Package ‘miceadds’

February 23, 2015

**Type** Package

**Title** Some Additional Multiple Imputation Functions, Especially for 'mice'

**Version** 1.1-1

**Date** 2015-02-23

**Author** Alexander Robitzsch [aut, cre]

**Maintainer** Alexander Robitzsch <a.robitzsch@bifie.at>

**Description** Contains some auxiliary functions for multiple imputation which complements existing functionality in R.
In addition to some utility functions, main features include plausible value imputation, multilevel imputation functions, imputation using partial least squares (PLS) for high dimensional predictors and two-way imputation.

**Depends** R (>= 2.15.0), MASS, mice, mvtnorm, pan

**Imports** Rcpp, sirt, TAM, lme4, MBESS, pls, mitools, bayesm, inline, car, foreign

**LinkingTo** Rcpp, RcppArmadillo

**Suggests** Amelia, Zelig, jomo

**License** GPL (>= 2)

**URL** [https://sites.google.com/site/alexanderrobitzsch/software](https://sites.google.com/site/alexanderrobitzsch/software)

**NeedsCompilation** yes

**Repository** CRAN

**Date/Publication** 2015-02-23 23:08:09

R topics documented:

- `miceadds-package`
- `crlrem`
- `cxxfunction.copy`
- `data.allison`

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>miceadds-package</td>
<td>3</td>
</tr>
<tr>
<td>crlrem</td>
<td>4</td>
</tr>
<tr>
<td>cxxfunction.copy</td>
<td>5</td>
</tr>
<tr>
<td>data.allison</td>
<td>6</td>
</tr>
<tr>
<td>R topics documented:</td>
<td></td>
</tr>
<tr>
<td>--------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>data.enders</td>
<td>8</td>
</tr>
<tr>
<td>data.internet</td>
<td>10</td>
</tr>
<tr>
<td>data.largescale</td>
<td>12</td>
</tr>
<tr>
<td>data.ma</td>
<td>12</td>
</tr>
<tr>
<td>data.smallscale</td>
<td>15</td>
</tr>
<tr>
<td>datalist2mids</td>
<td>15</td>
</tr>
<tr>
<td>draw.pv.ctt</td>
<td>17</td>
</tr>
<tr>
<td>fast.groupmean</td>
<td>19</td>
</tr>
<tr>
<td>grep.vec</td>
<td>20</td>
</tr>
<tr>
<td>index.dataframe</td>
<td>21</td>
</tr>
<tr>
<td>kernelpls.fit2</td>
<td>21</td>
</tr>
<tr>
<td>load.data</td>
<td>23</td>
</tr>
<tr>
<td>load.Rdata</td>
<td>24</td>
</tr>
<tr>
<td>ma.scale2</td>
<td>25</td>
</tr>
<tr>
<td>ma.wtd.statNA</td>
<td>27</td>
</tr>
<tr>
<td>mi.anova</td>
<td>28</td>
</tr>
<tr>
<td>mice.1chain</td>
<td>29</td>
</tr>
<tr>
<td>mice.impute.2l.contextual.norm</td>
<td>33</td>
</tr>
<tr>
<td>mice.impute.2l.contextual.pmm</td>
<td>34</td>
</tr>
<tr>
<td>mice.impute.2l.leap</td>
<td>36</td>
</tr>
<tr>
<td>mice.impute.2l.latentgroupmean</td>
<td>38</td>
</tr>
<tr>
<td>mice.impute.2l.plausible.values</td>
<td>41</td>
</tr>
<tr>
<td>mice.impute.2l.pls2</td>
<td>46</td>
</tr>
<tr>
<td>mice.impute.2lonly.pmm2</td>
<td>49</td>
</tr>
<tr>
<td>mice.impute.pmm3</td>
<td>50</td>
</tr>
<tr>
<td>mice.impute.tricube.pmm</td>
<td>52</td>
</tr>
<tr>
<td>mice.impute.weighted.norm</td>
<td>54</td>
</tr>
<tr>
<td>mice.impute.weighted.pmm</td>
<td>55</td>
</tr>
<tr>
<td>micombine.chisquare</td>
<td>56</td>
</tr>
<tr>
<td>micombine.cor</td>
<td>57</td>
</tr>
<tr>
<td>micombine.F</td>
<td>58</td>
</tr>
<tr>
<td>mids2datlist</td>
<td>59</td>
</tr>
<tr>
<td>output.format1</td>
<td>60</td>
</tr>
<tr>
<td>pca.covridge</td>
<td>61</td>
</tr>
<tr>
<td>Reval</td>
<td>62</td>
</tr>
<tr>
<td>Rhat.mice</td>
<td>63</td>
</tr>
<tr>
<td>round2</td>
<td>64</td>
</tr>
<tr>
<td>Rsessinfo</td>
<td>65</td>
</tr>
<tr>
<td>save.data</td>
<td>66</td>
</tr>
<tr>
<td>save.Rdata</td>
<td>67</td>
</tr>
<tr>
<td>scan.vec</td>
<td>68</td>
</tr>
<tr>
<td>source.all</td>
<td>69</td>
</tr>
<tr>
<td>str_C.expand.grid</td>
<td>69</td>
</tr>
<tr>
<td>sumpreserving.rounding</td>
<td>70</td>
</tr>
<tr>
<td>systime</td>
<td>72</td>
</tr>
<tr>
<td>tw.imputation</td>
<td>72</td>
</tr>
<tr>
<td>write.mice.imputation</td>
<td>74</td>
</tr>
<tr>
<td>write.pspp</td>
<td>76</td>
</tr>
</tbody>
</table>
**miceadds-package**

**Index**

<table>
<thead>
<tr>
<th>Package</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>miceadds-package</td>
<td>Some Additional Multiple Imputation Functions, Especially for mice</td>
</tr>
</tbody>
</table>

**Description**

Contains some auxiliary functions for multiple imputation which complements existing functionality in R. In addition to some utility functions, main features include plausible value imputation, multilevel imputation functions, imputation using partial least squares (PLS) for high dimensional predictors and two-way imputation.

**Details**

DESCRIPTION miceadds package

<table>
<thead>
<tr>
<th>Package:</th>
<th>miceadds</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type:</td>
<td>Package</td>
</tr>
<tr>
<td>Version:</td>
<td>1.1</td>
</tr>
<tr>
<td>Publication Year:</td>
<td>2015</td>
</tr>
<tr>
<td>License:</td>
<td>GPL (&gt;= 2)</td>
</tr>
<tr>
<td>URL:</td>
<td><a href="https://sites.google.com/site/alexanderrobitzsch/software">https://sites.google.com/site/alexanderrobitzsch/software</a></td>
</tr>
</tbody>
</table>

- In addition to the usual mice imputation function which employs parallel chains, the function `mice.1chain` does multiple imputation from a single chain.
- Imputation based on partial least squares regression is implemented in `mice.impute.2L.pls`.
- Unidimensional plausible value imputation for latent variables (or variables with measurement error) in the mice sequential imputation framework can be applied by using the method `mice.impute.2L.plausible.values`.
- Imputations for questionnaire items can be accomplished by two-way imputation (`tw.imputation`).
- The miceadds package also includes some functions R utility functions (e.g. write.pspp, ma.scale).

**Author(s)**

Alexander Robitzsch  
Federal Institute for Education Research, Innovation and Development of the Austrian School System (BIFIE Salzburg), Austria

Maintainer: Alexander Robitzsch <a.robitzsch@bifie.at>

URL: [https://sites.google.com/site/alexanderrobitzsch/](https://sites.google.com/site/alexanderrobitzsch/)

**See Also**

See other R packages for conducting multiple imputation: mice, Amelia, pan, mi, norm, BaBooN, VIM, ...
Some links to internet sites related to missing data:

http://missingdata.lshtm.ac.uk/
http://www.stefvanbuuren.nl/mi/
http://www.bristol.ac.uk/cmm/software/realcom/

Examples

```r
### :```````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````
**Author(s)**

This is code by Dirk Eddelbuettel copied from https://stat.ethz.ch/pipermail/r-devel/2010-September/058480.html

**Examples**

```r
## Not run:
filename1 <- "rm.arraymult_0.02.cpp"
filename2 <- "rm.arraymult_0.03.cpp"
crlrem( filename1, filename2 )
## End(Not run)
```

---

**Description**

Copy the Rcpp function into the working directory.

**Usage**

`cxxfunction.copy/cppfct, name)`

**Arguments**

- `cppfct`: The Rcpp function
- `name`: Name of the output Rcpp function to be generated

**Author(s)**

Alexander Robitzsch

**References**


**See Also**

See also the `cxxfunction` in the `inline` package
Examples

```r
## Not run:
# define Rcpp file
code1 <- 
// array A
Rcpp::NumericMatrix AA(A);
// Rcpp::IntegerVector dimAA(dimA);
int nrows = AA.nrow();
int ncolumns = AA.ncol();
NumericMatrix Alogis(nrows,ncolumns) ;
// compute logistic distribution
for (int ii=0; ii<nrows; ++ii){
    NumericVector h1=AA.row(ii) ;
    NumericVector res = plogis( h1 ) ;
    for (int jj=0; jj<ncolumns;++jj){
        Alogis(ii,jj) = res[jj] ;
    }
}

return( wrap(Alogis) );
```

# compile Rcpp code
calc1 <- cxxfunction( signature( A= "matrix"), code1, plugin = "Rcpp", verbose=TRUE )
cxxfunction.copy( cppfct=calc1, name="calclogis" )

## End(Not run)

---

data.allison  

Datasets from Allison’s Missing Data Book

Description

Datasets from Allison’s missing data book (Allison 2002).

Usage

data(data.allison.gssexp)
data(data.allison.hip)
data(data.allison.usnews)

Format

- Data `data.allison.gssexp`:

  'data.frame': 2991 obs. of 14 variables:
  $ AGE : num 33 59 NA 59 21 22 40 25 41 45 ...  
  $ EDUC : num 12 12 12 8 13 15 9 12 12 12 ...  
  $ FEMALE : num 1 0 1 0 1 1 0 1 1 ...  
  $ SPANKING: num 1 1 2 2 NA 1 3 1 1 NA ...
**Example Index**

*Dataset* data.allison.hip

*data.allison* (Example 1)

**Source**

The datasets are available from [http://www.ats.ucla.edu/stat/examples/md/](http://www.ats.ucla.edu/stat/examples/md/).

**References**


**Examples**

