Package 'translateSPSS2R'

June 23, 2015

Type Package

Title Toolset for Translating SPSS-Syntax to R-Code

Version 1.0.0

Date 2015-06-23

Description Package with translated commands of SPSS. The usage is oriented on the handling of SPSS-Syntax. Mainly the package has two purposes: It facilitates SPSS-Users to change over to R and aids migration projects from SPSS to R.

License GPL-3

LazyData true

Depends R (>= 3.1.3)

Imports car (>= 2.0), data.table (>= 1.9.4), e1071 (>= 1.6-3), foreign (>= 0.8), Hmisc (>= 3.14-5), plyr (>= 1.8.1), stringr (>= 0.6.2), tidyr (>= 0.1), zoo (>= 1.7-11)

URL http://www.eoda.de/en/

BugReports https://github.com/eodaGmbH/translateSPSS2R/issues

NeedsCompilation no

Author Andreas Wygrabek [aut, cre], Bastian Wiessner [aut], eoda GmbH [cph]

Maintainer Andreas Wygrabek <Andreas.Wygrabek@eoda.de>

Repository CRAN

Date/Publication 2015-06-23 17:50:10

R topics documented:

translateSPSS2R-package *Documentation*

Description

translateSPSS2R provides a set of functions with translated SPSS commands. The usage is oriented on the handling of the SPSS-Syntax.

Arguments

Package:	translateSPSS2R
Type:	Package
Version:	1.0.0
Date:	2015-23-06
Imports:	car ,data.table, e1071, foreign, Hmisc, plyr, stringr, tidyr, zoo
License:	GPL-3
Maintainer:	Andreas Wygrabek

Details

Mainly the package has two purposes:

1. It facilitates SPSS-Users to change over to R.
2. It facilitates migration projects from SPSS to R.

. For automatic translation a webtool (http://www.eoda.de/en/translate2R.html) is provided by eoda GmbH. The tool translates SPSS-Syntax to R-Code by the use of translateSPSS2R functions.

Author(s)

Andreas Wygrabek <Andreas.Wygrabek@eoda.de> and Bastian Wiessner <Bastian.Wiessner@eoda.de>

References

https://service.eoda.de/translater/?lang=en
http://www.eoda.de/en

applyAttributes *Apply stored attributes*

Description

Applies attributes stored by attributesBackup

Usage

```
applyAttributes(x, attributesToApply = NULL)
```

Arguments

x	a data.frame or input data of class xpssFrame.
attributesToApply	to applied attributes

Value

Object with attributes from attributesBackup

Author(s)

Andreas Wygrabek

See Also

attributes attr

Examples

```
#load data
data(fromXPSS)
attributes(fromXPSS)
attributes(fromXPSS$V7_2)
x <- attributesBackup(fromXPSS)
fromXPSS <- fromXPSS[order(fromXPSS$V2),]
attributes(fromXPSS)
attributes(fromXPSS$V7_2)
fromXPSS <- applyAttributes(fromXPSS, x)
attributes(fromXPSS)
attributes(fromXPSS$V7_2)
```

`attributesBackup` *Stores Attributes*

Description

Attribut backup function

Usage

```
attributesBackup(x)
```

Arguments

`x`	a (non-empty) data.frame, data.table object or input data of class `"xpssFrame"`.

Details

The conditions to select cases are specified in a logical expression. These logical expressions can contain relational operators, logical operators and arithmetic operations.

NOTE: For temporary case selection, specify a TEMPORARY command before SELECT IF.

Value

Output is a subset of the actual dataset under the condiftion of the logical expression.

Author(s)

Bastian Wiessner #'

See Also

`attributes` `attr`

Examples

```
#load data
data(fromXPSS)
attributes(fromXPSS)
attributes(fromXPSS$V7_2)
x <- attributesBackup(fromXPSS)
fromXPSS <- fromXPSS[order(fromXPSS$V2),]
attributes(fromXPSS)
attributes(fromXPSS$V7_2)
fromXPSS <- applyAttributes(fromXPSS, x)
attributes(fromXPSS)
attributes(fromXPSS$V7_2)
```

fromXPSS *Sample dataset*

Description

Sampledata imported by xpssFrame(). The dataset contains 20 different cars from 3 continents.

Usage

```
data(fromXPSS)
```

Format

A data.frame with 20 rows and 10 variables.

Details

The variables are as follows:

- V1 Manufacturer. name of the manufacturer (Audi, BMW, Chevrolet,...)
- V2 Model. name of the model (A8, 328i, Malibu,...)
- V3 Country. numeric indicator for the country (1 = Germany, 2 = US, 3= Japan)
- V4 Car-Type. numeric indicator for the car-type (1 = PKW,2 = SUV)
- V5 Sales volume in thousand. (0.954-230.902)
- V5_kl2 Sales volume grouped. (1 = Until 30.000, 2 = More than 30.000)
- V6 Purchase price in thousand. (15.35-85.50)
- V6_kl3 Purchase price in three groups. (1 = Until 20.000, 2 = more than 20.000 until 30.000, 3 = more than 30.000)
- V7_1 PS. Amount of PS (135-310)
- V7_2 Weight. Weight of the car (94.5-138.60)

span *Indicates range of varlist*

Description

Creates a list of variables within a specific range.

Usage

```
span(x, from = NULL, to = NULL, addDF = FALSE)
```

Arguments

x	a (non-empty) data.frame or input data of class "xpssFrame".
from	the variable that opens the span.
to	the variable that closes the span.
addDF	if the name of the input data should be used?

Value

Returns a varlist with the name of the variables which are within the range of the span indicator.

Author(s)

Andreas Wygrabek

Examples

```
data(fromXPSS)
span(x=fromXPSS,from="V1",to="V5",addDF=FALSE)
span(x=fromXPSS,from="V3",to="V1",addDF=TRUE)
```

---

xpssAddValueLabels *Modifies value labels*

Description

R implementation of the SPSS `ADD VALUE LABELS` function. `xpssAddValueLabels` appends value labels for specific variables. Those values labels get stored in the attributes of the selected variable.

Usage

```
xpssAddValueLabels(x, variables = NULL, values = NULL, labels = NULL,
  datevariables = NULL, datevalues = NULL, datelabels = NULL)
```

Arguments

x	a (non-empty) data.frame or input data of class `"xpssFrame"`.
variables	atomic character or character vector with the name of the variables.
values	atomic numeric or numeric vector, respectively as an atomic character or character vector containing the value of the variable.
labels	atomic numeric or numeric vector, respectively as an atomic character or character vector containing the label of the variable.
datevariables	atomic date or date vector with the name of the date variables.
datevalues	atomic date or date vector containing the value of the date, the value has to be like the old date format.
datelabels	atomic numeric or numeric vector, respectively as an atomic character or character vector containing a variable label.

Details

The values labels are stored in the variable itself.
In contrast to xpssValueLabels , xpssAddValueLabels do not erase existing value labels.
If the value label for a specific variable already exists, this value label get overwritten.
If the value label for a specific variable does not exist, the value label get created without deleting the existing value label of that variable.

Supported attributes are: value.labels, defined.MIS, MIS, varname, variable.label

Value

An xpssFrame object with modified value labels.

