Package 'eatPrep'

December 18, 2012

Type Package

Title eatPrep

Version 0.0.9

Depends R (>= 2.14.0), eatTools, xlsx, car, foreign, reshape

Date 2012-12-18

Author eat authors <eat-commits@lists.r-forge.r-project.org>

Maintainer eat authors <eat-commits@lists.r-forge.r-project.org>

Description preparing data for IRT modeling

License GPL (>= 2)

LazyLoad yes

LazyData yes

OS_type windows

R topics documented:

aggregateData *Aggregate Datasets with Several Kinds of Missing Values*

Description

Aggregate datasets with constraints on missing values

Usage

```
aggregateData(dat, subunits, units, aggregatemissings = NULL,
              rename = FALSE, recodedData = TRUE)
```

Arguments

dat	A data frame containing the data to be aggregated.
subunits	A data frame with subunit information. See 'Details'.
units	A data frame with unit information. See 'Details'.
aggregatemissings	Optional: A symmetrical *n x n* matrix with information on how missing values should be aggregated. If no matrix is given, the default will be used. See 'Examples'.
rename	Logical indicating whether units with only one subunit should be renamed to their unit name? Default is FALSE.
recodedData	Logical indicating whether colnames in dat are the subunit names (as in subunits$subunit) or recoded subunit names (as in subunits$subunitRecoded). Default is TRUE, meaning that colnames are recoded subitem names.

Details

aggregateData aggregates units in data frames with special consideration of missing values.The aggregation of missing values is specified in the argument aggregatemissings. The rownames and colnames of this *n x n* matrix correspond to the missing codes in the data. Additionally, the values "vc" (for valid code) and "err" (for error) are used. If aggregatemissings is a data frame, it will be coerced to a matrix with the first column of the data frame being transformed into the rownames of the matrix. A warning will be given if the matrix is not symmetrical.

aggregateData combines the subunits one by one, i.e. it aggregates the first two subunits of a unit, then adds the third subunit to the new aggregated variable and continues in this manner until all subunits are aggregated. In every step during the process a value of the first variable (e.g., the aggregated variable from the previous step) is matched with the rownames of aggregatemissings and the corresponding value of the second variable (e.g., the next subitem to be aggregated) is matched with the colnames of aggregatemissings. The new value of the aggregated variable will therefore be the value in aggregatemissings[firstVar, secondVar]. If the aggregation produces "err" at any point, it will stop. If the value in the final aggregated variable is "vc", either the mean or the sum of subunits will be calculated (the rule given in units$unitAggregateRule determines which one will be chosen).

Examples of data frames subunits and units can be found via data(inputList).

Value

A data frame with aggregated units and, if rename = TRUE, renamed subunits.

Warning

Missings are only correctly aggregated if their values correspond to the values given in aggregatemissings. aggregateData does not check for value types or whether codes are valid. Use of checkData and recodeData before using aggregateData is therefore strongly recommended.

Author(s)

Nicole Haag, Anna Lenski

References

For missing types see http://code.google.com/p/zkdlib/wiki/MissingHandling

See Also

recodeData, checkData

Examples

```
data(inputDat)
data(inputList)

dat1 <- inputDat[[1]]  # get first dataset from inputDat

# recode data
datRec <- recodeData(dat1, inputList$values, inputList$subunits)

# define matrix for missing aggregation (note: this is the default matrix)
am <- matrix(c("vc" , "mvi", "vc" , "mci", "err", "vc" , "vc" , "err",
"mvi", "mvi", "err", "mci", "err", "err", "err", "err",
"vc" , "err", "mnr", "mci", "err", "mir", "mnr", "err",
"mci", "mci", "mci", "mci", "err", "mci", "mci", "err",
"err", "err", "err", "err", "mbd", "err", "err", "err",
"vc" , "err", "mir", "mci", "err", "mir", "mir", "err",
"vc" , "err", "mnr", "mci", "err", "mir", "mbi", "err",
"err", "err", "err", "err", "err", "err", "err", "err" ),
nrow = 8, ncol = 8, byrow = TRUE)

dimnames(am) <- list(c("vc" ,"mvi", "mnr", "mci",  "mbd", "mir", "mbi", "err"),
c("vc" ,"mvi", "mnr", "mci",  "mbd", "mir", "mbi", "err"))

print(am)

datAggr <- aggregateData(datRec, inputList$subunits, inputList$units,
            aggregatemissings = am, rename = TRUE, recodedData = TRUE)
```