```r
## Not run:
```
# EXAMPLE 1: Hip datasets | Imputation using a wide format

# at first, the hip dataset is 'melted' for imputation

data(data.allison.hip)
<< head(data.allison.hip)
## 
## SID WAVE ADL PAIN SRH WALK CESD
## 1 1 1 3 0 2 1 9.000
## 2 1 2 2 5 4 0 28.000
## 3 1 3 3 0 2 0 31.000
## 4 1 4 3 0 2 0 11.579
## 5 2 1 3 0 4 0 NA
## 6 2 2 1 1 1 0 2.222

library(reshape)
hp.wide <- reshape(data.allison.hip, idvar = "SID", timevar = "WAVE", direction = "wide")
<< head(hp.wide, 2)
## 
## SID ADL.1 PAIN.1 SRH.1 WALK.1 CESD.1 ADL.2 PAIN.2 SRH.2 WALK.2 CESD.2 ADL.3
## 1 1 3 0 2 1 9 2 5 4 0 28.000 3
## 5 2 3 0 4 0 NA 1 1 1 0 2.222 2
## PAIN.3 SRH.3 WALK.3 CESD.3 ADL.4 PAIN.4 SRH.4 WALK.4 CESD.4
## 1 0 2 0 31 3 0 2 0 11.579
## 5 5 1 0 12 1 NA 2 0 NA

# imputation of the hip wide dataset
imp <- mice(as.matrix(hp.wide[-1]), m=5, maxit=3)
summary(imp)

## End(Not run)

data.enders  | **Datasets from Enders’ Missing Data Book**

**Description**


**Usage**

data(data.enders.depression)
data(data.enders.eatingattitudes)
data(data.enders.employee)

**Format**

- Dataset data.enders.depression:
  - 'data.frame': 280 obs. of 8 variables:
  - $ txgroup: int 0 0 0 0 0 0 0 0 0 0 ...
$ dep1 : int 46 49 40 47 33 44 45 53 40 55 ...
$ dep2 : int 44 42 28 47 33 41 43 35 43 45 ...
$ dep3 : int 26 29 31 NA 34 34 35 35 36 ...
$ r2 : int 0 0 0 0 0 0 0 0 0 0 ...
$ r3 : int 0 0 0 1 0 0 0 0 0 0 ...
$ pattern: int 3 3 3 2 3 3 3 3 3 3 ...
$ dropout: int 0 0 0 1 0 0 0 0 0 0 ...

- Dataset data.enders.eatingattitudes:

  'data.frame': 400 obs. of 14 variables:
  $ id : num 1 2 3 4 5 6 7 8 9 10 ...
  $ eat1 : num 4 6 3 3 3 4 5 4 4 6 ...
  $ eat2 : num 4 5 3 3 2 5 4 3 7 5 ...
  $ eat10: num 4 6 2 4 3 4 4 4 6 5 ...
  $ eat11: num 4 6 2 3 3 5 4 4 5 5 ...
  $ eat12: num 4 6 3 4 3 4 4 4 4 6 ...
  $ eat14: num 4 7 2 4 3 4 4 4 6 6 ...
  $ eat24: num 3 6 3 3 3 4 4 4 4 5 ...
  $ eat3 : num 4 5 3 3 4 4 3 6 4 5 ...
  $ eat18: num 5 6 3 5 4 5 3 6 4 6 ...
  $ eat21: num 4 5 2 4 4 4 3 5 4 5 ...
  $ bmi : num 18 9 26 18.3 18.2 24.4 ...
  $ wsb : num 9 13 6 5 10 7 11 8 10 12 ...
  $ anx : num 11 19 8 14 7 11 12 12 14 12 ..

- Dataset data.enders.employee:

  'data.frame': 480 obs. of 9 variables:
  $ id : num 1 2 3 4 5 6 7 8 9 10 ...
  $ age : num 40 53 46 37 44 39 33 43 35 37 ...
  $ tenure : num 10 14 10 8 9 10 7 9 9 10 ...
  $ female : num 1 1 1 1 1 1 1 1 1 1 ...
  $ wbeing : num 8 6 NA 7 NA 7 NA 7 7 5 ...
  $ jobsat : num 8 5 7 NA 5 NA 5 NA 7 6 ...
  $ jobperf : num 6 5 7 5 7 7 7 7 6 ...
  $ turnover: num 0 0 0 0 0 0 0 0 1 0 ...
  $ iq : num 106 93 107 94 107 118 103 106 108 97 ...

Example Index

Dataset data.enders.employee

Source

The datasets were downloaded from http://www.appliedmissingdata.com/book-examples.html.
References

data.internet

*Dataset Internet*

Description
Dataset with items corresponding to internet attitudes.

Usage
data(data.internet)

Format
A data frame with 281 observations on the following 22 variables.

The format of the dataset is

'data.frame': 281 obs. of 22 variables:
$ IN1 : num 1 5 2 3 1 3 2 3 2 1 ...
$ IN2 : num 4 3 2 7 4 4 7 4 3 ...
$ IN3 : num 4 5 4 2 1 2 5 2 2 4 ...
[...]
$ IN20: num 3 2 2 3 3 4 2 7 2 2 ...
$ IN21: num 3 3 6 5 4 4 5 5 6 5 ...
$ IN22: num 3 4 2 5 3 5 3 7 3 5 ...

Details
The following text is copied from [http://people.few.eur.nl/groenen/Data/index.htm](http://people.few.eur.nl/groenen/Data/index.htm)
The data set is based on a questionnaire on attitudes towards the Internet. It consists of evaluations of 22 statements about the Internet by 281 students at Erasmus University Rotterdam. These data were gathered around 2002 before the wide availability of broadband Internet access in the Netherlands. The statements were evaluated using a seven-point Likert scale, ranging from 1 (completely disagree) to 7 (completely agree).

We would like to thank Peter Verhoef for making these data available.

Each variable (statement) is coded as follows:
1. Completely disagree
2. Disagree
3. Slightly disagree
4. Neutral
5. Slightly agree
6. Agree
7. Completely agree

Internet items:

1. Paying using Internet is safe
2. Surfing the Internet is easy
3. Internet is unreliable
4. Internet is slow
5. Internet is user-friendly
6. Internet is the future’s means of communication
7. Internet is addictive
8. Internet is fast
9. Sending personal data using the Internet is unsafe
10. The prices of Internet subscriptions are high
11. Internet offers many possibilities for abuse
12. The costs of surfing are high
13. Internet offers unbounded opportunities
14. Internet phone costs are high
15. The content of web sites should be regulated
16. Internet is easy to use
17. I like surfing
18. I often speak with friends about the Internet
19. I like to be informed of important new things
20. I always attempt new things on the Internet first
21. I regularly visit websites recommended by others
22. I know much about the Internet

Example Index

\texttt{ma.scale2} (Example 1), \texttt{mice.impute.21.pls2} (Example 1), \texttt{pca.covridge} (Example 1), \texttt{tw.imputation} (Example 1)

Source

Peter Verhoef

\url{http://people.few.eur.nl/groenen/Data/index.htm}

Examples

\begin{verbatim}
data(data.internet) # missing proportions colMeans( is.na(data.internet) )
\end{verbatim}
### data.largescale

**Large-scale Dataset for Testing Purposes (Many Cases, Few Variables)**

**Description**

Large-scale dataset with many cases and few variables included for testing purposes.

**Usage**

```r
data(data.largescale)
```

**Format**

A data frame with 14000 observations on the following 13 variables. The format is a data frame with 14000 observations on the following 13 variables:

- `id`:
  - 1e+07 1e+07 1e+07 1e+07 1e+07 ... 1.0
- `D1`:
  - 0.0 0.0 0.0 1.0 0.0 0.0 0.0 ...
- `D2`:
  - 0.0 0.0 1.0 1.0 1.0 0.0 ...
- `D3`:
  - 0.0 0.0 0.0 0.0 0.0 0.0 0.0 ...
- `D4`:
  - 0.0 0.0 1.0 0.0 1.0 0.0 ...
- `D5`:
  - 0.0 0.0 0.0 1.0 0.0 0.0 ...
- `v1`:
  - 118 117 94 106 86 117 96 96 82 95 ...
- `v2`:
  - 101 101 86 101 65 94 72 75 70 99 ...
- `v3`:
  - 0.0 0.0 0.0 1.0 0.0 0.0 ...
- `v4`:
  - 3 NA 3 5 2 5 5 4 2 ...
- `v5`:
  - 0 NA 0 0 0 1 0 0 0 0 ...
- `v6`:
  - 3 3 3 4 NA 1 3 3 2 3 ...
- `v7`:
  - 51 36 14 47 22 17 13 37 47 38 ...

### data.ma

**Example Datasets for miceadds Package**

**Description**

Example datasets for miceadds package.

**Usage**

```r
data(data.ma01)
data(data.ma02)
data(data.ma03)
data(data.ma04)
data(data.ma05)
```
Data.ma

Format

• Dataset data.ma01:
  Dataset with students nested within school and student weights (studwgt). The format is
  'data.frame': 4073 obs. of 11 variables:
  $ idstud : num 1e+07 1e+07 1e+07 1e+07 1e+07 1e+07 ... 
  $ idschool: num 1001 1001 1001 1001 1001 1001 ... 
  $ studwgt : num 6.05 6.05 5.27 5.27 6.05 ... 
  $ math  : int 594 605 616 524 685 387 536 594 387 562 ...
  $ read  : int 647 651 539 551 689 502 503 597 580 576 ...
  $ migrant: int 0 0 0 1 0 0 0 0 0 ...
  $ books : int 6 6 5 2 6 3 4 6 5 ...
  $ hissei: int NA 77 69 45 66 53 43 NA 64 50 ...
  $ paredu: int 3 7 7 2 7 3 4 NA 7 3 ...
  $ female: int 1 1 0 0 1 1 0 0 1 1 ...
  $ urban : num 1 1 1 1 1 1 1 1 1 1 ...

• Dataset data.ma02:
  10 multiply imputed datasets of incomplete data data.ma01. The format is
  List of 10
  $ :'data.frame': 4073 obs. of 11 variables:
  $ :'data.frame': 4073 obs. of 11 variables:
  $ :'data.frame': 4073 obs. of 11 variables:
  $ :'data.frame': 4073 obs. of 11 variables:
  $ :'data.frame': 4073 obs. of 11 variables:
  $ :'data.frame': 4073 obs. of 11 variables:
  $ :'data.frame': 4073 obs. of 11 variables:
  $ :'data.frame': 4073 obs. of 11 variables:
  $ :'data.frame': 4073 obs. of 11 variables:

• Dataset data.ma03:
  This dataset contains one variable math_EAP for which a conditional posterior distribution
  with EAP and its associated standard deviation is available.
  'data.frame': 120 obs. of 8 variables:
  $ idstud : int 1001 1002 1003 1004 1005 1006 1007 1008 1009 1010 ...
  $ female : int 0 1 1 1 0 1 1 1 1 ...
  $ migrant: int 1 1 0 1 1 0 0 0 1 0 ...
  $ hissei: int 44 NA 26 NA 32 60 31 NA 34 26 ...
  $ educ  : int NA 2 NA 1 4 NA 2 NA 2 NA ...
  $ read_wle: num 74.8 78.1 103.2 81.2 119.2 ...
  $ math_EAP: num 337 342 264 285 420 ...
  $ math_SEEAP: num 28 29.5 28.6 28.5 27.5 ...

• Dataset data.ma04:
  This dataset contains two hypothetical scales A and B and single variables V5, V6 and V7.
## Example Index

**Dataset** data.ma01
- mice.1chain (Example 3), mice.impute.weighted.pmm (Example 1), ma.wtd.statNA (Example 1)

**Dataset** data.ma02
- fast.groupmean (Example 1),

**Dataset** data.ma03
- mice.impute.2l.eap (Example 1)

**Dataset** data.ma04
- mice.impute.2l.plausible.values (Example 1)

**Dataset** data.ma05
- mice.impute.2l.contextual.pmm (Example 1), mice.impute.2l.latentgroupmean (Example 1)
**data.smallscale**  

Small-Scale Dataset for Testing Purposes (Moderate Number of Cases, Many Variables)

**Description**

Small-scale dataset for testing purposes (moderate number of cases, many variables)

**Usage**

```r
data(data.smallscale)
```

**Format**

A data frame with 675 observations on the following 164 variables. The format is

```r
'data.frame': 675 obs. of 164 variables:
$ v1 : num 3 3 2 3 3 0 1 0 3 NA ... 
$ v2 : num 3 0 1 3 0 0 0 3 2 NA ... 
$ v3 : num 0 0 2 3 2 0 1 0 0 NA ... 
$ v4 : num 1 3 3 3 NA 0 0 0 3 NA ... 
$ v5 : num 0 0 3 3 0 0 3 1 3 3 ... 
$ v6 : num 8 8 9 8 9 9 9 8 9 9 ... 
[...]
```

**datalist2mids**  

Converting a List of Multiply Imputed Data Sets into a mids Object

**Description**

This function converts a list of multiply imputed data sets to a mids (mice package) object.

**Usage**

```r
datalist2mids(dat.list, progress = TRUE)
```

**Arguments**

- `dat.list`  
  List of multiply imputed data sets

- `progress`  
  An optional logical indicating whether conversion process be displayed

**Value**

An object of class mids
Author(s)
Alexander Robitzsch

See Also
See \texttt{as.mids} \texttt{(mice)} for converting a multiply imputed dataset in long format into a \texttt{mids} object.

Examples

### EXAMPLE 1: Imputation of NHANES data using Amelia

```r
library(mice)
library(Amelia)

data(nhanes, package = \"mice\")
set.seed(566)  # fix random seed

# impute 10 datasets using Amelia
a.out <- Amelia::amelia(x = nhanes, m = 10)
# plot of observed and imputed data
plot(a.out)

# convert list of multiply imputed datasets into a \texttt{mids} object
a.mids <- datalist2mids(a.out$imputations)

# linear regression: apply mice functionality \texttt{lm.mids}
mod <- with(a.mids, lm(bmi ~ age))
summary(pool(mod))
```

```
# (Intercept) 30.718881 2.22960 13.777753 12.581335 5.838025e-09 25.88578
# age -2.435746 1.08551 -2.243872 14.913153 4.043506e-02 -4.75038
# hi 95 nmis fmi lambda
# (Intercept) 35.5519834 NA 0.4013689 0.3132139
# age -0.1211117 0 0.3153636 0.2294162
```

## Not run:
# fit linear regression model in Zelig
library(Zelig)
mod2 <- Zelig::zelig(bmi ~ age, model = \"ls\", data = a.out$imputations, cite = FALSE)
summary(mod2)
```

```
# Model: ls
# Number of multiply imputed data sets: 10
# Combined results:
# Call:
# lm(formula = formula, weights = weights, model = F, data = data)
```
## Coefficients:

<table>
<thead>
<tr>
<th></th>
<th>Value</th>
<th>Std. Error</th>
<th>t-stat</th>
<th>p-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Intercept)</td>
<td>30.71881</td>
<td>2.22960</td>
<td>13.77753</td>
<td>4.603995e-24</td>
</tr>
<tr>
<td>age</td>
<td>-2.435746</td>
<td>1.08551</td>
<td>-2.243872</td>
<td>2.612377e-02</td>
</tr>
</tbody>
</table>

## For combined results from datasets i to j, use summary(x, subset = i:j).

## For separate results, use print(summary(x, subset = i:j)).

## End(Not run)

draw.pv.ctt

### Plausible Value Imputation Using a Known Measurement Error Variance (Based on Classical Test Theory)

#### Description
This function provides unidimensional plausible value imputation with a known measurement error variance or classical test theory (Mislevy, 1991). The reliability of the scale is estimated by Cronbach’s Alpha or can be provided by the user.

#### Usage
```r
draw.pv.ctt(y, dat.scale = NULL, x=NULL, samp.pars = TRUE,
             alpha = NULL, sig.e = NULL, var.e=NULL , true.var = NULL)
```

#### Arguments
- `y`: Vector of scale scores if `y` should not be used.
- `dat.scale`: Matrix of item responses
- `x`: Matrix of covariates
- `samp.pars`: An optional logical indicating whether scale parameters (reliability or measurement error standard deviation) should be sampled
- `alpha`: Reliability estimate of the scale. The default of `NULL` means that Cronbach’s alpha will be used as a reliability estimate.
- `sig.e`: Optional vector of the standard deviation of the error. Note that it is not the error variance.
- `var.e`: Optional vector of the variance of the error.
- `true.var`: True score variance

#### Details
The linear model is assumed for drawing plausible values of a variable $Y$ contaminated by measurement error. Assuming $Y = \theta + \epsilon$ and a linear regression model for $\theta$

$$
\theta = X \beta + \epsilon
$$

(plausible value) imputations from the posterior distribution $P(\theta|Y, X)$ are drawn. See Mislevy (1991) for details.
Value

A vector with plausible values

Note

Plausible value imputation is also labeled as multiple overimputation (Blackwell, Honaker & King, 2011).

Author(s)

Alexander Robitzsch

References


See Also

See also plausible.value.imputation.raschtype (sirt) for plausible value imputation.