Author(s)

Bastian Wiessner

See Also

read.spss xpssValueLabels xpssVariableLabels

Examples

```
data(fromXPSS)

fromXPSS <- xpssValueLabels(fromXPSS,
                            variables = "V1",
                            values = 1 ,
                            labels = "Label1")

fromXPSS <- xpssAddValueLabels(fromXPSS,
                            variables = "V1",
                            values = 2 ,
                            labels = "Label2")

fromXPSS <- xpssAddValueLabels(fromXPSS,
                            variables = "V1",
                            values = "B" ,
                            labels = "CharLabel")



attributes(fromXPSS$V1)$value.labels
```

xpssAny *Selecting cases by condition*

Description

xpssAny can be perceived as a wrapper function for %in% applicable on more than one variable.

Usage

```
xpssAny(x, st = NULL, nd = NULL)
```

Arguments

x	a (non-empty) data.frame or input data of class "xpssFrame".
st	atomic numeric or atomic character with a single value to search for OR variable where to search in.
nd	atomic numeric or atomic character, respecetively as numeric vector or character vector with values to search for OR variables to search in.

Value

A logical vector with matched conditions.

Author(s)

Andreas Wygrabek

See Also

xpssCount %in% is.element

Examples

```
data(fromXPSS)

xpssAny(fromXPSS, 310, c("V7_1", "V7_2"))

xpssAny(fromXPSS, "V7_1", c(310,320,170))

xpssAny(fromXPSS, "Audi", c("V1", "V7_2"))

xpssAny(fromXPSS, "V1", c("Audi"))
```

xpssCasenum *Sequential numbering of cases*

Description

R implementation of the SPSS $CASENUM system variable. xpssCasenum counts the number of cases within a variable, or respectively the number of observations in the dataset.

Usage

```
xpssCasenum(x)
```

Arguments

x	a (non-empty) data.frame or input data of class "xpssFrame".

Details

xpssCasenum fit well as ID-Variable.

Value

Returns a sequential atomic numeric or numeric vector.

Author(s)

Bastian Wiessner

See Also

Related Functions xpssCount , xpssDate , xpssDate11

Examples

```
data(fromXPSS)

fromXPSS$id <- xpssCasenum(fromXPSS)
```

xpssCompute *Creates data*

Description

R implementation of the SPSS COMPUTE argument.

Usage

```
xpssCompute(x, variables = NULL, fun = NULL,...)
```

Arguments

x	input data .
variables	atomic character or character vector with the names of the variables.
fun	atomic character as functionname.
...	further arguments passed to or from other methods.

Details

Missing Fuctions:

Functionname	Output
computeMiss	Computes missing values as logical (limited to one variable).
computeNmiss	Computes missing values as logical.
computeNvalid	Computes valid values as logical.
computeSysmis	Computes the system missing values as logical.
computeValue	Computes the user-defined values as valid values.

Numeric Fuctions:

Functionname	Output
computeAbs	Computes the absolute value.
computeArsin	Computes the arc-sine.
computeArtan	Computes the arc-tan.
computeCos	Computes the cosinus.
computeExp	Computes the exponential.
computeLn	Computes the logarithmus naturalis.
computeLg10	Computes the logarithmus base 10.
computeLngamma	Computes the logarithmus of the gamma function.
computeMax	Computes the maxima.
computeMean	Computes the atithmetic mean.
computeMedian	Computes the median value.
computeMin	Computes the minima.
computeMod	Computes the remainder of a division.
computeRnd	Computes rounded values.

computeSd	Computes the standard deviation.
computeVariance	Computes the variance.

Character Fuctions:

Functionname	Output
computeChar_index	Returns the position of the first occurence of a pattern.
computeChar_length	Computes the length of a string in characters.
computeChar_lpad	Returns an expanded strings.
computeChar_mblen	Computes the byte per character or sign.
computeChar_rindex	Returns the position of the last occurence of a pattern.
computeChar_rpad	Returns an expanded strings.
computeConcat	Returns a concatenated string.
computeLength	Computes Number of bytes in a string.
computeLower	Returns the input data to lower-case.
computeLtrim	Returns a trimmed string (left side trimmed).
computeNtrim	Returns the values of the input data.
computeReplace	Replaces a pattern in a string.
computeRtrim	Returns a trimmed string (right side trimmed).
computeStrunc	Returns a truncated string.
computeUpcase	Returns the input data to upper-case.

Lag Fuction:

Functionname	Output
computeLag	Shifts the variable backward or forward.

Date Fuctions:

Functionname	Output
computeCtime_days	Computes the difference of time between two dates in days.
computeCtime_hours	Computes the difference of time between two dates in hours.
computeCtime_minutes	Computes the difference of time between two dates in minutes.
computeCtime_seconds	Computes the difference of time between two dates in seconds.
computeDate_dmy	Computes a date with the format day-month-year.
computeDate_mdy	Computes a date with the format month-day-year.
computeDate_moyr	Computes a date with the format month-year.
computeDate_qyr	Computes a date with the format year/quarter.
computeDate_wkyr	Computes a date with the format year/calendar week.
computeDate_yrday	Computes a date with the format year/yearday.
computeTime_days	Computes the number of passed hours on basis of the given days
computeTime_hms	Computes a date with the format Hour-Minute-Second
computeXdate_date	Extracts the date out of a date string.
computeXdate_hour	Extracts the hour value out of a given date.
computeXdate_jday	Computes the date of year on basis of a given date.
computeXdate_mday	Extracts the date of month on basis of a given date.
computeXdate_minute	Extracts the minute component on basis of a given date.
computeXdate_month	Extracts the month component on basis of a given date.

computeXdate_quarter	Computes the quarter on basis of a given date.
computeXdate_second	Extracts the second on basis of a given date.
computeXdate_tday	Computes the difference of days between the entered date and October 14, 1582.
computeXdate_time	Extracts the time componente on basis of a given date.
computeXdate_week	Calcualtes the calendar week on basis of a given date.
computeXdate_wkday	Calcualtes the day of week on basis of a given date.
computeXdate_year	Extracts the year on basis of a given date.

Be careful about the input format of the numeric values in your data. It is possible to specify values which are outside the of valid range.
Those failures are called `Domain Errors`.

For example:

Function	Output
**	A negative number to a noninteger power.
/	A divisor of 0.
computeArsin	An argument whose absolute value exceeds 1.
computeExp	An argument that produces a result too large to be represented on the computer.
computeLg10	A negative or 0 argument.
computeLn	A negative or 0 argument.
computeMod	A divisor of 0.
computeSqrt	A negative argument.

Value

Output is a created vector.

Author(s)

Bastian Wiessner

See Also

Related Functions `xpssNumeric` `xpssString`

Examples

```
data(fromXPSS)

xpssCompute(x=fromXPSS, variables="V7_2",fun="computeValue")

xpssCompute(x=fromXPSS, variables=c("V5","V7_2"),fun="computeMean", na.rm=T)
```

`xpssCorrelations` *Pearson product-moment correlations*

Description

R implementation of the SPSS `CORRELATIONS` function.

Usage

```
xpssCorrelations(x,
                 variables = NULL,
                 miss = list(alternative = "pairwise",
                             missings = "exclude"),
                 print = list(test = "twotail",
                              level = "sig"),
                 matrix = NULL,
                 statistics = NULL)
```

Arguments

`x`	a (non-empty) data.frame or input data of class "xpss-Frame".
`variables`	atomic character or character vektor with the name of the variables.
`miss`	method which indicates what should happen when the data contain NAs. Default for alternative is 'pairwise', optionally listwise can be used as treatment for missings. The visualisation of the NAs can be specied via the argument missings. Default is 'exclude', optionally 'include' can be chosen to add missings in the statistics.
`print`	method which indicates what significnace level shall be used. Default significance test is 'twotail', optionally 'onetail' can be chosen. Default significance level is 'sig' to add significance asterisks, optionally 'nosig'.
`matrix`	exports the correlation matrix with observations, stddevs, means and variable names. Default is NULL.
`statistics`	method which enumerate the deskriptive statistics. Default is NULL. Optionally 'descriptives', 'xprod' or 'all' can be chosen.