---

automateDataPreparation

automateDataPreparation

Description

prepare datasets for automateModels

Usage

```
automateDataPreparation(datList = NULL, inputList, path = NULL,
readSpss, checkData,  mergeData , recodeData, recodeMnr = FALSE,
aggregateData, scoreData, writeSpss,
filedat = "mydata.txt", filesps = "readmydata.sps", breaks=NULL, nMbi = 2,
aggregatemissings = NULL, rename = TRUE, recodedData = TRUE,
correctDigits=FALSE, truncateSpaceChar = TRUE, newID = NULL, oldIDs = NULL,
missing.rule = list(mvi = 0, mnr = 0, mci = 0, mbd = NA, mir = 0, mbi = 0))
```

Arguments

datList	A list of data frames if no .sav files shall be read in.
inputList	A list of data frames containing additional information (see Details).
path	A character string containing the path where the logfolder will be created. Also required by readSpss (source of SPSS files) and writeSpss. Default is the current R working directory.
readSpss	logical (whether function readSpss shall be called).
checkData	logical (whether function checkData shall be called).
mergeData	logical (whether function mergeData shall be called).
recodeData	logical (whether function recodeData shall be called for subunits).
recodeMnr	logical (whether function recodeMbiToMnr shall be called for subunits).
aggregateData	logical (whether function aggregateData shall be called).
scoreData	logical (whether function recodeData shall be called for units).
writeSpss	logical (whether function writeSpss shall be called).
filedat	A character string with the name of the output data file required by writeSpss.
filesps	A character string with the name of the output syntax file required by writeSpss.
breaks	Numeric vector (argument used by recodeMbiToMnr).
nMbi	Numeric (argument used by recodeMbiToMnr).
missing.rule	A list containing recode information for character missings required by writeSpss. See 'References' for description of default values.
aggregatemissings	A symmetrical *n x n* matrix or a data frame from inputList$aggrMiss with information on how missing values should be aggregated. If no matrix is given, the default will be used. See 'Details' in aggregateData.
rename	logical. See aggregateData.
recodedData	logical. See aggregateData.
correctDigits	logical. See readSpss.
truncateSpaceChar	logical. See readSpss.
newID	A character string containing the case IDs name in the final data frame. Default is "ID" or a character string specified in inputList sheet 6 (see readDaemonXlsx).
oldIDs	A vector of character strings containing the IDs names in the original datasets. Default is as specified in inputList$savFiles.

Details

inputList is a list of data frames. It can be created by ZKDaemon via readDaemonXlsx. Compulsory: units, subunits, values. Optional: unitRecodings, savFiles, newID, aggregateMissings.

Value

A single data frame in last transformation status.

Author(s)

Karoline Sachse

References

http://code.google.com/p/zkdlib/wiki/MissingHandling

Examples

```
data(inputList)
data(inputDat)
test <- automateDataPreparation(inputList = inputList, datList = inputDat,
        path = "c:/temp/test_eat", readSpss = FALSE, checkData=TRUE,
        mergeData = TRUE, recodeData=TRUE, recodeMnr = TRUE, breaks = c(1,2),
    aggregateData=TRUE, scoreData= TRUE, writeSpss=TRUE)
```

---

catPbc — *Calculate Item Discrimination for Each Category of Categorical Variables*

Description

catPbc calculates discrimination statistics for the categories of categorical variables, i.e. the correlations of each category with the total score on the test. This information can be useful in determining which categories of an item are influencing the overall fit and discrimination in item response scaling applications and/or to find mistakes in recoding.

Usage

```
catPbc(datRaw, datRec, idRaw, idRec, context.vars, values, subunits, xlsx = NULL)
```

Arguments

datRaw	A merged unrecoded dataset
datRec	The same dataset as in datRaw, in recoded form
idRaw	Name or column number of the identifier (ID) variable in unrecoded dataset
idRec	Name or column number of the identifier (ID) variable in recoded dataset
context.vars	Names or column numbers of one or more context variables (e.g., sex, school). catPbc will ignore these variables.
values	Data frame with information about values, see inputList for details.
subunits	Data frame with information about subunits, see inputList for details.
xlsx	Optional: Full path of Excel file for results.