Examples

```
# SIMULATED EXAMPLE 1: Scale scores

set.seed(899)
n <- 5000 # number of students
x <- round( runif( n , 0 ,1 ) )
y <- rnorm(n)
# simulate true score theta
theta <- .6 + .4*x + .5*y + rnorm(n)
# simulate observed score by adding measurement error
sig.e <- rep( sqrt(.40) , n )
theta.obs <- theta + rnorm( n , sd=sig.e)
# calculate alpha
( alpha <- var( theta ) / var( theta.obs ) )
# [1] 0.7424108
# => Ordinarily, sig.e or alpha will be known, assumed or estimated by using items,
#    replications or an appropriate measurement model.

# create matrix of predictors
X <- as.matrix( cbind(x , y ) )

# plausible value imputation with scale score
impl1 <- draw.pv.ctt( y=theta.obs , x = X , sig.e =sig.e )
# check results
```
lm( imp1 ~ x + y )

# imputation with alpha as an input
imp2 <- draw.pv.ctt( y=theta_obs , x = X , alpha = .74 )
lm( imp2 ~ x + y )

document content:

Description

Calculates some groupwise descriptive statistics

Usage

fast.groupmean(data, group, weights=NULL)

fast.groupsum(data, group, weights=NULL)

Arguments

data A numeric data frame

group A vector of group identifiers

weights An optional vector of sample weights

Value

A data frame with groupwise calculated statistics

Author(s)

Alexander Robitzsch

Examples

#############################################################################
# EXAMPLE 1: Group means data.ma02
#############################################################################

data( data.ma02 )
dat <- data.ma02[[1]] # select first dataset

# group means for read and math
fast.groupmean( dat[, c("read","math") ] , group=dat$idschool )
R Utilities: Vector Based Version of grep

description

This function imitates the usage of grep but it is extended to a vector argument.

Usage

grep.vec(pattern.vec, x, operator="AND")

Arguments

pattern.vec String which should be looked for in vector x
x A character vector
operator An optional string. The default argument "AND" searches all entries in x which contain all elements of pattern.vec. If operator is different from the default, then the "OR" logic applies, i.e. the functions searches for vector entries which contain at least one of the strings in pattern.vec.

Author(s)

Alexander Robitzsch

Examples

vec <- c("abcd", "bcde", "aedf", "cdf")
# search for entries in vec with contain 'a' and 'f'
# -> operator = "AND"
grep.vec( pattern.vec=c("a","f") , x=vec )
## $x
## [1] "aedf"
## $index.x
## [1] 3

# search for entries in vec which contain 'a' or 'f'
grep.vec( pattern.vec=c("a","f") , x=vec , operator="OR")
## $x
## [1] "abcd" "aedf" "cdf"
## $index.x
## [1] 1 3 4
Utilities: Include an Index to a Data Frame

Description

This function includes an index variable to a data frame in the first column.

Usage

index.dataframe(data,systime=FALSE)

Arguments

data Data frame
systime Should system time be included in the second column of the data frame?

Author(s)

Alexander Robitzsch

Examples

dfr <- matrix( 2*1:12-3 , 4,3 )
colnames(dfr) <- paste0("X",1:ncol(dfr))
index.dataframe( dfr )
## index X1 X2 X3
## 1 1 1 -1 7 15
## 2 2 1 9 17
## 3 3 3 11 19
## 4 4 5 13 21

index.dataframe( dfr , systime=TRUE)
## index file_created X1 X2 X3
## 1 1 2013-08-22 10:26:28 -1 7 15
## 2 2 2013-08-22 10:26:28 1 9 17
## 3 3 2013-08-22 10:26:28 3 11 19
## 4 4 2013-08-22 10:26:28 5 13 21

Kernel PLS Regression

Description

Fits a PLS regression model with the kernel algorithm (Dayal & Macgregor, 1997).
Usage

```
kernelpls.fit2(X, Y, ncomp)
```

`## S3 method for class 'kernelpls.fit2'
predict(object, X, ...)`

Arguments

- `X`: Matrix of regressors
- `Y`: Vector of a univariate outcome
- `ncomp`: Number of components to be extracted
- `object`: Object of class `kernelpls.fit2`
- `...`: Further arguments to be passed

Value

The same list as in `kernelpls.fit` of the `pls` package is produced.
In addition, $R^2$ measures are contained in R2.

Author(s)

Alexander Robitzsch

This code is a Rcpp translation of the original `kernelpls.fit` function from the `pls` package (see Mevik & Wehrens, 2007).

References


See Also

See the `pls` package for further estimation algorithms.

Examples

```
set.seed(789)
N <- 300  # number of cases
p <- 100  # number of predictors
rho1 <- .6 # correlations between predictors

# simulate data
```
load.data <- function(filename, type = "Rdata", path = getwd(), spss.default = TRUE, ...) 

load.data

**R Utilities: Loading/Reading Data Files using "miceadds"

**Description**

This function is a wrapper function for loading or reading data frames or matrices.

**Usage**

load.data( filename, type="Rdata", path=getwd(), spss.default=TRUE, ...)

**Arguments**

- **filename** Name of the data file (matrix or data frame). This can also be a part of the file name and the most recent file is loaded.
- **type** The type of file in which the data frame or matrix should be loaded. This can be Rdata (for R binary format, using load.Rdata2), csv (using read.csv), csv2 (using read.csv2), table (using read.table; the dataset must have the file extension dat or txt) or sav (using read.spss(foreign)).
- **path** Directory from which the dataset should be loaded
- **spss.default** Optional logical which is only applied for type="sav" indicating whether the arguments to.data.frame=TRUE and use.value.labels=FALSE are used.
Further arguments to be passed to `load.Rdata`, `read.csv2`, `read.csv`, `read.table` or `read.spss` (foreign).

**Author(s)**
Alexander Robitzsch

**See Also**
See also `load.Rdata` for loading R data frames.
See `save.Rdata` and `save.data` for saving/writing R data frames.

**Examples**
```r
## Not run:
# load a data frame in the file "data_s3.Rdata" and save this
# as the object "dat.s3"
dat.s3 <- load.data( filename = "data_s3.Rdata" , type = "Rdata" )

## End(Not run)
```

---

**load.Rdata**

**R Utilities: Loading Rdata Files in a Convenient Way**

**Description**
These functions loads a Rdata object saved as a data frame or a matrix in the current R environment. The function `load.Rdata` saves the loaded object in the global environment while `load.Rdata2` loads the object only specified environments. Hence, usage of `load.Rdata2` instead of `load.Rdata` is recommended.

**Usage**
```r
load.Rdata(filename, objname)
load.Rdata2(filename, path=getwd())
```

**Arguments**
- `filename` : Rdata file (matrix or data frame)
- `objname` : Object name. This object will be a global variable in R.
- `path` : Directory from which the dataset should be loaded

**Author(s)**
Alexander Robitzsch
**ma.scale2**

**Standardization of a Matrix**

**Description**

This function performs a z-standardization for a numeric matrix. Note that in a case of a zero standard deviation all matrix entries are divided by a small number such that no NaNs occur.

**Usage**

```r
ma.scale2(x, missings=FALSE)
```

**Arguments**

- `x` A numeric matrix in which missing values are permitted
- `missings` A logical indicating whether missings occur (or could occur) in the dataset

**Value**

A matrix

**Author(s)**

Alexander Robitzsch
Examples

```r
# EXAMPLE 1: z-standardization data.internet

data(data.internet)
dat <- data.internet

# z-standardize all variables in this dataset
zdat <- ma.scale2( dat , missings=TRUE )

## Not run:

# SIMULATED EXAMPLE 2: Speed comparison for many cases and many variables

set.seed(9786)
N <- 3000; p <- 200
x <- matrix(rnorm(N*p), N, p)
x <- round(x, 2)

# compare computation times for 10 replications
B <- 10
s1 <- Sys.time() # scale in R
for (bb in 1:B){
  res <- scale(x)
} ; s2 <- Sys.time(); d1 <- s2-s1

s1 <- Sys.time() # scale in miceadds
for (bb in 1:B){
  res1 <- ma.scale2(x)
} ; s2 <- Sys.time(); d2 <- s2-s1

# scale in miceadds with missing handling
s1 <- Sys.time()
for (bb in 1:B){
  res1 <- ma.scale2(x,missings=TRUE)
} ; s2 <- Sys.time(); d3 <- s2-s1
d1 # scale in R
d2 # scale in miceadds (no missing handling)
d3 # scale in miceadds (with missing handling)

## Time difference of 1.622431 secs
## Time difference of 0.156003 secs
## Time difference of 0.2028039 secs
```

## End(Not run)
Some Multivariate Descriptive Statistics for Weighted Data in miceadds

Description

Some multivariate descriptive statistics for weighted data in miceadds.

Usage

ma withheld meanNA(data, weights = rep(1, nrow(data)) )
ma withheld sdNA(data, weights = rep(1, nrow(data)) )
ma withheld covNA(data, weights = rep(1, nrow(data)) )
ma withheld corNA(data, weights = rep(1, nrow(data)) )

Arguments

data Numeric data frame
weights Optional vector of sampling weights

Details

Contrary to ordinary R practice, missing values are ignored in the calculation of descriptive statistics.

ma withheld meanNA weighted means
ma withheld sdNA weighted standard deviations
ma withheld covNA weighted covariance matrix
ma withheld corNA weighted correlation matrix

Value

A matrix or a vector depending on the requested statistic.

Author(s)

Alexander Robitzsch

Examples

```r
# EXAMPLE 1: Weighted statistics data.ma01

# weighted mean
```
mi.anova

Analysis of Variance for Multiply Imputed Data Sets (Using the $D_2$ Statistic)

Description

This function combines $F$ values from analysis of variance using the $D_2$ statistic which is based on combining $\chi^2$ statistics (see Allison, 2001; micombine.F, micombine.chisquare).

Usage

mi.anova(mi.res, formula, type=2)

Arguments

mi.res Object of class mids or mids.1chain
formula Formula for lm function. Note that this can be also a string.
type Type for ANOVA calculations. For type=3, the Anova function form the car package is used.

Value

A list with the following entries:

r.squared Explained variance $R^2$
anova.table ANOVA table

Author(s)

Alexander Robitzsch

References


See Also

micombine.F, micombine.chisquare
Examples

```r
library(mice)
library(car)
data(nhanes2, package="mice")
set.seed(9090)
# nhanes data in one chain and 8 imputed datasets
mi.res <- mice.1chain(nhanes2, burnin=4, iter=20, Nimp=8)
# 2-way analysis of variance (type 2)
an2a <- mi.anova(mi.res=mi.res, formula="bmi ~ age * chl")
# 2-way analysis of variance (type 3)
an2b <- mi.anova(mi.res=mi.res, formula="bmi ~ age * chl", type=3)

# ***** analysis based on first imputed dataset

# extract first dataset
dat1 <- complete(mi.res$mids)
# type 2 ANOVA
lm1 <- lm(bmi ~ age * chl, data = dat1)
summary(aov(lm1))
# type 3 ANOVA
lm2 <- lm(bmi ~ age * chl, data = dat1, contrasts=list(age=contr.sum))
car::Anova(lm2, type=3)
```

**mice.1chain**

Multiple Imputation by Chained Equations using One Chain

**Description**

This function modifies the `mice` function to multiply impute a dataset using a long chain instead of multiple parallel chains which is the approach employed in mice.

**Usage**

```r
mice.1chain(data, burnin = 10, iter = 20, Nimp = 10,
method = vector("character", length = ncol(data)),
predictorMatrix = (1 - diag(1, ncol(data))),
visitSequence = (1:ncol(data))[apply(is.na(data), 2, any)],
form = vector("character", length = ncol(data)),
post = vector("character", length = ncol(data)),
defaultMethod = c("pmm", "logreg", "polyreg", "polr"),
diagnostics = TRUE, printFlag = TRUE, seed = NA, imputationMethod = NULL,
defaultImputationMethod = NULL, data.init = NULL, ...)
```
## S3 method for class 'mids.1chain'

```r
summary(object,...)
```

## S3 method for class 'mids.1chain'

```r
plot(x,plot.burnin=FALSE, ask=TRUE, ...)
```

### Arguments

- **data**
  Numeric matrix
- **burnin**
  Number of burn-in iterations
- **iter**
  Total number of imputations (larger than `burnin`)
- **Nimp**
  Number of imputations
- **method**
  See `mice`
- **predictorMatrix**
  See `mice`
- **visitSequence**
  See `mice`
- **form**
  See `mice`
- **post**
  See `mice`
- **defaultMethod**
  See `mice`
- **diagnostics**
  See `mice`
- **printFlag**
  See `mice`
- **seed**
  See `mice`
- **imputationMethod**
  See `mice`
- **defaultImputationMethod**
  See `mice`
- **data.init**
  See `mice`
- **object**
  Object of class `mids.1chain`
- **x**
  Object of class `mids.1chain`
- **plot.burnin**
  An optional logical indicating whether burnin iterations should be included in the traceplot
- **ask**
  An optional logical indicating a user request for viewing next plot
- **...**
  See `mice`

### Value

A list with following entries:

- **midsobj**
  Objects of class `mids`
- **datlist**
  List of multiply imputed datasets
- **datalong**
  Original and imputed dataset in the long format
- **implist**
  List of `mids` objects for every imputation
- **chainMpar**
  Trace of means for all imputed variables
- **chainVarpar**
  Trace of variances for all imputed variables
Note

Multiple imputation can also be used for determining causal effects (see Example 3; Schafer & Kang, 2008).