Details

`xpssCorrelations` produces Pearson product-moment correlations with significance levels and, optionally, univariate statistics, covariances, and cross-product deviations.

Value

Returns a matrix of Pearson's r correlation.

Author(s)

Benjamin Piest

See Also

cor cor.test rcorr

Examples

```
data(fromXPSS)



xpssCorrelations (fromXPSS,
                  variables =c("V5","V6","V7_2"))



xpssCorrelations (fromXPSS,
                  variables =c("V5","V6","V7_2") ,
                  miss = list(alternative = "pairwise",
                                missings = "exclude"),
                  print = list(test = "onetail",
                               level = "sig"),
                  statistics="all")



xpssCorrelations (fromXPSS,
                  variables =c("V5","V6","V7_2") ,
                  miss = list(alternative = "pairwise",
                                missings = "include"),
                  print = list(test = "onetail",
                               level = "sig"),
                  statistics="all")



xpssCorrelations (fromXPSS,
                  variables =c("V5","V6","V7_2") ,
                  miss = list(alternative = "listwise",
                                missings = "exclude"),
                  print = list(test = "twotail",
                               level = "sig"),
                  statistics="all")



xpssCorrelations (fromXPSS,
                  variables =c("V5","V6","V7_2"),
                  statistics = "all",
                  matrix = paste0(getwd(),"/correlations.txt"))
```

xpssCount *Counts frequencies of specific observations*

Description

R implementation of the SPSS $COUNT system variable.

Usage

```
xpssCount(x, variables, count)
```

Arguments

x	a (non-empty) data.frame or input data of class xpssFrame.
variables	atomic character or character vector with the name of the variables.
count	atomic character or atomic numeric pattern.

Details

Count displays the frequencies of observations matching the count statement.

Value

A vector of the same length as x.

Author(s)

Bastian Wiessner

See Also

Related Functions xpssAny %in% is.element

Examples

```
data(fromXPSS)

xpssCount(x=fromXPSS,
   variables = "V1", count=list(exact="Nissan"))

xpssCount(x=fromXPSS,
   variables = "V5", count=list(exact=2))

xpssCount(fromXPSS,
   variables = span(fromXPSS, from = "V5",
                              to = "V7_2"),
                  count = list(from = "lo",
                               to = 100))
```

xpssDate *Today's date*

Description

R implementation of the SPSS `$date` system variable.

Usage

```
xpssDate()
```

Details

xpssDate provides the SPSS format of dates.

Value

Current date as "dd-mmm-yy". Day and year are numeric values with the length two, month is a character string with the length three.

Author(s)

Bastian Wiessner

Examples

```
xpssDate()
```

xpssDate11 *Todays date*

Description

R implementation of the SPSS `$date11` system variable.

Usage

```
xpssDate11()
```

Details

Provides the SPSS format of dates

Value

Current date as "dd-mmm-yyyy". Day is a numeric with the length two, month is a character string with the length three and year is a numeric value with the length four.

Author(s)

Bastia Wiessner

Examples

```
xpssDate11()
```

---

xpssDeleteVariables *Deletes variables from dataset*

Description

R implementation of the SPSS `DELETE VARIABLES` argument.

Usage

```
xpssDeleteVariables(x, variables = NULL)
```

Arguments

`x`	input data.
`variables`	atomic character or character vector with the name of the variables.

Value

Output is the narrowed dataset.

Author(s)

Bastian Wiessner

See Also

Related Functions `drop` `subset`

Examples

```
data(fromXPSS)
xpssDeleteVariables(fromXPSS, variables = "V1")
```

`xpssDescriptives` *Simple descriptive statistics*

Description

R Implementation of the SPSS Function `Descriptives`

Usage

```
xpssDescriptives(x, variables, missing = "variable", statistics = c("mean",
  "max", "min", "stddev"), save = FALSE, ztrans = list(varname = NULL, zname
  = NULL))
```

Arguments

`x`	a (non-empty) data.frame or input data of class `"xpssFrame"`.
`variables`	atomic character or character vector with the name of the variables.
`missing`	atomic character which specifiy the missing method. The method indicates what should happen when the data contains NAs. Default is `"variable"`.
`statistics`	atomic chracter or character vector which determine the descriptiv statistics. Default are `"mean"`, `"max"`, `"min"`, `"stddev"`.
`save`	logical indicator. TRUE adds the z-score of each variable to `x`. Default is FALSE.
`ztrans`	list which specifies variables for z-transformation and name of z-transformed variables. Read Details for further information.

Details

The xpssDescriptives function provides a set of descriptive statistic tools.

`missing:`

`variable`	removes user-, and system-missing data explicitly for every variable.
`listwise`	performs a listwise-deletion.
`include`	includes all user-defined missing values.

`statistics:`

`kurtosis`	calculates the bulge of the variable.
`max`	displays the maximum of the variable.
`mean`	calculates the arithmetic mean, respectively the midpoint of the variable.
`min`	displays the minimum of the variable.
`kurtosis`	calculates the bulge of the variable.
`range`	displays the span between the minimum and the maximum value.
`sekurtosis`	calculates the standrard error of the bulge of the variable.
`semean`	displays the standard error of the arithmetic mean.

seskewness	calculates the standrard error of the inclination of the variable.
skewness	calculates the inclination of the variable.
stddev	displays the standard deviation of the variable.
sum	calculates the sum of each observation within the variable.
variance	displays the variance.

ztrans input, is a list with elements varname and zname. varname and zname are either atomic characters or character vectors.
It is necessary that either both parameters are filled or blank.

Value

Output is a list object with descriptive statistic parameters. The specific outcomes of the selected variables are stored in a list object. Every variable is stored in a different list element.

If the parameter save is TRUE, a matrix with z-transformed values will be appended at the end of the list. If ztrans is blank, the name of the matrix will be Z*varname*. Otherwise whether ztrans is not empty the user specified description in zname will be the name of the z-transformed matrix of the variable varname.

Author(s)

Bastian Wiessner

Examples

```
data(fromXPSS)

## Analyzing Variable V5, Output contains default statistics
xpssDescriptives(x=fromXPSS,
                 variables="V5")

## Analyzing Variable V7_1, Output contains default statistics
## and z-score of Variable V7_1
xpssDescriptives(x=fromXPSS,
                 variables="V7_1",
                 save = TRUE)

## Analyzing Variable V7_2, Output contains default statistics
## and z-score of Variable V7_2 store in myZname
xpssDescriptives(x=fromXPSS,
                 variables="V7_2",
                 save = TRUE,
                 ztrans = list(varname = "V7_2",
                               zname = "myZname"))

 ## Analyzing Variable V7_2, Output contains kurtosis, skewness, semean and mean
 ## missing values are included
 ## z-score get calculated and store in myZname

  xpssDescriptives(x=fromXPSS,
```

```
variables="V7_2",
statistics=c("kurtosis",
             "skewness",
             "semean",
             "mean"),
missing="include",
save = TRUE,
ztrans = list(varname = "V7_2",
             zname = "myZname"))
```

---

xpssDoIf *Creates a DO IF - END IF subset*

Description

R implementation of the SPSS `DO IF` argument.

