Combine Multiple Amelia Output Lists*

Description

This function combines output lists from multiple runs of Amelia, where each run used the same arguments. The result is one list, formatted as if Amelia had been run once.

Usage

```
combine.output(...)
```

Arguments

...	a list of Amelia output lists from runs of Amelia with the same arguments except the number of imputations.

Details

This function is useful for combining the output from Amelia runs that occurred at different times or in different sessions of R. It assumes that the arguments given to the runs of Amelia are the same except for `m`, the number of imputations, and it uses the arguments from the first output list as the arguments for the combined output list.

`compare.density` *Compare observed versus imputed densities*

Description

Plots smoothed density plots of observed and imputed values from output from the `amelia` function.

Usage

```
compare.density(
  output,
  var,
  col = c("indianred", "dodgerblue"),
  scaled = FALSE,
  lwd = 1,
  main,
  xlab,
  ylab,
  legend = TRUE,
  frontend = FALSE,
  ...
)
```

Arguments

`output`	output from the function `amelia`.
`var`	column number or variable name of the variable to plot.
`col`	a vector of length 2 containing the color to plot the (1) imputed density and (2) the observed density.
`scaled`	a logical indicating if the two densities should be scaled to reflect the difference in number of units in each.
`lwd`	the line width of the density plots.
`main`	main title of the plot. The default is to title the plot using the variable name.
`xlab`	the label for the x-axis. The default is the name of the variable.
`ylab`	the label for the y-axis. The default is "Relative Density."
`legend`	a logical value indicating if a legend should be plotted.
`frontend`	a logical value used internally for the Amelia GUI.
`...`	further graphical parameters for the plot.

Details

This function first plots a density plot of the observed units for the variable `var` in `col[2]`. The the function plots a density plot of the mean or modal imputations for the missing units in `col[1]`. If a variable is marked "ordinal" or "nominal" with the `ords` or `noms` options in `amelia`, then the modal imputation will be used. If `legend` is `TRUE`, then a legend is plotted as well.

References

Abayomi, K. and Gelman, A. and Levy, M. 2005 "Diagnostics for Multivariate Imputations," *Applied Statistics*. 57,3: 273–291.

See Also

For more information on how densities are computed, `density`; Other imputation diagnostics are `overimpute`, `disperse`, and `tscsPlot`.

Examples

```
data(africa)
```

`disperse` *Overdispersed starting values diagnostic for multiple imputation*

Description

A visual diagnostic of EM convergence from multiple overdispersed starting values for an output from `amelia`.

Usage

```
disperse(
  output,
  m = 5,
  dims = 1,
  p2s = 0,
  frontend = FALSE,
  ...,
  xlim = NULL,
  ylim = NULL
)
```

Arguments

output	output from the function amelia.
m	the number of EM chains to run from overdispersed starting values.
dims	the number of principle components of the parameters to display and assess convergence on (up to 2).
p2s	an integer that controls printing to screen. 0 (default) indicates no printing, 1 indicates normal screen output and 2 indicates diagnostic output.
frontend	a logical value used internally for the Amelia GUI.
...	further graphical parameters for the plot.
xlim	limits of the plot in the horizontal dimension.
ylim	limits of the plot in vertical dimension.

Details

This function tracks the convergence of m EM chains which start from various overdispersed starting values. This plot should give some indication of the sensitivity of the EM algorithm to the choice of starting values in the imputation model in output. If all of the lines converge to the same point, then we can be confident that starting values are not affecting the EM algorithm.

As the parameter space of the imputation model is of a high-dimension, this plot tracks how the first (and second if dims is 2) principle component(s) change over the iterations of the EM algorithm. Thus, the plot is a lower dimensional summary of the convergence and is subject to all the drawbacks inherent in said summaries.

For dims==1, the function plots a horizontal line at the position where the first EM chain converges. Thus, we are checking that the other chains converge close to that horizontal line. For dims==2, the function draws a convex hull around the point of convergence for the first EM chain. The hull is scaled to be within the tolerance of the EM algorithm. Thus, we should check that the other chains end up in this hull.

See Also

Other imputation diagnostics are compare.density, disperse, and tscsPlot

---

freetrade *Trade Policy and Democracy in 9 Asian States*

Description

Economic and political data on nine developing countries in Asia from 1980 to 1999. This dataset includes 9 variables including year, country, average tariff rates, Polity IV score, total population, gross domestic product per capita, gross international reserves, a dummy variable for if the country had signed an IMF agreement in that year, a measure of financial openness, and a measure of US hegemony. These data were used in Milner and Kubota (2005).