Details

The column names of datRaw and datRec must be consistent with the names provided by the columns subunit and subunitRecoded in data.frame subunits. Otherwise, catPbc will fail.

Value

A data frame with the discrimination values for each category of categorical variables. The data frame contains the following columns:

item	Name of unrecoded item
cat	Name of category
n	Number of responses for this item
freq	Absolute frequency of the category
freq.rel	Relative frequency of the category
catPbc	Discrimination value for the category (correlation with total score)
recodevalue	Recode value for the category
subunitType	Type of subunit, see inputList for details

Author(s)

Nicole Haag

Examples

```
data(inputDat)
data(inputList)

datRaw <- mergeData(newID = "ID", datList = inputDat, addMbd = TRUE)
datRec <- recodeData(datRaw, values = inputList$values, subunits=inputList$subunits)
pbcs   <- catPbc(datRaw, datRec, idRaw = "ID", idRec = "ID", context.vars = "hisei",
                 values = inputList$values, subunits = inputList$subunits)
```

---

checkData *Check Datasets for Missing Values and Invalid Codes*

Description

Check data frames for missing or duplicated entries in the ID variable, persons and/or variables without valid codes, and invalid codes. Invalid codes are codes which are not specified in table values.

Usage

```
checkData (dat, values, subunits, units)
```

Arguments

dat	A data frame to be checked.
values	A data frame with code information. See 'Details'.
subunits	A data frame with subunit information. See 'Details'.
units	A data frame with unit information. See 'Details'.

Details

The results of checkData will be written to the console.

Examples of data frames values, subunits and units can be found via data(inputList).

Author(s)

Nicole Haag, Anna Lenski

References

For missing types see http://code.google.com/p/zkdlib/wiki/MissingHandling

---

checkDesign *Check Datasets for test design deviations*

Description

Check data frames according to test design for valid codes instead of expected sysMis and for sysMis instead of valid codes.

Usage

```
checkDesign (dat, booklets, blocks, rotation, sysMis="NA", id="ID", subunits = NULL)
```

Arguments

dat	A data frame
booklets	A data frame with booklet information. See 'Examples'.
blocks	A data frame with block information. See 'Examples'.
rotation	A data frame with rotation information. See 'Examples'.
sysMis	sysMis identifier as character. Default is "NA".
id	Case-id identifier as character. Default is "ID".
subunits	Optional: A data frame with subunit information (c.f. inputList). See 'Details'. Used to find names of recoded subunits.

Author(s)

Karoline Sachse

Examples

```
data(inputDat)
data(inputList)

prepDat <- automateDataPreparation (inputList = inputList, datList = inputDat,
           readSpss = FALSE, checkData=FALSE, mergeData = TRUE, recodeData=TRUE,
       aggregateData=FALSE, scoreData= FALSE, writeSpss=FALSE)

checkDesign(dat = prepDat, booklets = inputList$booklets, blocks = inputList$blocks,
            rotation = inputList$rotation, sysMis = "mbd", id="ID",
            subunits = inputList$subunits)

prepDat[1:7,"I22R"] <- "1"
prepDat[1,c("I01R","I02R","I03R")] <- "mbd"

checkDesign(dat = prepDat, booklets = inputList$booklets, blocks = inputList$blocks,
            rotation = inputList$rotation, sysMis = "mbd", id="ID",
            subunits = inputList$subunits)
```

---

inputDat *List of Three Datasets from Educational Assessment*

Description

Simulated data for three booklets for an educational assessment study.

Usage

```
data(inputDat)
```

Format

This list contains 3 data frames, each with the following columns:

ID Person-ID

Hisei A continuous covariate.

Ixx Item responses to a selection of 30 test items.

Details

code, subunit and unit descriptions are stored in dataset `inputList`.

Examples

```
data(inputDat)
str(inputDat)
```

inputList *Data Frames with Code, Subunit and Unit Information for Datasets in* inputDat

Description

These data frames contain information about codes, subunits and units for the datasets in inputDat and are necessary inputs for functions automateDataPreparation, checkData, recodeData and aggregateData.