Usage

```
xpssDoIf(x, cond = NULL)
```

Arguments

x	a (non-empty) data.frame or input data of class `"xpssFrame"`.
cond	logical expression indicating the condition for subsetting.

Details

xpssDoIf selects cases for analysis based on one or more logical conditions. The conditions to select cases are specified in a logical expression. These logical expressions can contain relational operators, logical operators and arithmetic operations. xpssDoIf creates a subset of the actual dataset, without deleting the excluded variables, respectively without deleting the excluded values.

If its needed to modify more than one subset, every following subset get selected by `xpssElseIf`. `xpssDoIf` is working similar to `xpssElseIf`, the only difference is that `xpssDoIf` has to initiate the subsetting.

The data is subsetted until `xpssEndIf` restores the excluded data. All changes made at the subsetted data will be taken over, the excluded data will remain untouched! As noted before, those cases are not actually deleted and will be available after `xpssEndIf` restores the excluded data.

In a different way to SPSS. Not only data management functions like `xpssRecode` can be used within `xpssDoIf`, it is possible to use statistical and descriptiv functions like `xpssFrequencies` too.

NOTE: For temporary case selection, specify `xpssTemporary` before `xpssDoIf`.

Value

Output is a subset of the actual dataset under the condition of the logical expression.

Author(s)

Andreas Wygrabek

See Also

Related Functions xpssElseIf, xpssEndIf, xpssFilter, xpssSelectIf, xpssTemporary

Examples

```
data(fromXPSS)

temp <- xpssDoIf(x=fromXPSS, cond = "V3 == 1")

temp <- xpssRecode(x=temp,variables="V5",rec="lo:78 = 1; else = 2")

temp <- xpssEndIf(x=temp)
```

---

xpssElseIf *Creates a subcondition within a DO IF - END IF subset*

Description

R implementation of the SPSS DO IF argument.

Usage

```
xpssElseIf(x, cond = NULL)
```

Arguments

x	a (non-empty) data.frame or input data of class "xpssFrame".
cond	logical expression indicating the condition for subsetting.

Details

After using xpssDoIf every following subset command will be initiated by xpssElseIf. xpssElseIf is working similar to xpssDoIf. Both functions select cases for analysis based on one or more logical conditions. The conditions to select cases are specified in a logical expression. These logical expressions can contain relational operators, logical operators and arithmetic operations. xpssElseIf creates a subset of the actual dataset, without deleting the excluded variables, respectively without deleting the excluded values.

The data is subsetted until xpssEndIf restores the excluded data. All changes made at the subsetted data will be taken over, the excluded data will remain untouched! As noted before, those cases are not actually deleted and will be available after xpssEndIf restores the excluded data.

In a different way to SPSS. Not only data management functions like xpssRecode can be used within xpssElseIf, it is possible to use statistical and descriptiv functions like xpssFrequencies too.

NOTE: For temporary case selection, specify xpssTemporary before xpssDoIf.

Value

Output is a subset of the actual dataset under the condition of the logical expression.

Author(s)

Bastian Wiessner

See Also

Related Functions xpssDoIf, xpssEndIf, xpssFilter, xpssSelectIf, xpssTemporary

Examples

```
data(fromXPSS)
temp <- xpssDoIf(x=fromXPSS, cond = "V3 == 1")
temp <- xpssRecode(x=temp,variables="V5",rec="lo:78 = 1; else = 2")
temp <- xpssElseIf(x=temp, cond = "V3 == 1")
temp <- xpssRecode(x=temp,variables="V5",rec="lo:78 = 11; else = 22")
temp <- xpssEndIf(x=temp)
```

---

xpssEndIf *Ends a DO IF - END IF subset*

Description

R implementation of the SPSS END IF argument.

Usage

```
xpssEndIf(x)
```

Arguments

x	a (non-empty) data.frame or input data of class "xpssFrame".

Details

xpssEndIf determines the end of the analysis based on logical conditions via xpssDoIf. xpssEndIf merge the excluded data with the actual dataset, after xpssDoIf subsetted the data. All changes which were made until xpssEndIf will be taken over, the excluded data will remain untouched!

NOTE: For temporary case selection, specify xpssTemporary before xpssDoIf.

Value

Output is the original dataset.

Author(s)

Andreas Wygrabek

Examples

```
data(fromXPSS)

temp <- xpssDoIf(x=fromXPSS, cond = "V3 == 1")

temp <- xpssRecode(x=temp,variables="V5",rec="lo:78 = 1; else = 2")

temp <- xpssEndIf(x=temp)
```

---

xpssFilter *Creates a FILTER subset*

Description

R implementation of the SPSS FILTER argument.

Usage

```
xpssFilter(x,variable = NULL, filtervalue = 1)
```

Arguments

x	a (non-empty) data.frame or input data of class "xpssFrame".
variable	atomic character with the name of the variable.
filtervalue	atomic character or atomic numeric which contains the filtervalue.

Details

xpssFilter creates a subset of the actual dataset, without deleting the excluded variables, respectively without deleting the excluded values. After activating xpssFilter, only the subset will be used, the excluded data will be ignored for the following actions until xpssFilterOff terminates the filtering. As noted before those cases are **not actually** deleted and will be available after the filter is turned off.

Important:
All changes are used on the complete dataset, except for the function being an *data exploring* or *data analyzing* function

Type of Function	Example Function	Dataset Usage
Data Management	xpssSelectIf	Uses thecomplete dataset
Data Modifing	xpssRecode	Uses the complete dataset
Data Exploring	xpssDescriptives	Uses theworking dataset only
Data Analyzing	xpssRegression	Uses the working dataset only

Value

Output is a subset of the actual dataset under the predetermined condition of the filtervalue.

Author(s)

Bastian Wiessner

See Also

Related Functions xpssDoIf xpssFilterOff xpssSample xpssSelectIf xpssTemporary

Examples

```
data(fromXPSS)

fromXPSS <- xpssFilter(x=fromXPSS, variable = "V3", filtervalue=1)

xpssDescriptives(x=fromXPSS, variables = "V6")

xpssFilterOff(x=fromXPSS)
```

---

xpssFilterOff *Ends a FILTER subset*

Description

R implementation of the SPSS FILTER OFF Function.

Usage

```
xpssFilterOff(x)
```

Arguments

x	a (non-empty) data.frame or input data of class "xpssFrame".

Details

xpssFilterOff terminates the filtering and merges the excluded data with the actual subset of the dataset.

Important:
All changes are used on the complete dataset, except for the function beeing an *data exploring* or *data analyzing* function.

Type of Function	Example Function	Dataset Usage
Data Management	xpssSelectIf	Uses the complete dataset
Data Modifing	xpssRecode	Uses the complete dataset
Data Exploring	xpssDescriptives	Uses the working dataset only
Data Analyzing	xpssRegression	Uses the working dataset only

NOTE: For temporary case selection, specify xpssTemporary before xpssDoIf.

Value

Output is the original dataset.

Author(s)

Andreas Wygrabek

Examples

```
data(fromXPSS)

temp <- xpssDoIf(x=fromXPSS, cond = "V3 == 1")

temp <- xpssRecode(x=temp,variables="V5",rec="lo:78 = 1; else = 2")

temp <- xpssEndIf(x=temp)
```

xpssFlip *Flips variables*

Description

R Implementation of the SPSS FLIP Function.