Usage

```
freetrade
```

Format

A data frame with 10 variables and 171 observations.

Source

World Bank, World Trade Organization, Polity IV and others.

References

Helen Milner and Keiko Kubota (2005), "Why the move to free trade? Democracy and trade policy in the developing countries." *International Organization*, Vol 59, Issue 1.

---

mi.combine *Combine results from analyses on imputed data sets*

Description

Combine results from statistical models run on multiply imputed data sets using the so-called Rubin rules.

Usage

```
mi.combine(x, conf.int = FALSE, conf.level = 0.95)
```

Arguments

x	List of output from statistical models estimated on different imputed data sets, as outputted by `with(a.out,expr)` where `a.out` is the output of a call to `amelia`.
conf.int	Logical indicating if confidence intervals should be computed for each quantity of interest (default is `FALSE`).
conf.level	The confidence level to use for the confidence interval if `conf.level = TRUE`. Defaults to 0.95, which corresponds to a 95 percent confidence interval.

Value

Returns a `tibble` that contains:

term Name of the coefficient or parameter.

estimate Estimate of the parameter, averagine across imputations.

std.error Standard error of the estimate, accounting for imputation uncertainty.

statistic Value of the t-statistic for the estimated parameter.

p.value p-value associated with the test of a null hypothesis that the true coefficient is zero. Uses the t-distribution with an imputation-adjusted degrees of freedom.

df Imputation-adjusted degrees of freedom for each parameter.

r Relative increase in variance due to nonresponse.

miss.info Estimated fraction of missing information.

conf.low Lower bound of the estimated confidence interval. Only present if `conf.int = TRUE`.

conf.high Upper bound of the estimated confidence interval. Only present if `conf.int = TRUE`.

Author(s)

Matt Blackwell

Examples

```
data(africa)
a.out <- amelia(x = africa, cs = "country", ts = "year", logs =
"gdp_pc")

imp.mods <- with(a.out, lm(gdp_pc ~ infl + trade))

mi.combine(imp.mods, conf.int = TRUE)
```

`mi.meld` *Combine Multiple Results From Multiply Imputed Datasets*

Description

Combine sets of estimates (and their standard errors) generated from different multiply imputed datasets into one set of results.

Usage

```
mi.meld(q, se, byrow = TRUE)
```

Arguments

`q`	A matrix or data frame of (k) quantities of interest (eg. coefficients, parameters, means) from (m) multiply imputed datasets. Default is to assume the matrix is m-by-k (see `byrow`), thus each row represents a set of results from one dataset, and each column represents the different values of a particular quantity of interest across the imputed datasets.
`se`	A matrix or data frame of standard errors that correspond to each of the elements of the quantities of interest in `q`. Should be the same dimensions as `q`.
`byrow`	logical. If `TRUE`, `q` and `se` are treated as though each row represents the set of results from one dataset (thus m-by-k). If `FALSE`, each column represents results from one dataset (thus k-by-m).

Details

Uses Rubin's rules for combining a set of results from multiply imputed datasets to reflect the average result, with standard errors that both average uncertainty across models and account for disagreement in the estimated values across the models.

Value

`q.mi`	Average value of each quantity of interest across the m models
`se.mi`	Standard errors of each quantity of interest

References

Rubin, D. (1987). *Multiple Imputation for Nonresponse in Surveys*. New York: Wiley.

Honaker, J., King, G., Honaker, J. Joseph, A. Scheve K. (2001). Analyzing Incomplete Political Science Data: An Alternative Algorithm for Multiple Imputation *American Political Science Review*, **95(1)**, 49–69. (p53)

`missmap` *Missingness Map*

Description

Plots a missingness map showing where missingness occurs in the dataset passed to `amelia`.

Usage

```
missmap(
  obj,
  vars,
  legend = TRUE,
  col,
  main,
  y.cex = 0.8,
  x.cex = 0.8,
  y.labels,
  y.at,
  csvar = NULL,
  tsvar = NULL,
  rank.order = TRUE,
  margins = c(5, 5),
  gap.xaxis = 1,
  x.las = 2,
  ...
)
```