Usage

```
data(inputList)
```

Format

A list with three data frames:

1. units: Unit information, contains the following columns:

   unit Unit name.
   unitType Subunit types: ID = ID variable; TI = test item; CV = context variable.
   unitLabel Unit label, to be used by writeSpss.
   unitDescription Unit description.
   unitAggregateRule Aggregate rule for unit: SUM; MEAN.
   unitScoreRule Scoring rule for unit (not sure how this will be used in the future.)

2. subunits: Subunit information, contains the following columns:

   unit Unit name, for which subunits are given.
   subunit Subunit name.
   subunitType Subunit types:'?'.
   subunitLabel Subunit label, to be used by writeSpss.
   subunitDescription Subunit descriptions.
   subunitPosition Subunit position in test booklet (e.g., line 1).
   subunitTransniveau Subunit transformation level.
   subunitRecoded Name of recoded subunit.
   subunitLabelRecoded Label for recoded subunit, to be used when writeSpss is applied to a dataset produced by recodeData.

3. values: Value information, contains the following columns:

   subunit Subunit name, for which values are given.
   value Valid values for the respective subunit.
   valueRecode Recode values for the respective value.
   valueType Value types: vc = valid code; mbd = missing – by design; mvi = missing – volume insufficient; mnr = missing – not reached; mci = missing – coding impossible; mbi = missing – by intention.
   valueLabel Value labels, to be used by writeSpss.
   valueDescription Value descriptions.
   valueLabelRecoded Labels for recoded values, to be used when writeSpss is applied to a dataset produced by recodeData.

valueDescriptionRecoded Descriptions for recoded values.

4. unitRecodings: Unit recoding information, contains the following columns:

   unit Unit name

   value Valid values for the respective unit.

   valueRecode Recode values for the respective value.

   valueType Value types: `vc` = valid code; `mbd` = missing – by design; `mvi` = missing – volume insufficient; `mnr` = missing – not reached; `mci` = missing – coding impossible; `mbi` = missing – by intention.

   valueLabel Value labels, to be used by `writeSpss`.

   valueDescription Value descriptions.

   valueLabelRecoded Labels for recoded values, to be used when `writeSpss` is applied to a dataset produced by `recodeData`.

5. savFiles: information for `readSpss`, contains the following columns:

   filename SPSS filenames

   case.id ID variable in the respective dataset, used by `mergeData`

6. newID: information for `mergeData`, contains the following columns:

   key one of the entries should be `master-id`

   value the corresponding value; how the ID variable in the final dataset shall be named

7. aggrMiss: missing aggregation pattern for `aggregateData`

Examples

```
data(inputList)
str(inputList)
```

`mergeData` *Merge Data Frames using one Key Variable*

Description

Merges several data frames and matches them using one key variable

Usage

```
mergeData(newID = "ID", datList, oldIDs=NULL, addMbd = FALSE, writeLog=FALSE)
```

Arguments

`newID`	character string containing the key variable's name in the merged dataset
`datList`	list of data frames to be merged
`oldIDs`	character vector OR numeric vector containing either names of the key variables in datList or their column number in each dataframe in datList default is a vector containing replicates of the value of `newID`.
`addMbd`	logical; string "mbd" (missing by desgin) will be added instead of NA
`writeLog`	logical; if Logfile shall be written via `eatTools:::sunk`.

Value

A data frame containing unique cases and unique variables. All cases and all variables that could be identified the original data frames will be kept and matched.

Author(s)

Karoline Sachse, Nicole Haag

Examples

```
data(inputDat)
str(inputDat)

mergedDataset <- mergeData("idstud", inputDat, c("ID", "ID", "ID"), addMbd=TRUE)
str(mergedDataset)

mergedDataset <- mergeData("ID", inputDat, writeLog=FALSE)
str(mergedDataset)
```

---

readDaemonXlsx *read xlsx-Files produced by ZKDaemon*

Description

read xlsx-Files produced by ZKDaemon

Usage

```
readDaemonXlsx(filename)
```

Arguments

`filename`	A character string containing path, name and extension of .xlsx produced by ZKDaemon. Caution! Sheet names are important (see Datails).