Author(s)

Alexander Robitzsch

See Also

`mice`

Examples

```r
library(mice)
data(nhanes, package="mice")
set.seed(9090)

# nhanes data in one chain
imp.m1 <- mice.1chain(nhanes, burnin=5, iter=40, Nimp=4,
imputationMethod=rep("norm", 4))
summary(imp.m1)     # summary of mids.1chain
plot(imp.m1)        # trace plot excluding burnin iterations
plot(imp.m1, plot.burnin=TRUE) # trace plot including burnin iterations

# select mids object
imp.m2 <- imp.m1$midsobj
summary(imp.m2)     # summary of mids

# apply mice functionality lm.mids
mod <- with(imp.m2, lm(bmi ~ age))
summary(pool(mod))

# Not run:
library(mice)
data(nhanes2, package="mice")
set.seed(9090)

# nhanes2 data in one chain
imp.m1 <- mice.1chain(nhanes2, burnin=5, iter=25, Nimp=5)
# summary
summary(imp.m1$midsobj)
```
# EXAMPLE 3: Multiple imputation with counterfactuals for estimating causal effects (average treatment effects)
# Psychological Methods, 13, 279-313.

```
data(data.ma01)
dat <- data.ma01[, 4:11]

# define counterfactuals for reading score for students with and without migrational background
dat$read.migrant1 <- ifelse(paste(dat$migrant) == 1, dat$read, NA)
dat$read.migrant0 <- ifelse(paste(dat$migrant) == 0, dat$read, NA)

# define imputation method
impmethod <- rep("2l.pls2", ncol(dat))
names(impmethod) <- colnames(dat)

# define predictor matrix
pm <- 4*(1 - diag(ncol(dat))) # 4 - use all interactions
rownames(pm) <- colnames(pm) <- colnames(dat)
pm[c("read.migrant0", "read.migrant1"), ] <- 0
# do not use counterfactuals for 'read' as a predictor
pm[, "read.migrant0"] <- 0
pm[, "read.migrant1"] <- 0

# define control variables for creation of counterfactuals
pm[c("read.migrant0", "read.migrant1"), c("hisei","paredu","female","books") ] <- 4

# imputation using mice function and PLS imputation with predictive mean matching method 'pmm6'
imp <- mice( dat, imputationMethod=impmethod, predictorMatrix=pm, maxit=4, m=5, pls.impMethod="pmm5"
)

### Model 1: Raw score difference
mod1 <- with( imp, lm( read ~ migrant ) )
smod1 <- summary(pool(mod1))
```
mice.impute.2l.contextual.norm

Imputation by Normal Linear Regression with Contextual Variables

Description

This imputation method imputes a variable using linear regression with normally distributed residuals. Including a contextual effects means that an aggregated variable at a cluster level is included as a further covariate.
Usage

```r
mice.impute.2l.contextual.norm(y, ry, x, type, ridge = 10^(-5),
    imputationWeights = NULL, interactions = NULL, quadratics = NULL, ...)
```

Arguments

- `y`: Incomplete data vector of length \( n \)
- `ry`: Vector of missing data pattern (FALSE – missing, TRUE – observed)
- `x`: Matrix (\( n \times p \)) of complete covariates.
- `type`: Type of predictor variables. type\(-2\) refers to the cluster variable, type\(2\) denotes a variable for which also a contextual effect is included and type\(1\) denotes all other variables which are included as ‘ordinary’ predictors.
- `ridge`: Ridge parameter in the diagonal of \( X'X \)
- `imputationWeights`: Optional vector of sample weights
- `interactions`: Vector of variable names used for creating interactions
- `quadratics`: Vector of variable names used for creating quadratic terms
- `...`: Further arguments to be passed

Value

A vector of length \( n_{mis} = \text{sum}(\neg ry) \) with imputed values.

Author(s)

Alexander Robitzsch

See Also

For examples see `mice.impute.2l.contextual.pmm`.

---

**mice.impute.2l.contextual.pmm**

*Imputation by Predictive Mean Matching with Contextual Variables*

Description

This imputation method imputes a variable using linear regression with predictive mean matching as the imputation method. Including a contextual effects means that an aggregated variable at a cluster level is included as a further covariate.

Usage

```r
mice.impute.2l.contextual.pmm(y, ry, x, type, imputationWeights = NULL,
    interactions = NULL, quadratics = NULL, ...)
```
Arguments

- **y**: Incomplete data vector of length \( n \)
- **ry**: Vector of missing data pattern (FALSE – missing, TRUE – observed)
- **x**: Matrix \((n \times p)\) of complete covariates.
- **type**: Type of predictor variables. \( \text{type}=-2 \) refers to the cluster variable, \( \text{type}=2 \) denotes a variable for which also a contextual effect is included and \( \text{type}=1 \) denotes all other variables which are included as ‘ordinary’ predictors.
- **imputationWeights**: Optional vector of sample weights
- **interactions**: Vector of variable names used for creating interactions
- **quadratics**: Vector of variable names used for creating quadratic terms
- **...**: Further arguments to be passed

Value

A vector of length \( \text{nmis}=\sum(\neg \text{ry}) \) with imputed values.

Author(s)

Alexander Robitzsch

See Also

For imputations at level 2 variables see `mice.impute.2lonely.norm` (mice) and `mice.impute.2lonely.pmm` (mice).

Examples

```r
## Not run:
#########################################################################
# EXAMPLE 1: Sequential hierarchical imputation for data.ma05 dataset
#########################################################################

data(data.ma05)
dat <- data.ma05

# empty imputation
imp0 <- mice( dat , m=0 , maxit=0 )
summary(imp0)

# define predictor matrix
predM <- imp0$pred
# exclude student IDs
predM[, "idstud"] <- 0
# define idclass as the cluster variable (type=-2)
predM[, "idclass" ] <- -2

# define imputation methods
```
impMethod <- imp0$method
# initialisieren mit norm
impMethod <- rep("norm", length(impMethod))
names(impMethod) <- names(imp0$method)
impMethod[ c("idstud","idclass") ] <- ""

####
# STUDENT LEVEL (Level 1)

# Use a random slope model for Dscore and Mscore as the imputation method.
# Here, variance homogeneity of residuals is assumed (contrary to
# the 2l.norm imputation method in the mice package).
impMethod[ c("Dscore","Mscore") ] <- "2l.pan"
predM[ c("Dscore","Mscore"), "misei" ] <- 2 # random slopes on 'misei'
predM[ , "idclass" ] <- -2

# For imputing 'manote' and 'denote' use contextual effects (i.e. cluszer means)
# of variables 'misei' and 'migrant'
impMethod[ c("denote","manote") ] <- "2l.contextual.pmm"
predM[ c("denote","manote"), c("misei","migrant") ] <- 2

# Use no cluster variable 'idclass' for imputation of 'misei'
impMethod[ "misei"] <- "norm"
predM[ "misei","idclass" ] <- 0 # use no multilevel imputation model

# Variable migrant: contextual effects of Dscore and misei
impMethod[ "migrant"] <- "2l.contextual.pmm"
predM[ "migrant", c("Dscore","misei") ] <- 2
predM[ "migrant","idclass" ] <- -2

####
# CLASS LEVEL (Level 2)
# impute 'sprengel' and 'groesse' at the level of classes
impMethod[ "sprengel"] <- "2lonly.pmm"
impMethod[ "groesse"] <- "2lonly.norm"
predM[ c("sprengel","groesse"), "idclass" ] <- -2

# do imputation
imp <- mice( dat, predictorMatrix = predM, m = 3, maxit = 4,
imputationMethod = impMethod, paniter=100)
summary(imp)

## End(Not run)

---

**mice.impute.2l.eap**

*Imputation of a Variable with a Known Posterior Distribution*

---

**Description**

This function imputes values of a variable for which the mean and the standard deviation of the posterior distribution is known.
Usage

mice.impute.2l.eap(y, ry, x, eap, ...)

Arguments

y  Incomplete data vector of length n
ry  Vector of missing data pattern (FALSE – missing, TRUE – observed)
x  Matrix (n x p) of complete covariates.
eap  List with means and standard deviations of the posterior distribution (see Examples). If for multiple variables posterior distributions are known, then it is a list named in which each list entry is named according to the variable to be imputed and each list entry contains the variable’s EAP and standard deviation of the EAP.
...  Further arguments to be passed

Value

A vector of length nmiss=\sum(!ry) with imputed values.

Author(s)

Alexander Robitzsch

Examples

## Not run:
###############################################################################
# EXAMPLE 1: Imputation based on known posterior distribution
###############################################################################

data(data.ma03)
dat <- data.ma03

# define variable 'math_PV' as the plausible value imputation of math
dat$math_PV <- NA
vars <- colnames(dat)
dat1 <- as.matrix( dat[,vars] )

# define imputation methods
impmethod <- rep( "pmm" , length(vars ))
names(impmethod) <- vars
# define plausible value imputation based on EAP and SEEAP for 'math_PV'
impmethod[ "math_PV" ] <- "2l.eap"
eap <- list( "math_PV" = list( "M" = dat$math_EAP , "SE" = dat$math_SEEAP ) )
# define predictor matrix
pM <- 1 - diag(1,length(vars))
rownames(pM) <- colnames(pM) <- vars
pM[,c("idstud","math_EAP", "math_SEEAP") ] <- 0
    # remove some variables from imputation model

# imputation using three parallel chains
mice.impute.2l.latentgroupmean

Imputation of Latent and Manifest Group Means for Multilevel Data

Description

The imputation method 2l.latentgroupmean imputes a latent group mean assuming an infinite population of subjects within a group (see Luedtke et al., 2008 or Croon & van Veldhoven, 2007). Therefore, unreliability of group means when treating subjects as indicators is taken into account.

The imputation method mice.impute.2l.groupmean just imputes (i.e. computes) the manifest group mean. See also mice.impute.2lonely.mean (mice).

The imputation method mice.impute.2l.groupmean.elim computes the group mean eliminating the subject under study from the calculation. Therefore, this imputation method will lead to different values of individuals within the same group.

Usage

mice.impute.2l.latentgroupmean(y, ry, x, type, pls.facs = NULL,
imputationWeights = NULL, interactions = NULL, quadratics = NULL, ...)
mice.impute.2l.groupmean(y, ry, x, type, grmeanwarning = TRUE, ...)
mice.impute.2l.groupmean.elim(y, ry, x, type, ...)

Arguments

y Incomplete data vector of length n
ry Vector of missing data pattern (FALSE – missing, TRUE – observed)
x Matrix (n x p) of complete covariates.
type Type of predictor variables. type=-2 refers to the cluster variable, type=2 denotes a variable for which also a (latent) group mean should be calculated. Predictors with type=1 denote all other variables.
pls.facs Number of factors used for PLS regression (optional).
imputationWeights Optional vector of sample weights.
interactions  Vector of variable names used for creating interactions
quadratics   Vector of variable names used for creating quadratic terms
grmeanwarning An optional logical indicating whether some group means cannot be calculated.
...         Further arguments to be passed.

Details

The imputation of the latent group mean uses the \texttt{lmer} function of the \texttt{lme4} package. Latent group mean imputation also follows Mislevy (1991).

Value

A vector of length \(y\) containing imputed group means.

Author(s)

Alexander Robitzsch

References


Luedtke, O., Marsh, H. W., Robitzsch, A., Trautwein, U., Asparouhov, T., 


Examples

```r
## Not run:
#########################################################################
# EXAMPLE 1: Two-level imputation data.ma05 dataset with imputation
# of a latent group mean
#########################################################################

data(data.ma05)
dat <- data.ma05

# include manifest group mean for 'Mscore'
dat$M.Mscore <- NA
# include latent group group for 'Mscore'
dat$L.M.Mscore <- NA  # => LM: latent group mean

# empty imputation
imp <- mice( dat , m=0 , maxit=0 )
summary(imp)

# define predictor matrix
```r
predM <- imp$pred
# exclude student ISSs
predM[, "idstud"] <- 0
# idclass is the cluster identifier
predM[, "idclass"] <- -2

# define imputation methods
impMethod <- imp$method
# initialize with norm
impMethod <- rep("norm", length(impMethod))
names(impMethod) <- names(imp$method)
impMethod[ c("idstud","idclass") ] <- ""

#***
# STUDENT LEVEL (Level 1)

# Use a random slope model for Dscore and Mscore as the imputation method.
# Here, variance homogeneity of residuals is assumed (contrary to
# the 2l.norm imputation method in the mice package).
impMethod[ c("Dscore", "Mscore") ] <- "2l.pan"
predM[ c("Dscore","Mscore"), "misei" ] <- 2 # random slopes on 'misei'
predM[, "idclass"] <- -2

# For imputing 'manote' and 'denote' use contextual effects (i.e. cluster means)
# of variables 'misei' and 'migrant'
impMethod[ c("denote" , "manote") ] <- "2l.contextual.pmm"
predM[ c("denote" , "manote"), c("misei","migrant") ] <- 2

# Use no cluster variable 'idclass' for imputation of 'misei'
impMethod[ "misei"] <- "norm"
predM[ "misei" , "idclass"] <- 0 # use no multilevel imputation model

# Variable migrant: contextual effects of Dscore and misei
impMethod[ "migrant"] <- "2l.contextual.pmm"
predM[ "migrant" , c("Dscore", "misei") ] <- 2
predM[ "migrant" , "idclass"] <- -2

#***
# CLASS LEVEL (Level 2)
# impute 'sprengel' and 'groesse' at the level of classes
impMethod[ "sprengel"] <- "2lonly.pmm2"
impMethod[ "groesse"] <- "2lonly.norm2"
predM[ c("sprengel","groesse"), "idclass"] <- -2

# manifest group mean for Mscore
impMethod[ "M.Mscore" ] <- "2l.groupmean"
# latent group mean for Mscore
impMethod[ "LM.Mscore" ] <- "2l.latentgroupmean"
predM[ "M.Mscore" , "Mscore"] <- 2