Arguments

obj	an object of class "amelia"; typically output from the function amelia, a matrix or a dataframe.
vars	a vector of column numbers or column names of the data to include in the plot. The default is to plot all variables.
legend	should a legend be drawn? (True or False)
col	a vector of length two where the first element specifies the color for missing cells and the second element specifies
main	main title of the plot. Defaults to "Missingness Map".
y.cex	expansion for the variables names on the x-axis.
x.cex	expansion for the unit names on the y-axis.
y.labels	a vector of row labels to print on the y-axis
y.at	a vector of the same length as y.labels with row nmumbers associated with the labels.
csvar	column number or name of the variable corresponding to the unit indicator. Only used when the obj is not of class amelia.
tsvar	column number or name of the variable corresponding to the time indicator. Only used when the obj is not of class amelia.
rank.order	a logical value. If TRUE, the default, then the order of the variables along the the x-axis is sorted by the percent missing (from highest to lowest). If FALSE, it is simply the order of the variables in the data.
margins	a vector of length two that specifies the bottom and left margins of the plot. Useful for when variable names or row names are long.
gap.xaxis	value to pass to the gap.axis argument of the axis function that plots the x-axis. See axis for more details. Ignored on R versions less than 4.0.0.
x.las	value of the las argument to pass to the axis function creating the x-axis.
...	further graphical arguments.

Details

missmap draws a map of the missingness in a dataset using the image function. The columns are reordered to put the most missing variable farthest to the left. The rows are reordered to a unit-period order if the ts and cs arguments were passed to amelia. If not, the rows are not reordered.

The y.labels and y.at commands can be used to associate labels with rows in the data to identify them in the plot. The y-axis is internally inverted so that the first row of the data is associated with the top-most row of the missingness map. The values of y.at should refer to the rows of the data, not to any point on the plotting region.

See Also

compare.density, overimpute, tscsPlot, image, heatmap

moPrep *Prepare Multiple Overimputation Settings*

Description

A function to generate priors for multiple overimputation of a variable measured with error.

Usage

```
moPrep(
  x,
  formula,
  subset,
  error.proportion,
  gold.standard = !missing(subset),
  error.sd
)

## S3 method for class 'molist'
moPrep(x, formula, subset, error.proportion, gold.standard = FALSE, error.sd)

## Default S3 method:
moPrep(
  x,
  formula,
  subset,
  error.proportion,
  gold.standard = !missing(subset),
  error.sd
)
```

Arguments

x	either a matrix, data.frame, or a object of class "molist" from a previous moPrep call. The first two derive the priors from the data given, and the third will derive the priors from the first moPrep call and add them to the already defined priors.
formula	a formula describing the nature of the measurement error for the variable. See "Details."
subset	an optional vector specifying a subset of observations which possess measurement error.
error.proportion	an optional vector specifying the fraction of the observed variance that is due to measurement error.
gold.standard	a logical value indicating if values with no measurement error should be used to estimate the measurement error variance.
error.sd	an optional vector specifying the standard error of the measurement error.

Details

This function generates priors for multiple overimputation of data measured with error. With the `formula` arugment, you can specify which variable has the error, what the mean of the latent data is, and if there are any other proxy measures of the mismeasured variable. The general syntax for the formula is: `errvar ~ mean | proxy`, where `errvar` is the mismeasured variable, `mean` is a formula for the mean of the latent variable (usually just `errvar` itself), and `proxy` is a another mismeasurement of the same latent variable. The proxies are used to estimate the variance of the measurement error.

`subset` and `gold.standard` refer to the the rows of the data which are and are not measured with error. Gold-standard rows are used to estimate the variance of the measurement. error. `error.proportion` is used to estimate the variance of the measurement error by estimating the variance of the mismeasurement and taking the proportion assumed to be due to error. `error.sd` sets the standard error of the measurement error directly.

Value

An instance of the S3 class "molist" with the following objects:

- priors a four-column matrix of the multiple overimputation priors associated with the data. Each row of the matrix is `c(row,column, prior.mean, prior.sd)`
- overimp a two-column matrix of cells to be overimputed. Each row of the matrix is of the form `c(row, column)`, which indicate the row and column of the cell to be overimputed.
- data the object name of the matrix or data.frame to which priors refer.

Note that `priors` and `overimp` might contain results from multiple calls to `moPrep`, not just the most recent.

Methods (by class)

- `moPrep(molist)`: Alter existing moPrep output
- `moPrep(default)`: Default call to moPrep

See Also

`amelia`

Examples

```
data(africa)
m.out <- moPrep(africa, trade ~ trade, error.proportion = 0.1)
a.out <- amelia(m.out, ts = "year", cs = "country")
plot(a.out)
m.out <- moPrep(africa, trade ~ trade, error.sd = 1)
a.out <- amelia(m.out, ts = "year", cs = "country")
```

`overimpute` *Overimputation diagnostic plot*

Description

Treats each observed value as missing and imputes from the imputation model from `amelia` output.