Details

Reads in the following .xlsx sheets: "units", "subunits", "values", "unitrecoding", "sav-files", "params", "aggregate-missings", "itemproperties", "propertylabels", "booklets", "blocks"

Value

A list of data frames containing information that is required by `automateDataPreparation`

Author(s)

Karoline Sachse

Examples

```
str(inputList)
```

readSpss *Read SPSS Data Files and Truncate Space in String Variables and Change Column Width*

Description

Read SPSS Data Files and Truncate Space in String Variables and Change Column Width

Usage

```
readSpss(file, correctDigits = FALSE, truncateSpaceChar = TRUE, oldID = NULL, newID = NULL)
```

Arguments

file	Name of the SPSS data file to be read in
correctDigits	Logical: whether values should be transformed to have uniform width in each column, see 'Details'.
truncateSpaceChar	Logical: whether string variables should be trimmed to remove leading and trailing spaces.
oldID	Optional: A character string containing the ID name in the original SPSS dataset.
newID	Optional: A character string containing the ID name after reading in the data.

Details

If correctDigits=TRUE, the values in each column are transformed to have uniform width by adding leading zeros. The width of a column is determined by the longest value in this column, e.g., if a column contains the values 1, 10, 100 the transformed column will be 001, 010, 100.This can be useful if the values had leading zeros which were removed by reading in the SPSS file.

If oldID and newID are used, the ID variable in the SPSS dataset will be changed to newID when the file is read.

Value

A data frame with trimmed character variables and corrected values (if specified). All columns are of mode character.

Author(s)

Nicole Haag, Sebastian Weirich

recodeData *Recode Datasets with Missing Values*

Description

Recode datasets with special consideration of missing values.

Usage

```
recodeData(dat, values, subunits)
```

Arguments

dat	A data frame
values	A data frame with code information. See 'Details'.
subunits	A data frame with subunit information. See 'Details'.

Details

recodeData recodes data frames with special consideration of missing values. The results of recodeData will be written to a protocol file with sunk. recodeData will give warnings, if missing or incomplete recode informations are found. Values without recode information will NOT be recoded!

Examples of data frames values and subunits can be found via data(inputList)

Value

A data frame with recoded variables according to the specifications in values and subunits. Colnames will be the names specified in subunits$subunitRecoded.

Author(s)

Martin Hecht, Christiane Penk, Nicole Haag

References

http://code.google.com/p/zkdlib/wiki/MissingHandling

See Also

aggregateData, checkData

Examples

```
data(inputDat)
data(inputList)

dat1 <- inputDat[[1]]  # get first dataset from inputDat
datRec <- recodeData(dat1, inputList$values, inputList$subunits)
str(datRec)
```

`recodeMbiToMnr` *Recode Missing by Intention to Missing not Reached*

Description

`recodeMbiToMnr` converts missing responses coded as missing by intention at the end of a block of items to missing not reached.

Usage

```
recodeMbiToMnr (dat, id, booklets, blocks, rotation, breaks, nMbi = 2, subunits = NULL)
```

Arguments

`dat`	A dataset. Missing by intention needs to be coded `mbi`.
`id`	Name or column number of identifier (ID) variable in dataset.
`booklets`	A data frame containing the sequence of blocks in each booklet in wide format. The column names need to be `booklet, block1, block2, block3 ....`
`blocks`	A data frame containing the sequence of subunits in each block in long format. The column names need to be `subunit, block, subunitBlockPosition`.
`rotation`	A data frame containing the assignment of booklets to participants. The first column should have the same name as the ID variable in `dat` and the second column needs to be named`booklet`.
`breaks`	Number of blocks after which `mbi` shall be recoded to `mnr`, e.g., `c(1,2)` to specify breaks after the first and second block.
`nMbi`	Number of `mbi`-Codes required at the end of a block to code `mnr`. Needs to be >=1.
`subunits`	Optional: A data frame with subunit information if a dataset is used that has been recoded with `recodeData`. This data frame will be used to find the names of recoded subunits in `dat`.

Details

In order to code `mnr`, a certain number of subunits at the end of a block need to be coded `mbi`. This number can be specified with the argument `nMbi`. The default is 2, i.e. if the last and second to last subitem in a block are coded `mbi`, both subunits, as well as the preceding subunits coded `mbi`, will be recoded to `mnr`. If `nMbi` is larger than the number of subunits in a given block, no subitem in this block will be recoded. If all subunits in a block are coded `mbi`, none of them will be recoded to `mnr`.