# covariates for latent group mean of 'Mscore'
predM[ "LM.Mscore" , "Mscore"] <- 2
predM[ "LM.Mscore" , c("Dscore","sprengel") ] <- 1
```
# do imputations
imp <- mice( dat , predictorMatrix = predM , m =3 , maxit = 4 ,
imputationMethod = impMethod , allow.na = TRUE , pan.iter=100)

## End(Not run)

mice.impute.2l.plausible.values

Plausible Value Imputation using Classical Test Theory and Based on Individual Likelihood

Description

This imputation function performs unidimensional plausible value imputation if (subject-wise) measurement errors or the reliability of the scale is known (Mislevy, 1991; see also Asparouhov & Muthen, 2010; Blackwell, Honaker & King, 2011). The function also allows the input of an individual likelihood obtained by fitting an item response model.

Usage

mice.impute.2l.plausible.values(y, ry, x, type, alpha = NULL,
alpha.se = 0, scale.values = NULL, sig.e.miss = 1e+06,
like=NULL , theta=NULL , normal.approx=NULL ,
pviter = 15, imputationWeights = rep(1, length(y)), plausible.value.print = TRUE,
pls.facs = NULL, interactions = NULL,
quadratics = NULL, ...)

Arguments

y
Incomplete data vector of length n
ry
Vector of missing data pattern (FALSE – missing, TRUE – observed)
x
Matrix (n × p) of complete covariates.
type
Type of predictor variables. type=3 refers to items belonging to a scale to be imputed. A cluster (grouping) variable is defined by type=-2. If for some predictors, the cluster means should also be included as predictors, then specify type=2 (see Imputation Model 3 of Example 1).
alpa
A known reliability estimate. An optional standard error of the estimate can be provided in alpha.se
alpha.se
Optional numeric value of the standard error of the alpha reliability estimate if in every iteration a new reliability should be sampled.
scale.values
A list consisting of scale values of scale values and its corresponding standard errors (see Example 1).
sig.e.miss
A standard error of measurement for cases with missing values on a scale
like
Individual likelihood evaluated at theta
theta Grid of unidimensional latent variable
normal.approx Logical indicating whether the individual posterior should be approximated by a normal distribution
pviter Number of iterations in each imputation which should be run until the plausible values are drawn
imputationWeights Optional vector of sample weights
plausible.value.print An optional logical indicating whether some information about the plausible value imputation should be printed at the console
pls.facs Number of PLS factors if PLS dimension reduction is used
interactions Vector of variable names used for creating interactions
quadratics Vector of variable names used for creating quadratic terms
... Further objects to be passed

Details

The linear model is assumed for drawing plausible values of a variable $Y$ contaminated by measurement error. Assuming $Y = \theta + e$ and a linear regression model for $\theta$

$$\theta = X \beta + \epsilon$$

(plausible value) imputations from the posterior distribution $P(\theta|Y, X)$ are drawn. See Mislevy (1991) for details.

Value

A vector of length nrow(x) containing imputed plausible values.

Author(s)

Alexander Robitzsch

References


Examples

```r
### Not run:
# # EXAMPLE 1: Plausible value imputation for data.ma04 | 2 scales
# 
# data(data.ma04)
dat <- data.ma04

# Scale 1 consists of items A1,...,A4
# Scale 2 consists of items B1,...,B5
data$scale1 <- NA
data$scale2 <- NA

# empty imputation
imp <- mice( dat , m=0 , maxit=0 )
summary(imp)

# define predictors
predM <- imp$pred

# define imputation methods
impMethod <- imp$method
impMethod <- rep("norm", length(impMethod) )
names(impMethod) <- names( imp$method )

# look at missing proportions
colSums( is.na(dat) )

# redefine imputation methods for plausible value imputation
impMethod[ "scale1" ] <- "2l.plausible.values"
predM[ "scale1" , ] <- 1
predM[ "scale1" , c("A1","A2","A3","A4") ] <- 3
# items corresponding to a scale should be declared by a 3 in the predictor matrix
impMethod[ "scale2" ] <- "2l.plausible.values"
predM[ "scale2" ] <- 0
predM[ "scale2" , c("A2","A3","A4","V6","V7") ] <- 1
diag(predM) <- 0

# use imputed scale values as predictors for V5, V6 and V7
predM[c("V5","V6","V7") , c("scale1","scale2") ] <- 1
# exclude for V5, V6 and V7 the items of scales A and B as predictors
predM[c("V5","V6","V7") , c( paste0("A",2:4) , paste0("B",1:5) ) ] <- 0
# exclude 'group' as a predictor
predM[,"group"] <- 0

# look at imputation method and predictor matrix
impMethod
predM
```

```r
# Parameter for imputation
###
```
\# scale 1 (A1,\ldots,A4)
\# known Cronbach's Alpha
alpha <- NULL
alpha <- list( "scale1" = .8 )
alpha.se <- list( "scale1" = .05 )  # sample alpha with a standard deviation of .05

***
\# scale 2 (B1,\ldots,B5)
\# means and SE's of scale scores are assumed to be known
M.scale2 <- rowMeans( dat[, paste("B","1:5",sep="" ) ] )
# M.scale2[ is.na( m1 ) ] <- mean( M.scale2 , na.rm=TRUE )
SE.scale2 <- rep( sqrt( var(M.scale2,na.rm=T)*(1-.8) ) , nrow(dat) )
# => heterogeneous measurement errors are allowed
scale.values <- list( "scale2" = list( "M" = M.scale2 , "SE" = SE.scale2 ) )

*** Imputation Model 1: Imputation four using parallel chains
imp1 <- mice( dat , predictorMatrix = predM , m = 4 , maxit = 5 ,
    alpha.se = alpha.se , imputationMethod = impMethod , allow.na = TRUE , alpha = alpha,
    scale.values = scale.values )
summary(imp1)

# extract first imputed dataset
dat11 <- complete( imp , 1 )

*** Imputation Model 2: Imputation using one long chain
imp2 <- mice1chain( dat , predictorMatrix = predM , burnin=10 , iter=20 , Nimp=4 ,
    alpha.se = alpha.se , imputationMethod = impMethod , allow.na = TRUE , alpha = alpha,
    scale.values = scale.values )
summary(imp2)

-------------

*** Imputation Model 3: Imputation including group level variables

# use group indicator for plausible value estimation
predM[ "scale1" , "group" ] <- -2
# V7 and B1 should be aggregated at the group level
predM[ "scale1" , c("V7","B1") ] <- -2
predM[ "scale2" , "group" ] <- -2
predM[ "scale2" , c("V7","A1") ] <- -2

# perform single imputation (m=1)
imp <- mice( dat , predictorMatrix = predM , m = 1 , maxit=10 ,
    imputationMethod = impMethod , allow.na = TRUE , alpha = alpha,
    scale.values = scale.values )
dat10 <- complete(imp)

# multilevel model
library(lme4)
mod <- lmer( scale1 ~ ( 1 | group ) , data = dat11 )
summary(mod)

mod <- lmer( scale1 ~ ( 1 | group ) , data = dat10)
summary(mod)
# SIMULATED EXAMPLE 2: Plausible value imputation with chained equations

# - simulate a latent variable theta and dichotomous item responses
# - two covariates X in which the second covariate has measurement error

library(sirt)
library(TAM)
library(lavaan)

set.seed(7756)
N <- 2000  # number of persons
I <- 10    # number of items

# simulate covariates
X <- mvrnorm(N, mu = c(0, 0), Sigma = matrix(c(1, .5, .5, 1), 2, 2))
colnames(X) <- paste0("X", 1:2)
# second covariate with measurement error with variance var.err
var.err <- .3
X.err <- X
X.err[, 2] <- X[, 2] + rnorm(N, sd = sqrt(var.err))
# simulate theta
theta <- .5 * X[, 1] + .4 * X[, 2] + rnorm(N, sd = .5)
# simulate item responses
itemdiff <- seq(-2, 2, length=I)  # item difficulties
dat <- sirt::sim.raschtype(theta, b = itemdiff)

#***************
#*** Model 0: Regression model with true variables
mod0 <- lm(theta ~ X)
summary(mod0)

#***************
# plausible value imputation for abilities and error-prone
# covariates using the mice package

# creating the likelihood for plausible value for abilities
mod11 <- TAM::tam.mml(dat)
likePV <- IRT.likelihood(mod11)
# creating the likelihood for error-prone covariate X2
# The known measurement error variance is 0.3.
lavmodel <- "
  X2true <- 1*X2
  X2 <- 0.3*X2"
mod12 <- lavaan::cfa(lavmodel, data = as.data.frame(X.err))
summary(mod12)
likeX2 <- IRTLikelihood.cfa(data = X.err, cfaobj=mod12)
str(likeX2)

#-- create data input for mice package
mice.impute.2l.pls2

Imputation using Partial Least Squares for Dimension Reduction

Description

This function imputes a variable with missing values using PLS regression (Mevik & Wehrens, 2007) for a dimension reduction of the predictor space.

Usage

mice.impute.2l.pls2(y, ry, x, type, pls.facs = NULL,
  pls.impMethod = "pmm", pls.print.progress = TRUE,
  imputationWeights = rep(1, length(y)), pcamaxcols = 1E+09,
  tricube.pmm.scale = NULL, min.int.cor = 0, min.all.cor=0,
  N.largest = 0, pls.title = NULL, print.dims = TRUE,
  pls.maxcols=5000 , ...)

mice.impute.2l.pls(y, ry, x, type, pls.facs = NULL,
pls.impMethod = "tricube.pmm2", pls.method = NULL,
pls.print.progress = TRUE, imputationWeights = rep(1, length(y)),
pcamaxcols = 1E+09, tricube.pmm.scale = NULL, min.int.cor = 0, min.all.cor = 0,
N.largest = 0, pls.title = NULL, print.dims = TRUE, ...)

Arguments

y
Incomplete data vector of length \( n \)

ry
Vector of missing data pattern (FALSE – missing, TRUE – observed)

x
Matrix \((n x p)\) of complete covariates.

type
type=1 – variable is used as a predictor,
type=4 – create interactions with the specified variable with all other predictors,
type=5 – create a quadratic term of the specified variable

type=6 – if some interactions are specified, ignore the variables with entry 6
when creating interactions

type=2 – specification of a cluster variable. The cluster mean of the outcome \( y \) (when eliminating the subject under study) is included as a further predictor in the imputation.

pls.facs
Number of factors used in PLS regression. This argument can also be specified
as a list defining different numbers of factors for all variables to be imputed.

pls.impMethod
Imputation method based in the PLS regression model:

norm – normal linear regression
pmm – predictive mean matching (pmm method from mice)

pmm5 – predictive mean matching (pmm5 method from miceadds)

tricube.pmm/tricube.pmm2 – predictive mean matching with tricube kernel

xpls facs – create only PLS factors of the regression model

pls.method
Calculation method of PLS regression. See pls (pls) for more details.

pls.print.progress
Print progress during PLS regression.

imputationWeights
Vector of sample weights to be used in imputation models.

pcamaxcols
Maximum number of principal components.

tricube.pmm.scale
Scale factor for tricube predictive mean matching.

min.int.cor
Minimum absolute correlation for an interaction of two predictors to be included
in the PLS regression model

min.all.cor
Minimum absolute correlation for inclusion in the PLS regression model.

N.largest
Number of variable to be included which do have the largest absolute correla-
tions.

pls.title
Title for progress print in console output.

print.dims
An optional logical indicating whether dimensions of inputs should be printed.

pls.maxcols
Maximum number of interactions to be created.

... Further arguments to be passed.
Details

The function mice.impute.2l.pls2 uses kernelpls.fit2 instead of kernelpls.fit from the pls package and is a bit faster.

Value

A vector of length nmis=sum(!is.na(y)) with imputations if pls.impMethod != "xplsfacs". In case of pls.impMethod == "xplsfacs" a matrix with PLS factors is computed.

Author(s)

Alexander Robitzsch

References


Examples

```r
## Not run:
########################################################################
# EXAMPLE 1: PLS imputation method for internet data
########################################################################

data(data.internet)
dat <- data.internet

# specify predictor matrix
predictorMatrix <- matrix(1, ncol(dat), ncol(dat))
rownames(predictorMatrix) <- colnames(predictorMatrix) <- colnames(dat)
diag(predictorMatrix) <- 0

# use PLS imputation method for all variables
impMethod <- rep("2l.pls2", ncol(dat))
names(impMethod) <- colnames(dat)

# define predictors for interactions (entries with type 4 in predictorMatrix)
predictorMatrix[c("IN1","IN15","IN16"),c("IN1","IN3","IN10","IN13")]<-4
# define predictors which should appear as linear and quadratic terms (type 5)
predictorMatrix[c("IN1","IN8","IN9","IN10","IN11"),c("IN1","IN2","IN7","IN5")]<-5

# use 9 PLS factors for all variables
pls.facs <- as.list(rep(9, length(impMethod)))
names(pls.facs) <- names(impMethod)
pls.facs$IN1 <- 15  # use 15 PLS factors for variable IN1

# choose norm or pmm imputation method
pls.impMethod <- as.list(rep("norm", length(impMethod)))
names(pls.impMethod) <- names(impMethod)
pls.impMethod[c("IN1","IN6")]<-"pmm5"
```
mice.impute.2lonly.pmm2

---

**Description**

These functions are just bug fixes of `mice.impute.2lonly.pmm` and `mice.impute.2lonly.norm` in `mice` (version 2.21).

**Usage**