Usage

```
xpssFlip(x, variables = "all", names = NULL)
```

Arguments

x	a (non-empty) data.frame or input data of class "xpssFrame".
variables	atomic character or character vector with the names of the variables to flip
names	atomic character with the name of the variable for coloumn names.

Value

A flipped, respectively transposed xpssFrame object.

Author(s)

Bastian Wiessner

See Also

t

Examples

```
data(fromXPSS)
xpssFlip(x=fromXPSS,variables=c("V4","V5","V6"),names="V1")
```

xpssFrame *Creates a xpssFrame Object*

Description

xpssFrame creates a dataset as an xpssFrame object from a local file.

Usage

```
xpssFrame(x, ...)
```

Arguments

x	as character string with the name of the file.
...	Arguments to pass on read.spss() from foreign.

Details

x the input data should be of the format *.sav*.

The SPSS dataset contains the following attributes: `names`, `codepage`, `row.names`, `FILTER`, `TEMPORAY`, `SPLIT_FILE`, `DO_IF`, `SELECT_IF`, `WEIGHTS`, `class`

Attribute	Type	Contain
`names`	atomic character or character vector	Name of the variable in the dataset
`codepage`	atomic numeric	ANSI code, which describe the encoding
`row.names`	atomic numeric or numeric vector	row ID's
`FILTER`	atomic logical or condition	Filtercondition or FALSE if `xpssFilter` is off
`TEMPORARY`	atomic logical	Logical statement if `xpssTemporary` is on or off
`SPLIT_FILE`	atomic logical	Split-File condition or FALSE if `xpssSplitFile` is off
`DO_IF`	atomic logical or condition	Do-If-condition or FALSE if `xpssDoIf` is off
`SELECT_IF`	atomic logical or condition	Select-If-condition or FALSE if `xpssSelectIf` is off
`WEIGHTS`	atomic character	* will be implemented in a following update
`class`	atomic character or character	atomic or vector of names of classes the datasets inherits from.

Variable attributes are stored in the variable itself. A variable can have the following attributes: `defined.MIS`, `MIS`, `value.labels`, `variable.label`, `varname`

Attribute	Type	Contain
`defined.MIS`	Atomic numerics or atomic characters, respectively a numeric vector or character vector	Values whic
`MIS`	list with user-defined missings	POS contai
`value.labels`	Named numeric or named character	Value and la
`variable.label`	Atomic character	Label of the
`varname`	Atomic character	Name of the

Author(s)

Andreas Wygrabek

See Also

`read.spss` `as.xpssFrame`

Examples

```
## Not run:
data <- xpssFrame(x="Testdata_1.sav")

## End(Not run)
```

xpssFrequencies *Simple descriptive statistics*

Description

R Implementation of the SPSS Function FREQUENCIES.

Usage

```
xpssFrequencies(x, variables, missing = NULL, barchart = list(plot = FALSE,
  min = NULL, max = NULL, freq = NULL, percent = NULL), piechart = list(plot =
  FALSE, min = NULL, max = NULL, freq = NULL, percent = NULL, missing = FALSE),
  histogram = list(plot = FALSE, min = NULL, max = NULL, freq = NULL, percent
  = NULL, normal = FALSE), ntiles = NULL, percentiles = NULL,
  statistics = c("mean", "stddev", "minimum", "maximum"))
```

Arguments

x	a (non-empty) data.frame or input data of class "xpssFrame".
variables	atomic character or character vector with the name of the variables.
missing	atomic character which specifiy the missing method. The method indicates what should happen when the data contains NAs. Default is "NULL". Optionally it is possible to include user-defined missing values with "include".
barchart	plot a barchart. Default for plot is NULL. If plot is TRUE an default barchart will be plotted. Optional for customized barchart a list with the following arguments has to be assigned minimum(n), maximum(n) to bound lower and upper values which are not plotted. See notes for more.
piechart	plot a piechart. Default for plot is NULL. If plot is TRUE an default an default piechart will be plotted. Optional for customized piechart a list with the following arguments has to be assigned minimum(n), maximum(n) to bound lower and upper values which are not plotted. See notes for more.
histogram	plot a histogram. Default for plot is NULL. If plot is TRUE an default histogram will be plotted. Optional for customized histogram a list with the following arguments has to be assigned minimum(n), maximum(n) to bound lower and upper values which are not plotted. With normal a overlapping normal distrubtion line will drawn. Default is FALSE. See notes for more.
ntiles	divides the distribution in a specific percentage amount of categories. multiple dividing in distributions is allowed. Default is NULL.
percentiles	displays the value between customized percentiles. Default is NULL.
statistics	Method which enumerate the descriptive statistics. Default is mean, stddev, minimum, maximum. Optional arguments are all,kurtosis, median, mode, none, range, sekurt, semean, seskew, skewness, sum, variance.

Details

The xpssFrenquencies function provides a set of descriptive statistic tools. The function delivers frequency tables containing value labels, values, frequencies, percentages of the selected variables in the dataset. Furthermore, xpssFrequency supplies three types of visualization of categorical or continous numerical:

1. barchart
2. histogram
3. piechart

It is possible to customize the graphics by indiviual parameters. If TRUE is set, the default graphic will be plotted.

`individual graphic parameter (for all charts):`

`max=n`	Cut the amount of maximum till n elements.
`min=n`	Cut the amount of n till minimum elements.
`freq=n`	Displays the distrubtion in absolute values on the basis of a user-defined maxima, the maxima has to be highe
`percent=n`	Displays the distrubtion in relative values on the basis of a user-defined maxima, the maxima has to be higher

`individual graphic parameter (for histogram):`

`normal=T`	Draws a overlapping normal curve.

`individual graphic parameter (for piechart):`

`missing=T`	Displays or excludes Missing Values.

`statistics:`

`kurtosis`	calculates the bulge of the variable.
`maxixmum`	displays the maximum of the variable.
`mean`	calculates the arithmetic mean.
`median`	calculates the median.
`minimum`	displays the minimum of the variable.
`mode`	displays the modal value of the variable.
`none`	displays no statistics.
`range`	displays the span between the minimum and the maximum value.
`sekurtosis`	calculates the standrard error of the bulge of the variable.
`semean`	displays the standard error of the arithmetic mean.
`seskewness`	calculates the standrard error of the inclination of the variable.
`skewness`	calculates the inclination of the variable.
`stddev`	displays the standard deviation of the variable.
`sum`	calculates the sum of each observation within the variable.
`variance`	displays the variance.

Author(s)

Bastian Wiessner

Examples

```
data(fromXPSS)
xpssFrequencies(x=fromXPSS,
 variables=c("V5"))

xpssFrequencies(x=fromXPSS,
 variables=c("V3","V7_2"),
 ntiles=c(0.25,0.3),
 percentiles=c(0.23,0.46,0.88))

xpssFrequencies(x=fromXPSS,
 variables=c("V3","V7_2"),
 histogram=list(plot=TRUE))

 xpssFrequencies(x=fromXPSS,
 variables=c("V3"),
 piechart=list(plot=TRUE,min=0,max=2))

 xpssFrequencies(x=fromXPSS,
 variables=c("V3"),
 barchart=list(plot=TRUE,precent=50))
```

---

xpssJDate *Days since 14. Oct 1582*

Description

Count the number of days since October 14, 1582.