If a `subunits` data frame is specified, `recodeMbiToMnr` expects to find the recoded subunits in `dat`.

Examples for data frames `booklets`, `blocks`, `rotation` and `subunits` can be found via `data(inputList)`

Value

A data frame with missing not reached coded as `mnr`. For each person with at least one `mnr` in the returned dataset the names of recoded variables are given as an attribute to `dat`.

Author(s)

Nicole Haag

Examples

```
data(inputDat)
data(inputList)

prepDat <- automateDataPreparation (inputList = inputList, datList = inputDat,
           readSpss = FALSE, checkData=FALSE, mergeData = TRUE, recodeData=TRUE,
       aggregateData=FALSE, scoreData= FALSE, writeSpss=FALSE)

mnrDat <- recodeMbiToMnr (dat = prepDat, id = "ID", booklets = inputList$booklets,
          blocks = inputList$blocks, rotation = inputList$rotation, breaks = c(1, 2),
          nMbi = 2, subunits = inputList$subunits)
```

---

scoreData *Score Datasets with Missing Values*

Description

Score datasets with special consideration of missing values.

Usage

```
scoreData(dat, unitrecodings, units)
```

Arguments

dat	A data frame
unitrecodings	A data frame with information about the scoring of units. See 'Details'.
units	A data frame with unit information. See 'Details'.

Details

This function is very similar to `recodeData`, but with a few defaults that are more sensible for scoring. `scoreData` will give warnings when incomplete scoring informations are found. Values without scoring information will not be scored!

Examples of data frames `unitrecodings` and `units` can be found via `data(inputList)`

Value

A data frame with scored variables according to the specifications in `unitrecodings` and `units`.

Author(s)

Nicole Haag

References

http://code.google.com/p/zkdlib/wiki/MissingHandling

See Also

`recodeData`, `automateDataPreparation`, `inputList`

Examples

```
data(inputDat)
data(inputList)

prepDat <- automateDataPreparation (inputList = inputList, datList = inputDat,
           readSpss = FALSE, checkData=FALSE, mergeData = TRUE, recodeData=TRUE,
          aggregateData=TRUE, scoreData= FALSE, writeSpss=FALSE)

datSco <- scoreData(prepDat, inputList$unitRecodings, inputList$units)
str(datSco)
```

`writeSpss` *Export Datasets to SPSS*

Description

Writes data and SPSS syntax files.

Usage

```
writeSpss(dat, values, subunits, units,
      filedat = "mydata.txt", filesps = "readmydata.sps",
      missing.rule = list(mvi = 0, mnr = 0, mci = NA, mbd = NA, mir = 0, mbi = 0),
          path = getwd(), sep = "\t", dec = ",", silent = FALSE)
```

Arguments

`dat`	A data frame which should be exported to SPSS.
`values`	A data frame with code information. See 'Details'.
`subunits`	A data frame with subunit information. See 'Details'.
`units`	A data frame with unit information. See 'Details'.
`filedat`	A character string with the name of the output data file.
`filesps`	A character string with the name of the output syntax file.
`missing.rule`	A list containing recode information for character missings. See 'References' for description of default values.
`path`	A character string containing the path of the output file. The value in `path` is appended to `filedat` and `filesps`. By default, files are written to the current R working directory. If `path=NULL` then no file path appending is done.
`sep`	The separator between the data fields.
`dec`	The decimal separator for numerical data.
`silent`	A logical flag stating whether the names of the files should be printed.

Details

This function automates most of the work needed to export a dataset to SPSS. It uses a modified version of `writeForeignSPSS()` from the `foreign` package and of `mids2spss()` from the `mice` package. The modified version allows for a choice of the field and decimal separators, makes some improvements to the formatting and provides variable labels and value labels according to the information in the data frames `values`, `subunits` and `units`.

Examples of data frames `values`, `subunits` and `units` can be found on `data(inputList)`

The SPSS syntax file has the proper file names and separators set, so in principle it should run and read the data without alteration. SPSS is more strict than R with respect to the paths. Always use the full path, otherwise SPSS may not be able to find the data file.

Value

Used for its side effects. The return value is `NULL`.

Author(s)

Nicole Haag

References

http://code.google.com/p/zkdlib/wiki/MissingHandling

See Also

`inputList`

Index