```r
mice.impute.2lonly.pmm2(y, ry, x, type, ...)  
mice.impute.2lonly.norm2(y, ry, x, type, ...)
```

**Arguments**

- `y`  
  Incomplete data vector of length `n`
- `ry`  
  Vector of missing data pattern (FALSE=missing, TRUE=observed)
- `x`  
  Matrix (`n` x `p`) of complete covariates. Only numeric variables are permitted for usage of this function.
- `type`  
  Group identifier must be specified by `-2`. Predictors must be specified by `1`.
- `...`  
  Other named arguments.

**Value**

A vector of length `n` with imputations.

**Author(s)**

Alexander Robitzsch
mice.impute.pmm3

See Also

mice.impute.2lonly.pmm(mice), mice.impute.2lonly.norm(mice)

Description

This function imputes values by predictive mean matching like the pmm method in the mice package.

Usage

mice.impute.pmm3(y, ry, x, donors = 3, noise = 10^5, ridge = 10^(-5), ...)  
mice.impute.pmm4(y, ry, x, donors = 3, noise = 10^5, ridge = 10^(-5), ...)  
mice.impute.pmm5(y, ry, x, donors = 3, noise = 10^5, ridge = 10^(-5), ...)  
mice.impute.pmm6(y, ry, x, donors = 3, noise = 10^5, ridge = 10^(-5), ...)

Arguments

y        Incomplete data vector of length n  
ry       Vector of missing data pattern (FALSE – missing, TRUE – observed)  
x        Matrix (n x p) of complete covariates.  
donors   Number of donors used for imputation  
noise    Numerical value to break ties  
ridge    Ridge parameter in the diagonal of $X'X$  
...      Further arguments to be passed

Details

The imputation method pmm3 imitates pmm in mice.

The imputation method pmm4 ignores ties in predicted y values. With many predictors, this does not probably implies any substantial problem.

The imputation method pmm5 suffers from the same problem. Contrary to the other PMM methods, it searches D donors (specified by donors) smaller than the predicted value and D donors larger than the predicted value and randomly samples a value from this set of 2 · D donors.

The imputation method pmm6 is just the Rcpp implementation of pmm5.

Value

A vector of length nmis=sum(!ry) with imputed values.

Author(s)

Alexander Robitzsch
See Also

See [data.largescale](#) and [data.smallscale](#) for speed comparisons of different functions for predictive mean matching.

Examples

```r
## Not run:
# SIMULATED EXAMPLE 1: Two variables x and y with missing y
# simulate response tendency z and missings on y
set.seed(1413)

rho <- .6  # correlation between x and y
N <- 6800   # number of cases
x <- rnorm(N)
My <- .35  # mean of y
y.com <- y <- My + rho * x + rnorm(N, sd = sqrt(1 - rho^2))

# create missingness on y depending on rho.MAR parameter
rho.mar <- .4  # correlation response tendency z and x
missrate <- .25  # missing response rate
# simulate response tendency z and missings on y
z <- rho.mar * x + rnorm(N, sd = sqrt(1 - rho.mar^2))
y[ z < qnorm( missrate ) ] <- NA
dat <- data.frame(x, y)

data imputation
impmethod <- rep("pmm", 2)
names(impmethod) <- colnames(dat)

# pmm (in mice)
imp1 <- mice( as.matrix(dat), m=1, maxit=1, imputationMethod=impmethod)
# pmm3 (in miceadds)
imp3 <- mice( as.matrix(dat), m=1, maxit=1, imputationMethod=gsub("pmm","pmm3",impmethod) )
# pmm4 (in miceadds)
imp4 <- mice( as.matrix(dat), m=1, maxit=1, imputationMethod=gsub("pmm","pmm4",impmethod) )
# pmm5 (in miceadds)
imp5 <- mice( as.matrix(dat), m=1, maxit=1, imputationMethod=gsub("pmm","pmm5",impmethod) )
# pmm6 (in miceadds)
imp6 <- mice( as.matrix(dat), m=1, maxit=1, imputationMethod=gsub("pmm","pmm6",impmethod) )
dat.imp1 <- complete( imp1 , 1)
dat.imp3 <- complete( imp3 , 1)
dat.imp4 <- complete( imp4 , 1)
dat.imp5 <- complete( imp5 , 1)
dat.imp6 <- complete( imp6 , 1)
dfr <- NULL
```
# means
dfr <- rbind( dfr , c( mean( y.com ) , mean( y , na.rm=TRUE ) , mean( dat.imp1$y) ,
    mean( dat.imp3$y) , mean( dat.imp4$y) , mean( dat.imp5$y) , mean( dat.imp6$y) ) )
# SD
dfr <- rbind( dfr , c( sd( y.com ) , sd( y , na.rm=TRUE ) , sd( dat.imp1$y) ,
    sd( dat.imp3$y) , sd( dat.imp4$y) , sd( dat.imp5$y) , sd( dat.imp6$y) ) )
# correlations
dfr <- rbind( dfr , c( cor( x,y.com ) , cor( x[ ! is.na(y) ] , y[ ! is.na(y) ] ) ,
    cor( dat.imp1$x , dat.imp1$y) , cor( dat.imp3$x , dat.imp3$y) ,
    cor( dat.imp4$x , dat.imp4$y) , cor( dat.imp5$x , dat.imp5$y) ,
    cor( dat.imp6$x , dat.imp6$y) ) )
rownames(dfr) <- c("M_y", "SD_y", "cor_xy")
colnames(dfr) <- c("compl", "ld", "pmm", "pmm3", "pmm4", "pmm5","pmm6")
## compl  ld  pmm  pmm3  pmm4  pmm5  pmm6
## M_y  0.3306 0.4282 0.3144 0.3234 0.3223 0.3264 0.3310
## SD_y  0.9910 0.9801 0.9873 0.9887 0.9891 0.9882 0.9877
## cor_xy  0.6057 0.5950 0.6072 0.6021 0.6100 0.6057 0.6069
## End(Not run)

mice.impute.tricube.pmm

Imputation by Tricube Predictive Mean Matching

Description
This function performs tricube predictive mean matching (see http://www.rdocumentation.org/
packages/Hmisc/functions/aregImpute) in which donors are weighted according to distances
of predicted values.

Usage
mice.impute.tricube.pmm(y, ry, x, tricube.pmm.scale = 0.2, tricube.boot = FALSE, ...)
mice.impute.tricube.pmm2(y, ry, x, tricube.pmm.scale = 0.2, tricube.boot = FALSE, ...)

Arguments

y                  Incomplete data vector of length n
ry                 Vector of missing data pattern (FALSE – missing, TRUE – observed)
x                  Matrix (n x p) of complete covariates.
tricube.pmm.scale
A scaling factor for tricube matching. The default is 0.2.
tricube.boot       A logical indicating whether tricube matching should be performed using a boot-
                   strap sample
...                Further arguments to be passed
Value

A vector of length \( \text{nmis} = \text{sum}(!\text{ry}) \) with imputed values.

Note

The imputation method `tricube.pmm2` is ordinarily somewhat faster than `tricube.pmm`.

Author(s)

Alexander Robitzsch

See Also

http://www.rdocumentation.org/packages/Hmisc/functions/aregImpute

Examples

```r
## Not run:
# Example 1: Tricube predictive mean matching for nhanes data
library(mice)
data(nhanes, package="mice")
set.seed(9090)

### Model 1: Use default of tricube predictive mean matching
varnames <- colnames(nhanes)
Vv <- length(varnames)
imputationMethod <- rep("tricube.pmm2", Vv)
names(imputationMethod) <- varnames
# imputation with mice
imp.ml1 <- mice(nhanes, m=5, maxit=4, imputationMethod=imputationMethod)

### Model 2: use item-specific imputation methods
iM2 <- imputationMethod
iM2["bmi"] <- "pmm6"
# use tricube.pmm2 for hyp and chl
# select different scale parameters for these variables
tricube.pmm.scale1 <- list( "hyp" = .15, "chl" = .30)
imp.ml2 <- mice.lchain( nhanes, burnin=5, iter=20, Nimp=4, imputationMethod= iM2, tricube.pmm.scale=tricube.pmm.scale1)

## End(Not run)
```
mice.impute.weighted.norm

Imputation by a Weighted Linear Normal Regression

Description

Imputation by a weighted linear normal regression.

Usage

mice.impute.weighted.norm(y, ry, x, ridge = 1e-05, pls.facs = NULL,
imputationWeights = NULL, interactions = NULL, quadratics = NULL, ...)

Arguments

y  Incomplete data vector of length n
ry  Vector of missing data pattern (FALSE – missing, TRUE – observed)
x  Matrix (n x p) of complete covariates.
ridge  Ridge parameter in the diagonal of \( X'X \)
imputationWeights  Optional vector of sampling weights
pls.facs  Number of factors in PLS regression (if used). The default is NULL which means that no PLS regression is used for dimension reduction.
interactions  Optional vector of variables for which interactions should be created
quadratics  Optional vector of variables which should also be included as quadratic effects.
...  Further arguments to be passed

Value

A vector of length \( n_{mis} = \text{sum}(\neg ry) \) with imputed values.

Author(s)

Alexander Robitzsch

See Also

For examples see mice.impute.weighted.pmm
**mice.impute.weighted.pmm**

*Imputation by Weighted Predictive Mean Matching*

**Description**

Imputation by predictive mean matching using sampling weights.

**Usage**

```r
mice.impute.weighted.pmm(y, ry, x, imputationWeights = NULL,
                  pls.facs = NULL, interactions = NULL, quadratics = NULL, ...)
```

**Arguments**

- `y` Incomplete data vector of length \( n \)
- `ry` Vector of missing data pattern (FALSE – missing, TRUE – observed)
- `x` Matrix \((n \times p)\) of complete covariates.
- `imputationWeights` Optional vector of sampling weights
- `pls.facs` Number of factors in PLS regression (if used). The default is NULL which means that no PLS regression is used for dimension reduction.
- `interactions` Optional vector of variables for which interactions should be created
- `quadratics` Optional vector of variables which should also be included as quadratic effects.
- `...` Further arguments to be passed

**Value**

A vector of length \( nmis = \text{sum}(!ry) \) with imputed values.

**Author(s)**

Alexander Robitzsch

**See Also**

For imputation with the linear normal regression and sampling weights see `mice.impute.weighted.norm`.

**Examples**

```r
## Not run:
# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# EXAMPLE 1: Imputation using sample weights
# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

data(data.ma01)
```
micombine.chisquare

Combination of Chi Square Statistics of Multiply Imputed Datasets

Description

This function does inference for the $\chi^2$ statistic based on multiply imputed datasets (see e.g. Enders, 2010, p. 239 ff.; Allison, 2002). This function is also denoted as the $D_2$ statistic.

Usage

micombine.chisquare(dk, df, display = TRUE)

Arguments

dk Vector of chi square statistics
df Degrees of freedom of $\chi^2$ statistic
display An optional logical indicating whether results should be printed at the R console.

Value

A vector with following entries

D Combined $D_2$ statistic which is approximately $F$-distributed with (df, df2) degrees of freedom
p The p value corresponding to D
df Denominator degrees of freedom
**micombine.cor**

<table>
<thead>
<tr>
<th>df2</th>
<th>Numerator degrees of freedom</th>
</tr>
</thead>
<tbody>
<tr>
<td>chisq.approx</td>
<td>Chi square approximation of the $D_2$ statistic</td>
</tr>
<tr>
<td>p.approx</td>
<td>The p value corresponding to the $D_2$ statistic</td>
</tr>
</tbody>
</table>

**Author(s)**

Alexander Robitzsch

**References**


**See Also**

See also `pool.compare` (*mice*) for a Wald test to compare two fitted models in the *mice* package.

**Examples**

```r
# Example 1: Chi square values of analyses from 7 multiply imputed datasets
dk.comb <- micombine.chisquare(dk, df=4)

# Vector of 7 chi square statistics
# Combination of Chi Square Statistics for Multiply Imputed Data
# Using 7 Imputed Data Sets
# F(4,594.01)=4.486   p=0.00141
# Chi Square Approximation Chi2(4)=17.946   p=0.00126
```

---

**micombine.cor**  
*Combination of Correlations for Multiply Imputed Data Sets*

**Description**

Statistical inference for correlation coefficients for multiply imputed datasets

**Usage**

```r
micombine.cor(mi.res, variables = 1:(ncol(mi.list[[1]])), conf.level = 0.95)
```

**Arguments**

- `mi.res`  
  Object of class `mids` or `mids.1chain`
- `variables`  
  Indices of variables for selection
- `conf.level`  
  Confidence level
**Value**

A data frame containing the correlation coefficient \((r)\) and its corresponding standard error \((rse)\), fraction of missing information \((fmi)\) and a \(t\) value \((t)\).

**Author(s)**

Alexander Robitzsch

**Examples**

```r
## Not run:

library(mice)
data(nhanes, package="mice")
set.seed(9090)

# nhanes data in one chain
imp.mi <- mice1chain(nhanes, burnin=5, iter=20, Nimp=4, 
imputationMethod=rep("norm", 4))

# correlation coefficients of variables 2 and 3 (indexed in nhanes data)
res <- micombine.cor(mi.res=imp.mi, variables = c(2,3))

# variable1 variable2  r     rse  fisher_r  fisher_rse  fmi    t    p
# 1   chl  bmi  0.2458 0.2236 0.2510 0.2540 0.3246 0.9879 0.3232
# 2   chl  hyp  0.2286 0.2152 0.2327 0.2413 0.2377 0.9643 0.3349
# 3   bmi  hyp -0.0084 0.2198 -0.0084 0.2351 0.1904 -0.0358 0.9714

## End(Not run)
```

---

**micombine.F**

*Combination of \(F\) Statistics for Multiply Imputed Datasets Using a Chi Square Approximation*

**Description**

Several \(F\) statistics from multiply imputed datasets are combined using an approximation based on \(\chi^2\) statistics (see *micombine.chisquare*).

**Usage**