Usage

```
overimpute(
  output,
  var,
  draws = 20,
  subset,
  legend = TRUE,
  xlab,
  ylab,
  main,
  frontend = FALSE,
  ...
)
```

Arguments

`output`	output from the function `amelia`.
`var`	column number or variable name of the variable to overimpute.
`draws`	the number of draws per imputed dataset to generate overimputations. Total number of simulations will `m * draws` where `m` is the number of imputations.
`subset`	an optional vector specifying a subset of observations to be used in the overimputation.
`legend`	a logical value indicating if a legend should be plotted.
`xlab`	the label for the x-axis. The default is "Observed Values."
`ylab`	the label for the y-axis. The default is "Imputed Values."
`main`	main title of the plot. The default is to smartly title the plot using the variable name.
`frontend`	a logical value used internally for the Amelia GUI.
`...`	further graphical parameters for the plot.

Details

This function temporarily treats each observed value in `var` as missing and imputes that value based on the imputation model of `output`. The dots are the mean imputation and the vertical lines are the 90% percent confidence intervals for imputations of each observed value. The diagonal line is the $y = x$ line. If all of the imputations were perfect, then our points would all fall on the line. A good

imputation model would have about 90% of the confidence intervals containing the truth; that is, about 90% of the vertical lines should cross the diagonal.

The color of the vertical lines displays the fraction of missing observations in the pattern of missingness for that observation. The legend codes this information. Obviously, the imputations will be much tighter if there are more observed covariates to use to impute that observation.

The `subset` argument evaluates in the environment of the data. That is, it can but is not required to refer to variables in the data frame as if it were attached.

Value

A list that contains (1) the row in the original data (`row`), (2) the observed value of that observation (`orig`), (2) the mean of the overimputations (`mean.overimputed`), (3) the lower bound of the 95% confidence interval of the overimputations (`lower.overimputed`), (4) the upper bound of the 95% confidence interval of the overimputations (`upper.overimputed`), (5) the fraction of the variables that were missing for that observation in the original data (`prcntmiss`), and (6) a matrix of the raw overimputations, with observations in rows and the different draws in columns (`overimps`).

See Also

Other imputation diagnostics are `compare.density`, `disperse`, and `tscsPlot`.

`plot.amelia` *Summary plots for Amelia objects*

Description

Plots diagnostic plots for the output from the `amelia` function.

Usage

```
## S3 method for class 'amelia'
plot(x, which.vars, compare = TRUE, overimpute = FALSE, ask = TRUE, ...)
```

Arguments

`x`	an object of class "amelia"; typically output from the function `amelia`.
`which.vars`	a vector indicating the variables to plot. The default is to plot all of the numeric variables that were actually imputed.
`compare`	plot the density comparisons for each variable (True or False)
`overimpute`	plot the overimputation for each variable (True or False)
`ask`	prompt user before changing pages of a plot (True or False)
`...`	further graphical arguments.

summary.amelia *Summary of an Amelia object*

Description

Returns summary information from the Amelia run along with missingles information.

Usage

```
## S3 method for class 'amelia'
summary(object, ...)
```

Arguments

object	an object of class amelia. Typically, an output from the function amelia.
...	further arguments.

See Also

amelia, plot.amelia

transform.amelia *Transform imputed datasets from Amelia objects*

Description

Updates the imputed datasets from an amelia output with the specified transformations.

Usage

```
## S3 method for class 'amelia'
transform(`_data`, ...)
```

Arguments

_data	an object of class "amelia"; typically output from the function amelia.
...	further arguments of the form tag = value.

Details

The ... arugments to transform.amelia are expressions of the form tag = value, where tag is the variable that is being updated or created and value is an expression that is a function of the variables in the imputed datasets. For instance, if you wanted to create an interaction of two imputed variables, you could have one argument be intervar = var1 * var2. This would either update the current variable intervar in the imputed data or append a new variable called intervar to the imputed datasets.

Value

An object of class `amelia` with its `imputations` and `missMatrix` values updated according to the transformations. In addition, each of the calls to `transform.amelia` are stored in

See Also

`transform`

`tscsPlot` *Plot observed and imputed time-series for a single cross-section*

Description

Plots a time series for a given variable in a given cross-section and provides confidence intervals for the imputed values.