Usage

```
xpssJDate()
```

Value

Returns days since 14 Oct 1582

Author(s)

Bastian Wiessner

Examples

```
xpssJDate()
```

xpssList *Displays the content of variables.*

Description

R implementation of the SPSS LIST Function

Usage

```
xpssList(x, variables = colnames(x), cases = list(from = 1, to = nrow(x), by = 1))
```

Arguments

x	a (non-empty) data.frame or input data of class xpssFrame.
variables	atomic character or character vector with the names of the variables.
cases	list containing the arguments from, to, by. All parameters are atomic numerics. See Details for more.

Details

LIST displays the content of selected variables. It is possible to display a sequence with the cases argument.from determine the begin of the sequence, to determine the end of the sequence. by determine the increment of the sequence.

Value

A data.frame with case values for specified variables in the dataset. If cases and variables are not specified, List return the complete dataset. If cases are specified the output is a user-defined sequence.

Author(s)

Bastian Wiessner

Examples

```
data(fromXPSS)

xpssList(x=fromXPSS)

xpssList(x=fromXPSS,
   variables = "V1")

xpssList(x=fromXPSS,
   variables = c("V1","V2"))


xpssList(x=fromXPSS,
```

```
variables = span(fromXPSS,
             from="V1",
             to="V4"),
cases =list(from=2,
            to=18,
            by=2))
```

---

xpssMeans *Simple descriptive statistics*

Description

R Implementation of the SPSS Function MEANS.

Usage

```
xpssMeans(x, variables = NULL, by = NULL, missing = NULL,
  cells = "default", statistics = NULL)
```

Arguments

x	a (non-empty) data.frame or input data of class "xpssFrame".
variables	atomic character or character vector with the name of the variables.
by	atomic character or character vector with the name of the variables.
missing	atomic numeric with the name of the missing method. Default is NULL. Optionally table,dependent or include can be chosen. See note for more.
cells	specifies descriptiv statistics for the results. Default is mean, stddev and n. See notes for more.
statistics	specifies a anova or linearity test for each result. Default is NULL. Optionally anova can be chosen.

Details

The xpssMeans function displays by default the mean, standard deviation and the amount of observations for a numeric dependent variable. and group counts for a string variable within groups defined by one or more control (independent) variables. Other procedures that display univariate statistics are SUMMARIZE, FREQUENCIES, and DESCRIPTIVES.

missing:

table	Deletes cases tablewise.
include	Include user-missing values.
dependent	Exclude user-missing values for dependent variables only.

statistics:

anova	calculates sumsquare, degrees of freedom, meansquare, f-value and significance level.

cells:

all	calculates all following descriptiv functions.
count	displays the amount of observations.
first	displays the first observation.
geometric	displays the geometric mean
harmonic	displays the harmonic mean
kurt	calculates the bulge of the variable.
last	displays the last observation.
max	displays the maximum of the variable.
mean	calculates the arithmetic mean, respectively the midpoint of the variable.
median	calculates the median of the variable.
min	displays the minimum of the variable.
range	displays the span between the minimum and the maximum value.
sekurtosis	calculates the standrard error of the bulge of the variable.
semean	displays the standard error of the arithmetic mean.
seskewness	calculates the standrard error of the inclination of the variable.
skew	calculates the inclination of the variable.
stddev	displays the standard deviation of the variable.
sum	calculates the sum of each observation within the variable.
variance	displays the variance.

Author(s)

Bastian Wiessner

Examples

```
# mean of variable V3
xpssMeans(x=fromXPSS,variables="V3")

# mean of variable V3 and V4
xpssMeans(x=fromXPSS,variables=c("V3","V4"))

# mean of variable V3 and V6 by V4
xpssMeans(x=fromXPSS,variables=c("V3","V6"),by="V4")

# Filtering by V4 and calculate Mean of Variable V6 by V3 and V6_kl3
fromXPSS <- xpssFilter(x=fromXPSS,variable="V4",filtervalue=1)
xpssMeans(x=fromXPSS,variables="V6",by=c("V3","V6_kl3"))
fromXPSS <- xpssFilterOff(fromXPSS)
```

---

xpssMissingValues *Defines missing values for variables.*

Description

R implementation of the SPSS MISSING VALUES function. xpssMissingValues defines values as missing and replaces them with NA. Position and Value are stored in the attributes of the specific variables.

Usage

```
xpssMissingValues(x, variables = NULL,
as.missing = list(range = c(from=NULL,to=NULL),singlevalues = NULL),
append = FALSE)
```

Arguments

x	a (non-empty) data.frame, data.table object or input data of class "xpssFrame".
variables	atomic character or character vector with the names of the variables.
as.missing	numeric list containing range and singlevalues.
append	logical. Indicating, if the existing missings should get overwritten or not.
range	numeric vector containing a missing range from i to n.
singlevalues	atomic numeric or numeric vectors containing singlevalues which determine missing values.

Details

xpssMissingValues specifies values for missing data for the selected variables. Those variables which match the terms of beeing a missing data get treated as NA. In most cases, variables which contain NA receive a special treatment in data management, case selection, and descriptive, respectively inductive statistics. User-missing values and system-missing values get treated as exactly one kind of missing data. The only difference in those missing values are that system missings get automatically assigned by the program when no legal value can be produced (e.g. character input at a numeric varibale, failed datatransformation) and user-defined missings, which are missing user data (e.g. the respondent forgot to answer, or skipped the question).

Common is that this empty spaces are filled with *-9 till -999* (for e.g. refusal to respond, inability to respond, Non-contact).

The as.missing statement indicates the handling of values which are matched by the as.missing statement. Input format is a list with the arguments range to determine a range of values with the arguments from and to as NA or singlevalues to specify one more singlevalues as missing.

NOTE: The special arguments lo and hi can be used to determine the lowest and highest value of a numeric value, wheter a missing range gets indexed.

Value

a xpssFrame object with NAs located at the position where the specified values in as.missing used to be. In the attributes of the object the position and the value itself is stored.

Author(s)

Andreas Wygrabek

Examples

```
data(fromXPSS)

fromXPSS <- xpssMissingValues(fromXPSS,
variables = "V6",
as.missing = list(range=c(from="lo",
to=45)))

fromXPSS <- xpssMissingValues(fromXPSS,
variables = "V3",
as.missing = list(singlevalues=c(1,
2)))

fromXPSS <- xpssMissingValues(fromXPSS,
variables = "V6",
as.missing = list(singlevalues="lo",
range=c(from="50",
to="hi")))
```

---

xpssNofCases *Selects the first n rows of the dataset*

Description

R implementation of the SPSS `N OF CASES` argument.

Usage

```
xpssNofCases(x, n = NULL)
```

Arguments

x	input data.
n	atomic numeric with the value of n.

Details

`xpssNofCases` can be used to select via command the first N cases in the data file. xpssNofCases permanently modifies the data set. **NOTE:** For temporary case selection, specify `xpssTemporary` before `xpssNofCases`.

Value

Output is the narrowed dataset.

Author(s)

Bastian Wiessner

See Also

Related Functions drop subset

Examples

```
data(fromXPSS)
xpssNofCases(fromXPSS, n = 10)
```

---

xpssNumeric *Creates numeric variables*

Description

R implementation of the SPSS NUMERIC function. Creates new numeric variables, which get appended at the end of the dataset.