```
micombine.F(Fvalues, df1, display = TRUE)
```
Arguments

- **Fvalues**: Vector containing F values.
- **df1**: Degrees of freedom of the denominator. Degrees of freedom of the numerator are approximated by \( \infty \) (large number of degrees of freedom).
- **display**: A logical indicating whether results should be displayed at the console.

Value

The same output as in `micombine.chisquare`

Author(s)

Alexander Robitzsch

See Also

- `micombine.chisquare`

Examples

```r
# EXAMPLE 1: F statistics for 5 imputed datasets
Fvalues <- c(6.76, 4.54, 4.23, 5.45, 4.78)
micombine.F(Fvalues, df1=4)
## Combination of Chi Square Statistics for Multiply Imputed Data
## Using 5 Imputed Data Sets
## F(4,67.11)=4.097  p=0.00497
## Chi Square Approximation Chi2(4)=16.387  p=0.00254
```

---

### mids2datlist

*Converting a mids or mids.chain Object in a Dataset List*

**Description**

Converts a mids or mids.chain object in a dataset list.

**Usage**

`mids2datlist(midsobj)`

**Arguments**

- **midsobj**: Object of class mids or mids.chain
Value
List of multiply imputed datasets

Author(s)
Alexander Robitzsch

Examples

```r
# Not run:
# Example 1: Imputing nhanes data and convert result into a dataset list

data(nhanes, package = "mice")

#***** imputation using mice
imp1 <- mice(nhanes, m = 3, maxit = 5)
# convert mids object into list
datlist1 <- mids2datlist(imp1)

#***** imputation using mice.1chain
imp2 <- mice.1chain(nhanes, burnin = 4, iter = 20, Nimp = 5)
# convert mids.1chain object into list
datlist2 <- mids2datlist(imp2)

# End(Not run)
```

---

**output.format1**  

*R Utilities: Formatting R Output on the R Console*

Description
This function does some formatting of output.

Usage

```r
output.format1(stringtype, label, rep.N = 1, stringlength = 70)
```

Arguments

- `stringtype`  
  Type of string for display, e.g. "*", "-", ...

- `label`  
  Some comment which should be displayed at the console

- `rep.N`  
  Number of lines which shall be left blank

- `stringlength`  
  Length of vector with `label`
Value
Generates a string output at the R console

Author(s)
Alexander Robitzsch

Examples
```
outputNformat1Hstringtype"#" , label="HELLO WORLD" , stringlength = 20)
## '##'##'##'##'##'##'##'##'##'##'##'##'##'##'##'##'##'##'##'##'##'##'##'##'
## HELLO WORLD
```

Description
Performs a principal component analysis for a dataset while a ridge parameter is added on the diagonal of the covariance matrix.

Usage
```
pca.covridge(x, ridge = 10^(-10))
```

Arguments
- **x**: A numeric matrix
- **ridge**: Ridge regularization parameter for the covariance matrix

Value
A list with following entries:
- **loadings**: Matrix of factor loadings
- **scores**: Matrix of principal component scores
- **sdev**: Vector of standard deviations of factors (square root of eigenvalues)

Author(s)
Alexander Robitzsch

See Also
Principal component analysis in stats: *princomp*
For calculating first eigenvalues of a symmetric matrix see also *eigenvalues.sirt* in the *sirt* package.
Examples

```r
# Not run:
# REVAL: Evaluates a String as an Expression in R

### EXAMPLE 1: PCA on imputed internet data

library(mice)
data(data.internet)
dat <- as.matrix(data.internet)

# single imputation in mice
imp <- mice(dat, m=1, maxit=10)

# apply PCA
pca.imp <- pca.covridge(complete(imp))
## > pca.imp$sdev
## Comp.1 Comp.2 Comp.3 Comp.4 Comp.5 Comp.6 Comp.7
## 3.0316816 2.3907523 2.2067445 2.0625173 1.8220392 1.6979627 1.6350428

# compare results with princomp
pca2.imp <- princomp(complete(imp))
## > pca2.imp
## Call:
## princomp(x = complete(imp))
## ## Standard deviations:
## Comp.1 Comp.2 Comp.3 Comp.4 Comp.5 Comp.6 Comp.7
## 3.0316816 2.3907523 2.2067445 2.0625173 1.8220392 1.6979627 1.6350428

## End(Not run)
```

---

### R Utilities: Evaluates a String as an Expression in R

**Description**

This function evaluates a string as an R expression.

**Usage**

```r
Reval(Rstring, print.string=TRUE)
Revalpr(Rstring, print.string=TRUE) # = Reval(print(Rstring))
Revalprstr(Rstring, print.string=TRUE) # = Reval(print(str(Rstring)))
```

**Arguments**

- `Rstring`: String which shall be evaluated in R
- `print.string`: Should the string printed on the console?
Details

The string is evaluated in the parent environment. See `eval` for the definition of environments in R.

Author(s)

Alexander Robitzsch

Examples

```r
# This function is simply a shortage function
# See the definition of this function:
Reval <- function(Rstring, print.string=TRUE){
  if (print.string) cat(paste(Rstring, "\n"))
  eval.parent(parse(text = paste(Rstring), n=1))
}

Reval("a <- 2^3")
## a <- 2^3
a
## [1] 8
```

---

**Rhat.mice**

*Rhat Convergence Statistic of a mice Imputation*

Description

Computes the Rhat statistic for a `mids` object.

Usage

`Rhat.mice(mice.object)`

Arguments

- **mice.object** Object of class `mids`

Value

Data frame containing the Rhat statistic for mean and variances for all variables of the Markov chains used for imputation

Author(s)

Alexander Robitzsch

References

Examples

## Not run:

### EXAMPLE 1: Rhat statistic for nhanes data

```r
library(mice)
data(nhanes, package="mice")
set.seed(9090)

# nhanes 3 parallel chains
imp1 <- mice( nhanes ,m=3 , maxit=10 , imputationMethod=rep("norm", 4 ) )
Rhat.mice( imp1 )

# variable MissProp Rhat.M.imp Rhat.Var.imp
#  1  bmi  36  1.0181998  1.155807
#  2  hyp  32  1.0717677  1.061174
#  3  chl  40  0.9717109  1.318721
```

## End(Not run)

---

**round2**

R Utilities: Rounding DIN 1333 (Kaufmaennisches Runden)

Description

This is a rounding function which rounds up for all numbers according to the rule of 'kaufmaennisches Runden' (DIN 1333).

Usage

```r
round2(vec, digits = 0)
```

Arguments

- `vec` Numeric vector
- `digits` Number of digits after decimal for rounding

Value

Vector with rounded values

Author(s)

Alexander Robitzsch
Examples

```r
### EXAMPLE 1
vec <- c( 1.5 , 2.5 , 3.5 , 1.51 , 1.49 )
vec
round(vec)
round2(vec)
  ## > vec
  ## [1] 1.50 2.50 3.50 1.51 1.49
  ## > round(vec)
  ## [1] 2 2 4 2 1
  ## > round2(vec)
  ## [1] 2 3 4 2 1

### EXAMPLE 2
vec <- c( 1.5 , 2.5 , 3.5 , 1.51 , 1.49 )
vec
round(vec)
round2(vec)
  ## > vec
  ## [1] -1.50 -2.50 -3.50 -1.51 -1.49
  ## > round(vec)
  ## [1] -2 -2 -4 -2 -1
  ## > round2(vec)
  ## [1] -2 -3 -4 -2 -1

### EXAMPLE 3
vec <- c(8.4999999 , 8.5 , 8.501 , 7.4999999 , 7.5 , 7.501 )
round(vec)
round2( vec , digits=1)
round2( -vec )
  ## > round(vec)
  ## [1] 8 8 9 7 8 8
  ## > round2( vec )
  ## [1] 8 9 9 7 8 8
  ## > round2( vec , digits=1)
  ## [1] 8.5 8.5 8.5 7.5 7.5 7.5
  ## > round2( -vec )
  ## [1] -8 -9 -9 -7 -8 -8
```

Rsessinfo

**R Utilities: R Session Information**

Description

Informs about current R session.

Usage

Rsessinfo()
Value

A string containing reduced information about R session info

Author(s)

Alexander Robitzsch

Examples

rsessinfo()

```r
#> Rsessinfo()
#> [1] "R version 2.15.2 (2012-10-26) x86_64, mingw32 | nodename = SD70 | login = robitzsch"
```

save.data

R Utilities: Saving/Writing Data Files using miceadds

Description

This function is a wrapper function for saving or writing data frames or matrices.

Usage

```r
save.data( data, filename, type="Rdata", path=getwd(), row.names=FALSE, na=NULL, ...)
```

Arguments

data
Data frame or matrix to be saved

filename
Name of data file

type
The type of file in which the data frame or matrix should be loaded. This can be Rdata (for R binary format, using save), csv (using write.csv), csv (using write.csv), table (using write.table) or sav (using write.pspp (miceadds)).

path
Directory from which the dataset should be loaded

row.names
Optional logical indicating whether row names should be included in saved csv or csv2 files.

na
Missing value handling. The default is "" for type="csv" and type="csv2" and is "." for type="table".

... Further arguments to be passed to save, write.csv2, write.csv, write.table or write.pspp (miceadds).

Details

For writing sav files, an installation of PSPP is necessary. See write.pspp.
Author(s)

Alexander Robitzsch

See Also

See `load.Rdata` and `load.data` for saving/writing R data frames.

Examples

```r
## Not run:
### use data.ma01 as an example for writing data files using save.data
data(data.ma01)
dat <- data.ma01

# set a working directory
pf2 <- "P:/ARb/temp_miceadds"

# save data in Rdata format
save.data(dat, filename="ma01data.Rdata", type="Rdata", path=pf2)

# save data in table format
save.data(dat, filename="ma01data.dat", type="table", path=pf2, row.names=FALSE, na = ".")

# save data in csv format
save.data(dat, filename="ma01data.csv", type="csv", path=pf2, row.names=FALSE, na = "")

# save data in sav format
# - do not forget to specify the PSPP path on your computer
# - note that an installed PSPP version is necessary (see write.pspp)
save.data(dat, filename="ma02data", type="sav", path=pf2, pspp.path = "C:/Program Files (x86)/PSPP/bin/")

## End(Not run)
```

Description

This function saves a data frame in a Rdata format.

Usage

```r
save.Rdata(dat, name, path = NULL, part.numb = 1000)
```
Arguments

dat Data frame
name Name of the R object to be saved
path Directory for saving the object
part.numb Number of rows of the data frame which should also be saved in csv format. The default is saving 1000 rows.

Author(s)

Alexander Robitzsch

Examples

## Not run:
dfr <- matrix( 2:1:12-3 , 4 ,3 )
save.Rdata( dfr , "dataframe_test" )
## End(Not run)

scan.vec R Utilities: Scan a Character Vector

Description

This function splits a string into a character vector.

Usage

scan.vec(vec)
scan.vector(vec)

Arguments

vec A string which should be splitted according to blanks

Author(s)

Alexander Robitzsch

Examples

vars <- scan.vector( "female urbgrad groesse Nausg grpgroesse privat ")
## Read 6 items
vars
## [1] "female" "urbgrad" "groesse" "Nausg" "grpgroesse" "privat"
## [6] "privat"
source.all  

**R Utilities: Source All R Files Within a Directory**

**Description**

Sources all R files within a specified directory.

**Usage**

```r
source.all( path, grepstring="\.R", print.source=TRUE )
```

**Arguments**

- **path**: Path where the files are located
- **grepstring**: Which strings should be looked for? grepstring can also be a vector.
- **print.source**: An optional logical whether the source process printed on the console?

**Author(s)**

Alexander Robitzsch

**Examples**

```r
## Not run:
# define path
path <- "c:/myfiles/
# source all files containing the string 'Rex'
source.all( path, "Rex" )
## End(Not run)
```

---

str_C.expand.grid  

**R Utilities: String Paste Combined with expand.grid**

**Description**

String paste combined with expand.grid

**Usage**

```r
str_C.expand.grid(xlist, indices=NULL)
```

**Arguments**

- **xlist**: A list of character vectors
- **indices**: Optional vector of indices to be permuted in xlist
Value

A character vector

Author(s)

Alexander Robitzsch

Examples

```r
# EXAMPLE 1: Some toy examples
x1 <- list(c("a","b"), c("t", "r","v")
str_C.expand.grid(x1)
## [1] "at" "bt" "ar" "br" "av" "bv"

x1 <- list(c("a","b"), paste0("_", 1:4 ), c("t", "r","v")
str_C.expand.grid(x1, indices=c(2,1,3))
## [1] "_1at" "_1bt" "_2at" "_2bt" "_3at" "_3bt" "_4at" "_4bt" "_1ar" "_1br"
## [11] "_2ar" "_2br" "_3ar" "_3br" "_4ar" "_4br" "_1av" "_1bv" "_2av" "_2bv"
## [21] "_3av" "_3bv" "_4av" "_4bv"

## Not run:
# The function 'str_C.expand.grid' is currently defined as
function( xlist , indices=NA ){
  xeg <- expand.grid( xlist)
  if (!is.null(indices))[ xeg <- xeg[ , indices ]
  apply( xeg , 1 , FUN = function(vv){ paste0( vv , collapse="" ) } )
}
## End(Not run)
```

Description

This function implements sum preserving rounding. If the supplied data is a matrix, then the sum of all row entries is preserved.

Usage

```r
sumpreserving.rounding(data, digits=0, preserve=TRUE)
```
Arguments

- **data**: Vector or data frame
- **digits**: Number of digits to be round
- **preserve**: Should the sum be preserved?

Author(s)

Alexander Robitzsch

Examples

```r
# Example 1
# define example data
data <- c(1455, 1261, 1067, 970, 582, 97)
data <- 100 * data / sum(data)

(x1 <- round(data))
sum(x1)
(x2 <- sumpreserving.rounding(data))
sum(x2)