Usage

```
tscsPlot(
  output,
  var,
  cs,
  draws = 100,
  conf = 0.9,
  misscol = "red",
  obscol = "black",
  xlab,
  ylab,
  main,
  pch,
  ylim,
  xlim,
  frontend = FALSE,
  plotall = FALSE,
  nr,
  nc,
  pdfstub,
  ...
)
```

Arguments

`output`	output from the function `amelia`.
`var`	the column number or variable name of the variable to plot.
`cs`	the name (or level) of the cross-sectional unit to plot. Maybe a vector of names which will panel a window of plots

draws	the number of imputations on which to base the confidence intervals.
conf	the confidence level of the confidence intervals to plot for the imputated values.
misscol	the color of the imputed values and their confidence intervals.
obscol	the color of the points for observed units.
xlab	x axis label
ylab	y axis label
main	overall plot title
pch	point shapes for the plot.
ylim	y limits (y1, y2) of the plot.
xlim	x limits (x1, x2) of the plot.
frontend	a logical value for use with the AmeliaView GUI.
plotall	a logical value that provides a shortcut for ploting all unique values of the level. A shortcut for the cs argument, a TRUE value overwrites any cs argument.
nr	the number of rows of plots to use when ploting multiple cross-sectional units. The default value will try to minimize this value to create a roughly square representation, up to a value of four. If all plots do not fit on the window, a new window will be started.
nc	the number of columns of plots to use. See nr
pdfstub	a stub string used to write pdf copies of each window created by the plot. The default is not to write pdf output, but any string value will turn on pdf output to the local working directory. If the stub is mystub, then plots will be saved as mystub1.pdf, mystub2.pdf, etc.
...	further graphical parameters for the plot.

Details

The cs argument should be a value from the variable set to the cs argument in the amelia function for this output. This function will not work if the ts and cs arguments were not set in the amelia function. If an observation has been overimputed, tscsPlot will plot both an observed and an imputed value.

---

with.amelia *Execute commands within each imputed data set*

Description

Evaluate an R expression in the environments constructed from the imputed data sets of a call to amelia function.

Usage

```
## S3 method for class 'amelia'
with(data, expr, ...)
```

Arguments

data	imputation output from the amelia funtion.
expr	expression to evaluate in each imputed data set in data.
...	arguments to be passed to (future) methods.

Value

a list the same length as data$imputations that contains the output of the expression as evaluated in each imputed data set of data.

Author(s)

Matt Blackwell

Examples

```
data(africa)
a.out <- amelia(x = africa, cs = "country", ts = "year", logs =
"gdp_pc")

imp.mods <- with(a.out, lm(gdp_pc ~ infl + trade))

mi.combine(imp.mods, conf.int = TRUE)
```

---

write.amelia *Write Amelia imputations to file*

Description

Writes the imptuted datasets to file from a run of amelia

Usage

```
write.amelia(
  obj,
  separate = TRUE,
  file.stem,
  extension = NULL,
  format = "csv",
  impvar = "imp",
  orig.data = TRUE,
  ...
)
```

Arguments

obj	an object of class "amelia"; typically output from the function amelia
separate	logical variable. If TRUE (default), the imputed datasets will be written to separate files, whose names come from the file.stem and extension arguments. If FALSE, the imputations are stacked and written as a single file.
file.stem	the leading part of the filename to save to output The imputation number and extension will be added to complete the filename. This can include a directory path.
extension	the extension of the filename. This is simply what follows file.stem and the imputation number.
format	one of the following output formats: csv, dta or table. See details.
impvar	the name of imputation number variable written to the stacked dataset when separate is FALSE.
orig.data	logical variable indicating whether the original, unimputed dataset should be included in the stacked dataset when separate is FALSE.
...	further arguments for the write functions.

Details

write.amelia writes the imputed datasets to a file or a set of files using one of the following functions: write.csv, write.dta, or write.table. You can pass arguments to these functions from write.amelia.

When separate is TRUE, each imputed dataset is written to its own file. If you were to set file.stem to "outdata" and the extension to ".csv" , then the resulting filename of the written files will be

```
outdata1.csv
outdata2.csv
outdata3.csv
...
```

and so on.

When separate is FALSE, the function adds a variable called impvar to each dataset which indicates the imputed dataset to which the row belongs. Then, each of the datasets are stacked together to create one dataset. If orig.data is TRUE, then the original, unimputed dataset is included at the top of the stack, with its imputation number set to 0.

See Also

write.csv, write.table, write.dta

Index