Usage

```
xpssNumeric(x, varname = NULL, fill = NA)
```

Arguments

x	a (non-empty) data.frame or input data of class "xpssFrame".
varname	atomic character or character vector with the name of the variables which should be created.
fill	atomic numeric or atomic character values, which fill the variables. By default the value is NA for all new variables. It is possible to assign each new variable an own value to fill with.

Details

xpssNumeric creates new numeric variables, which get appended at the end of the dataset. The new variables are as long as the selected dataset. By default the new variables are blank and get filled with NA, otherwise every case for the selected variable get filled with the speficied value.

Value

Returns the input data extended by the new variables.

Author(s)

Andreas Wygrabek

See Also

xpssString

Examples

```
## Not run:
xpssNumeric(fromXPSS, varname = c("A"), fill = c(NA))

## End(Not run)
```

---

xpssRecode *Recodes variables*

Description

R implementation of the SPSS RECODE Function. xpssRecode recodes atomics or vectors of the format numeric, character or factor under the terms of recode specifications.

Usage

```
xpssRecode(x, variables, rec = NULL, varout = NULL)
```

Arguments

x	a (non-empty) data.frame or input data of class "xpssFrame".
variables	atomic character or character vector with the names of the variabless to recode.
rec	character string with recoding specifications: for more informations see details.
varout	atomic character or character vector with the names of new variables.

Details

The input of the recoding is a character string with the recoding procedure seperated with a semicolon and a optional else statement.

single data transformation**:** rec = "1 = 99; else = test"

For a numeric vector transformation**:** rec = "c(1,2,3) = 1; else = 11"

For a character vector transformation**:** rec = "c('A','B') = 'AB'; c('C','D') = 'CD'; else = 'ZZ'"

For a range of values**:** rec = "lo:10 = 1; 11:22 = 2; 23:hi = 3; else = 'copy'".

NOTE: lo and hi are special values and determine the lowest and highest value of a numeric variable.
The ":"-Operator differs in this context from the sequence operator. In xpssRecode it specifies the range from A to B. F.e. 1:10 defines the range from 1 till 10, all values which are within this range get recoded.

The else statement indicates the handling of the values which are not selected by the recoding statement, this statement matches all unspecified values, including missing values.
System default, if no else statement is given, is else='copy'.

`else='copy'`	overwrites all unmatched values with the original value.
`else='NA'`	overwrites all unmatched values in the new dataset with `NA`.
`else='Other'`	overwrites all unmatched values with Other, **only** possible for character values.
`else=99`	overwrites all unmatched values with 99, **only** possible for numeric values.

`varout` determines whether a new variable with the recoded values should appended at the end of the dataset.

Value

A xpssFrame with the recoded variables.

Author(s)

Andreas Wygrabek

Examples

```
data(fromXPSS)
fromXPSS <- xpssRecode(fromXPSS,
           variables ="V1",
           rec="'Audi' = 'Porsche'; else= copy",
           varout = NULL)

fromXPSS <- xpssRecode(fromXPSS,
                  variables = c("V5","V7_2"),
                  rec = "lo:50 =1; 51:200=2; 201:hi=3; else = copy",
                  varout =c("V5_new","V7_new"))

fromXPSS <- xpssRecode(fromXPSS,
                  variables = c("V6_kl3","V7_2"),
                  rec = "sysmis = 99",
                  varout =c("V6_new","V7_new"))
```

---

xpssRegression *Calculates a linear Regression*

Description

R Implementation of the SPSS `REGRESSION` Function.xpssRegression calculates linear regressions with associated statistics and plots.

Usage

```
xpssRegression(x, variables = NULL, dependent, method = enter(),
missing = "listwise", statistics = c("COEFF", "OUTS", "R", "ANOVA"), origin = FALSE)
```

Arguments

x	a (non-empty) data.frame, data.table object or input data of class "xpssFrame".
variables	vector with independent Variables as characters
dependent	dependent Variable as character
method	regression method to apply. Currently only `enter()` is implemented
missing	Method to handle missing values. Currently only listwise is implemented
statistics	character vector, which statistics should be in the output. Currently the function will return standard `lm` output. Only output for ANOVA is optional
origin	Should the constant be compressed?

Details

Implementation from SPSS Regression in R

Value

returns a list of lists, each applied method is its own list with regression, anova and beta-coeffizients as elements

Author(s)

Martin Schneider

Examples

```
data(fromXPSS)

xpssRegression(x = fromXPSS,
  variables = c("V7_1","V7_2"),
  dependent = "V5",
  method = list(enter()))
```

---

xpssRenameVariables *Renaming Variables*

Description

R implementation of the SPSS `RENAME VARIABLES` function. xpssRenameVariables renames variables within an exisiting data.frame or xpssFrame object.

Usage

```
xpssRenameVariables(x, oldVarNames = NULL, newVarNames = NULL)
```

Arguments

x	a (non-empty) data.frame or input data of class "xpssFrame".
oldVarNames	atomic character or character vector with the names of the variables to rename. oldVarNames must be a variable that already exists in the data set.
newVarNames	atomic character or character vector with the new variable names.

Details

Modifies names of one or more variables within a selected dataset. The arguments oldVarNames and newVarNames must have the same length.

Value

Returns the data with the renamed variables.

Author(s)

Andreas Wygrabek

Examples

```
data(fromXPSS)

xpssRenameVariables(fromXPSS,
oldVarNames= c("V1", "V2", "V3"),
newVarNames= c("Manufacturer", " Car Type", "Country"))
```

---

xpssSample *Creates a sample*

Description

R Implementation of the SPSS SAMPLE argument. Takes a sample from a xpssFrame object, data frame or matrix.

Usage

```
xpssSample(x, pct = NULL, n = NULL, from = NULL)
```

Arguments

x	a (non-empty) data.frame or input data of class "xpssFrame".
pct	atomic numeric, determines the percentage to keep.
n	atomic numeric, specifies the number of cases to keep.
from	atomic numeric, indicates the Basis for n.

Details

xpssSample takes a sample of the specified size from the elements of x either with or without replacement. The subset get specified by pct or n.

`pct` specifies a percentage value, for the amount of data which should be kept, allowed value range is from 0.01 to 1.
`n` indicates the amount of values to keep.
`from` determines the basis for `n`. `from` has to be higher then `n`.

Value

Returns a subset of the actual dataset.

Author(s)

Andreas Wygrabek

See Also

`sample`

Examples

```
data(fromXPSS)

xpssSample(fromXPSS, pct = 0.5)
```

---

xpssSelectIf *Creates a subset of cases*

Description

R Implementation of the SPSS `SELECT IF` argument. xpssSelectIf permanently selects cases for analysis based on logical conditions.

Usage

```
xpssSelectIf(x, cond = NULL)
```

Arguments

x	a (non-empty) data.frame or input data of class `"xpssFrame"`.
cond	logical expression for subsetting the data.

Details

The condition to select cases are specified in a logical expression. These logical expressions can contain relational operators, logical operators and arithmetic operations.

NOTE: For temporary case selection, specify `xpssTemporary` before `SELECT IF`.

Value

Returns a subset of the actual dataset under the condition of the logical expression.

Author(s)

Andreas Wygrabek

See Also

Related Functions `xpssDoIf`, `xpssFilter`, `xpssTemporary`

Examples

```
data(fromXPSS)

temp <- xpssSelectIf(x=fromXPSS, cond = "V3 == 1")

temp <- xpssSelectIf(x=fromXPSS, cond="V4 == 1 & V7_1 < 200")
```

---

xpssSortCases *Sorts data ascending or descending order*

Description

R implementation of the SPSS `SORT CASES` argument. xpssSortCases reorders the sequence of cases in the dataset based on the values of one or more variables.