> ( x1 <- round(data) )
> [1] 27 23 20 18 11 2
> sum(x1)
> [1] 101
> (x2 <- sumpreserving.rounding(data))
> [1] 27 23 20 18 10 2
> sum(x2)
> [1] 100

# Example 2
# matrix input
data <- rbind(data, data)
(x1 <- round(data))
rowSums(x1)
(x2 <- sumpreserving.rounding(data))
rowSums(x2)

# Example 3
x2 <- c(1.4, 1.4, 1.2)
round(x2)
sumpreserving.rounding(x2)
```
### Description

This function generates system time strings in several formats.

### Usage

```r
systime()
```

### Value

A vector with entries of system time (see Examples).

### Author(s)

Alexander Robitzsch

### Examples

```r
systime()
## > systime()
## [5] "2013-09-22_1000"
```
Arguments

- `data`: Matrix of item responses corresponding to a scale
- `integer`: A logical indicating whether imputed values should be integers. The default is `FALSE`.
- `iter`: Number of iterations

Details

For persons `p` and items `i`, the two-way imputation is conducted by posing a linear model of tau-equivalent measurements:

\[ X_{pi} = \theta_p + b_i + \varepsilon_{ij} \]

If the score `X_{pi}` is missing then it is imputed by

\[ \hat{X}_{pi} = \bar{X}_p + b_i \]

where \( \bar{X}_p \) is the person mean of person `p` of the remaining items with observed responses.

The two-way imputation can also be seen as a scaling procedure to obtain a scale score which takes different item means into account.

Value

A matrix with original and imputed values

Author(s)

Alexander Robitzsch

References


Examples

```r
# EXAMPLE 1: Two-way imputation data.internet

data(data.internet)
data <- data.internet

set.seed(765)
dat.imp <- tw.imputation(data)
dat.imp[278:281,]
```
write.mice.imputation

Exports Multiply Imputed Datasets from a mids Object

Description

Exports multiply imputed datasets and information about the imputation. Objects of class mids (generated by mice) and mids.1chain (generated by mice.1chain) are supported.

Usage

\[
\text{write.mice.imputation}(\text{mi.res, name, include.varnames = TRUE, long = TRUE, mids2spss = TRUE, spss.dec = ",", dattype = NULL})
\]

Arguments

- mi.res: Object of class mids
- name: Name of created folder and datasets
- include.varnames: An optional logical indicating whether variable names should be included in the imputed dataset. The default is TRUE.
- long: An optional logical indicating whether the dataset should also be saved in a long format?
- mids2spss: An optional logical indicating whether a syntax for reading imputed datasets in SPSS should be included
- spss.dec: SPSS decimal separator (can be "," or ".")
- dattype: Format of the saved dataset: csv or csv2
Value

Several files are saved using impxxx (the name) as the prefix:

- **impxxx.Rdata**: Saved object of class mids
- **impxxx__DATA LIST.Rdata**: Saved object of a list containing multiply imputed datasets
- **impxxx__IMP LIST**: File with list of multiply imputed datasets
- **impxxx__IMP SUMMARY**: Summary file of the imputation
- **impxxx__IMPDATA nn**: Imputed datasets nn
- **impxxx__IMPMETHOD**: File containing imputation methods
- **impxxx__LEGENDE**: File with variable names of the dataset
- **impxxx__LONG**: Imputed datasets in long format
- **impxxx__PREDICTORMATRIX**: File containing the predictor matrix
- **impxxx__SPSS.sps**: SPSS syntax for reading the corresponding txt file into SPSS format.

Author(s)

Alexander Robitzsch

See Also

See also mids2mplus and mids2spss (in mice package)

Examples

```
# Not run:
# example 1: Imputation of nhanes data and write imputed datasets on disk

data(nhanes, package="mice")

# model 1: Imputation using mice
imp1 <- mice( nhanes , m=3 , maxit=5 )
# write results
write.mice.imputation(mi.res=imp1, name="mice_imp1")

# model 2: Imputation using mice.1chain
imp2 <- mice.1chain( nhanes , burnin=10 , iter=20 , Nimp=4 )
```
# write results
write.mice.imputation(mi.res=imp2, name="mice_imp2")

## End(Not run)

---

### write.pspp

**Writing a Data Frame into SPSS Format Using PSPP Software**

#### Description

Writes a data frame into SPSS format using the *PSPP* software. To use this function, download and install PSPP at first: [http://www.gnu.org/software/pspp/pspp.html](http://www.gnu.org/software/pspp/pspp.html).

#### Usage

```r
write.pspp(data, datafile, pspp.path, decmax = 6,
            as.factors=TRUE , use.bat=FALSE)
```

#### Arguments

- **data**: Data frame
- **datafile**: Name of the output file (without file ending)
- **pspp.path**: Path where the PSPP executable is located, e.g. "C:/Program Files (x86)/PSPP/bin/"
- **decmax**: Maximum number of digits after decimal
- **as.factors**: A logical indicating whether all factors and string entries should be treated as factors in the output file.
- **use.bat**: A logical indicating whether PSPP executed via a batch file in the DOS mode (TRUE) or directly envoked via the system command from within R (FALSE).

#### Value

A dataset in sav format (SPSS format).

#### Author(s)

Alexander Robitzsch


#### See Also

See also `write.foreign` in the *foreign* package.

For convenient viewing sav files we recommend the freeware program ViewSav, see [http://www.asselberghs.dds.nl/stuff.htm](http://www.asselberghs.dds.nl/stuff.htm).
Examples

## Not run:

# EXAMPLE 1: Write a data frame into SPSS format

```r
# (1) define data frame
data <- data.frame("pid" = 1000:1:5, "height" = round(rnorm(5), 4),
                 "y" = 10*c(1,1,1,2,2), "r2" = round(rnorm(5), 2),
                 "land" = as.factor(c(rep("A",1), rep("B", 4))))

# (2) define variable labels
v1 <- rep("", ncol(data))
names(v1) <- colnames(data)
attr(data, "variable.labels") <- v1
attr(data,"variable.labels")["pid"] <- "Person ID"
attr(data,"variable.labels")["height"] <- "Height of a person"
attr(data,"variable.labels")["y"] <- "Gender"

# (3) define some value labels
v1 <- c(10,20)
names(v1) <- c("male", "female")
attr(data,y,"value.labels") <- v1

# run PSPP to produce a sav file
write.pspp(data, datafile = "example_data1",
           pspp.path = "C:\Program Files (x86)/PSPP/bin/"
           )

# produce strings instead of factors
write.pspp(data, datafile = "example_data2",
           pspp.path = "C:\Program Files (x86)/PSPP/bin/", as.factors=FALSE)
```

## End(Not run)
Index

*Topic **ANOVA**
  mi.anova, 28
*Topic **Chi square statistic**
  micombine.chisquare, 56
*Topic **Convergence**
  Rhat.mice, 63
*Topic **D2 statistic**
  micombine.chisquare, 56
*Topic **Decriptives**
  micombine.cor, 57
*Topic **Descriptives**
  ma.wtd.statNA, 27
*Topic **Dimension reduction**
  kernelpls.fit2, 21
  mice.impute.2l.pls2, 46
  pca.covridge, 61
*Topic **F statistic**
  micombine.F, 58
*Topic **Imputation**
  mice.1chain, 29
*Topic **Latent variables**
  draw.pv.ctt, 17
*Topic **PSPP (SPSS)**
  write.pssp, 76
*Topic **Partial least squares regression**
  (PLS)
  kernelpls.fit2, 21
  mice.impute.2l.pls2, 46
*Topic **Plausible value imputation**
  draw.pv.ctt, 17
*Topic **Predictive mean matching**
  mice.impute.pmm3, 50
  mice.impute.weighted.pmm, 55
*Topic **Principal component analysis**
  pca.covridge, 61
*Topic **R utilities**
  crlrem, 4
  cxxfunction.copy, 5
  grep.vec, 20
  index.dataframe, 21
  load.data, 23
  load.Rdata, 24
  output.format1, 60
  Reval, 62
  round2, 64
  Rsessinfo, 65
  save.data, 66
  save.Rdata, 67
  scan.vec, 68
  source.all, 69
  systime, 72
*Topic **Rhat statistic**
  Rhat.mice, 63
*Topic **Rounding**
  round2, 64
  sumpreserving.rounding, 70
*Topic **Sampling weights**
  mice.impute.weighted.norm, 54
  mice.impute.weighted.pmm, 55
*Topic **Two-way imputation**
  tw.imputation, 72
*Topic **Utility function**
  fast.groupmean, 19
  ma.scale2, 25
  str_C.expand.grid, 69
  write.pssp, 76
*Topic **datasets**
  data.allison, 6
  data.enders, 8
  data.internet, 10
  data.largescale, 12
  data.ma, 12
  data.smallscale, 15
*Topic **mice imputation method**
  mice.impute.2l.contextual.norm, 33
  mice.impute.2l.contextual.pmm, 34
  mice.impute.2l.eap, 36
  mice.impute.2l.latentgroupmean, 38
INDEX

mice.impute.2l.plausible.values, 41
mice.impute.2l.pls2, 46
mice.impute.pmm3, 50
mice.impute.tricube.pmm, 52
mice.impute.weighted.norm, 54
mice.impute.weighted.pmm, 55

*Topic **mice utility function**
data.list2mids, 15
write.mice.imputation, 74

*Topic **mids.1chain**
mice.1chain, 29

*Topic **mids**
data.list2mids, 15
mi.anova, 28
mice.1chain, 29
micombine.cor, 57
mids2datlist, 59
write.mice.imputation, 74

*Topic **package**
miceadds-package, 3

*Topic **plot**
mice.1chain, 29

*Topic **predict**
kernelpls.fit2, 21

*Topic **summary**
mice.1chain, 29

*Topic **z-Standardization**
ma.scale, 25
2lonly.norm2 (mice.impute.2lonly.pmm2), 49
2lonly.pmm2 (mice.impute.2lonly.pmm2), 49

as.mids, 16
crlrem, 4
cxxfunction.copy, 5
data.allison, 6, 7
data.enders, 8
data.internet, 10
data.largescale, 12, 51
data.ma, 12
data.ma01 (data.ma), 12
data.ma02 (data.ma), 12
data.ma03 (data.ma), 12
data.ma04 (data.ma), 12
data.ma05 (data.ma), 12
data.smallscale, 15, 51
data.list2mids, 15
draw.pv.ctt, 17
eigenvalues.sirt, 61
eval, 63
fast.groupmean, 14, 19
fast.groupsum (fast.groupmean), 19
grep.vec, 20
index.dataframe, 21
kernelpls.fit2, 21, 48
load, 25
load.data, 23, 67
load.Rdata, 24, 24, 67
load.Rdata2, 23
load.Rdata2 (load.Rdata), 24
ma.scale, 3, 11, 25
ma.wtd.corNA (ma.wtd.statNA), 27
ma.wtd.covNA (ma.wtd.statNA), 27
ma.wtd.meanNA (ma.wtd.statNA), 27
ma.wtd.pmiNA (ma.wtd.statNA), 27
ma.wtd.pmiNA (ma.wtd.statNA), 27
ma.wtd.statNA, 14, 27
mi.anova, 28
mice, 29–31, 74
mice.1chain, 3, 14, 29, 74
mice.impute.2l.contextual.norm, 33
mice.impute.2l.contextual.pmm, 14, 34, 34
mice.impute.2l.eap, 14, 36
mice.impute.2l.groupmean
(mice.impute.2l.latentgroupmean), 38
mice.impute.2l.latentgroupmean, 14, 38
mice.impute.2l.plausible.values, 3, 14, 41
mice.impute.2l.pls, 3
mice.impute.2l.pls
(mice.impute.2l.pls2), 46
mice.impute.2l.pls2, 11, 46
mice.impute.2lonly.mean, 38
mice.impute.2lonly.norm, 35, 49, 50
mice.impute.2lonly.norm2
(mice.impute.2lonly.pmm2), 49
mice.impute.2lonly.pmm, 35, 49, 50
mice.impute.2only.pmm2, 49
mice.impute.pmm3, 50
mice.impute.pmm4(mice.impute.pmm3), 50
mice.impute.pmm5(mice.impute.pmm3), 50
mice.impute.pmm6(mice.impute.pmm3), 50
mice.impute.tricube.pmm, 52
mice.impute.tricube.pmm2
  (mice.impute.tricube.pmm), 52
mice.impute.weighted.norm, 54, 55
mice.impute.weighted.pmm, 14, 54, 55
miceadds (miceadds-package), 3
miceadds-package, 3
micombine.chisquare, 28, 56, 58, 59
micombine.cor, 57
micombine.F, 28, 58
mids, 15
mids2datlist, 59
mids2mplus, 75
mids2spss, 75

output.format1, 60

pca.covridge, 11, 61
plausible.value.imputation.raschtype, 18
plot.mids.1chain (mice.1chain), 29
plsr, 47
pmm, 50
pool.compare, 57
predict.kernelpls.fit2
  (kernelpls.fit2), 21
princomp, 61

read.csv, 23
read.csv2, 23
read.spss, 23
read.table, 23
Reval, 62
Revalpr (Reval), 62
Revalprstr (Reval), 62
Rhat.mice, 63
round2, 64
Rsessinfo, 65

save, 25, 66
save.data, 24, 66
save.Rdata, 24, 25, 67
scan.vec, 68
scan.vector (scan.vec), 68

source.all, 69
str_C.expand.grid, 69
summary.mids.1chain (mice.1chain), 29
sumpreserving.rounding, 70
systime, 72
tw.imputation, 3, 11, 72
tw.mcmc.imputation (tw.imputation), 72
write.csv, 66
write.csv2, 66
write.mice.imputation, 74
write.pspp, 3, 66, 76
write.table, 66