Usage

```
xpssSortCases(x, variables = NULL, order ="A")
```

Arguments

`x`	a (non-empty) data.frame or input data of class "xpssFrame".
`variables`	atomic character or character vector with the names of the variables. Also `rownames` can be used to sort the data.
`order`	atomic character or character vector containing either "A" for ascending order or "D" for descending order.

Details

The argument order has to be of the same length as the argument variables. Optionally, the sorting can be specified in ascending or descending order for any variable. It is also possible to use combinations of ascending and descending order for different variables.

Value

Returns a sorted xpssFrame.

Author(s)

Andreas Wygrabek

See Also

`sort` `order`

Examples

```
data(fromXPSS)

xpssSortCases(fromXPSS, variables = c("V4", "V7_1", "V7_2"), order = c("A","D","A"))
```

---

xpssSortVariables *Sorting variables*

Description

R implementation of the SPSS `SORT VARIABLES` function.

Usage

```
xpssSortVariables(x, by = NULL, order = "A")
```

Arguments

`x`	a (non-empty) data.frame or input data of class "xpssFrame".
`by`	atomic character with the name of the argument to sort by
`order`	atomic character which indicate the sort direction.

Details

x can be sorted based on the following by arguments.

- *NAME.* Sort by variable names. Primary sort is alphabetical. Trailing digits are sorted numerically
- *TYPE.* Sort variables by type (numeric or string). Sort string variables by width.
- *LABEL.* Sort variables alphabetical by variable labels.
- *COLUMNS.* Sort variables by column width.

Valid input for order are "Up" or "Down", respectively "A" or "D".

Value

Returns x sorted by the by argument.

Author(s)

Benjamin Piest

See Also

sort

Examples

```
data(fromXPSS)

xpssSortVariables(fromXPSS, by = "NAME")

xpssSortVariables(fromXPSS, by = "NAME", order = "D")

xpssSortVariables(fromXPSS, by = "TYPE", order = "A")

xpssSortVariables(fromXPSS, by = "TYPE", order = "D")

xpssSortVariables(fromXPSS, by = "COLUMNS", order = "A")

xpssSortVariables(fromXPSS, by = "COLUMNS", order = "D")
```

---

xpssSplitFile *Splits the data.frame into pieces by a factor*

Description

R implementation of the SPSS SPLIT FILE argument

Usage

```
xpssSplitFile(x, by = NULL, type = "layered", attachSplit = FALSE)
```

Arguments

x	a (non-empty) data.frame or input data of class "xpssFrame".
by	factor variable which splits the data frame.
type	character vector containing, either seperated or layered. Specifies the results from an analysis function applied on a splitted data frame. Default is "layered" design.
attachSplit	logical. Indicating if the splitted data should get attached to the object as an attribute. Default is "FALSE".

Value

The function return the input data with modified attributes.

Author(s)

Bastian Wiessner

See Also

xpssDoIf xpssFilter xpssSelectIf xpssSplitFileOff xpssTemporary

---

xpssSplitFileOff *Set the data split off*

Description

R implementation of the SPSS SPLIT FILE argument

Usage

```
xpssSplitFileOff(x)
```

Arguments

x	a (non-empty) data.frame or input data of class "xpssFrame".

Value

The function return the input data with modified attributes.

Author(s)

Bastian Wiessner

See Also

xpssSplitFile xpssFilterOff xpssTemporary

xpssString *Creates string variables*

Description

R implementation of the SPSS STRING function. Creates new string variables, which get appended at the end of the dataset.

Usage

```
xpssString(x, varname = NULL, fill = NA)
```

Arguments

x	a (non-empty) data.frame or input data of class "xpssFrame".
varname	atomic character or character vector with the names of the variables which should be created.
fill	atomic numeric or atomic character values to fill the variables. By default the value is NA for all new variables. It is possible to assign each new variable an own value to fill with.

Details

xpssString creates new string variables, which get appended at the end of the dataset. The new variables are as long as the selected dataset. By default the new variables are blank and get filled with NA, otherwise every case for the selected variable gets filled with the filled value.

Value

Returns the input data extended by the new variables

Author(s)

Andreas Wygrabek

See Also

xpssNumeric

Examples

```
data(fromXPSS)
xpssString(fromXPSS, varname = c("D","E"), fill = c("placeholder",NA))
```

xpssTemporary *Temporary modification of the data*

Description

R implementation of the SPSS TEMPORARY Function.

Usage

```
xpssTemporary(x)
```

Arguments

x	a (non-empty) data.frame or input data of class "xpssFrame".

Details

xpssTemporary signals the beginning of temporary transformation. Only the following data-management procedure takes affect on the data: , xpssCount, xpssDoIf, xpssFilter, xpssMissingValues, xpssNumeric, xpssNofCases, xpssRecode, xpssSelectIf, xpssSample, xpssString, xpssVariableLabels, xpssValueLabels,.

All the changes that are made are temporary. After the next modification the data is restored.
For example: all created variables, e.g. numeric or string variables created while xpssTemporary is in effect are temporary variables!
Any changes or modifications made to existing variables while the xpssTemporary command is in effect are also **temporary**!
Any variables which are created or modified after this procedure are again permanent.

Function Status	Effect on created variables	Effect on modified variables
Temporary ON	Any created variable is temporary.	Any change on a variables is temporary.
Temporary OFF	Any created variable is permanent.	Any change on a variables is permanent.

The xpssTemporary function allows analyses for subgroups without affecting the data and then repeat the analysis for the file as a whole.

Author(s)

Andreas Wygrabek

See Also

Related Functions xpssDoIf xpssFilter xpssSample xpssSelectIf

Examples

```
data(fromXPSS)
obj <- xpssTemporary(fromXPSS)
```

xpssTime *Minutes since 14. Oct 1582*

Description

R implementation of the SPSS `$TIME` function. Count of the number of minutes since October 14, 1582 (the first day of the Gregorian calendar).

Usage

```
xpssTime()
```

Value

Returns minutes since 14 Oct 1582

Author(s)

Bastian Wiessner

xpssTtest *Performs a T-Test*

Description

R implementation of the SPSS `T-TEST` function.

Usage

```
xpssTtest(x, variables = NULL, t_test = "testval", testval = NULL,
  criteria = 0.95, groupvar = NULL, groups = NULL, withvars = NULL,
  paired = FALSE, missing = "analysis")
```

Arguments

`x`	a (non-empty) data.frame or input data of class `"xpssFrame"`.
`variables`	atomic character or character vector with the names of the variables.
`t_test`	atomic character definies the type of t-test. Default is `testval`, a one-sample t-test. Optional arguments are `groups` for an independent-sample test and `pairs` for an paires-sample test
`testval`	atomic numeric which indicates the value of mean difference.
`criteria`	atomic numeric which specifies the confidence interval for the mean differences. Default is `"0.95"`, optionally a customized value between 0 and 1 can be used.
`groupvar`	atomic character with the name of the variable which shall be used for grouping.

groups	factor variable which specify the variables grouped for an independentsample t-test.
withvars	atomic characters or character vector with the name of the paired variables which shall be used for compare the means. Optionally the argument with can be chosen to compare the means of 2 pairs.
paired	logical. Indicating whether the comparison should be pair based or not.
missing	atomic character determines the method which indicates what should happen when the data contains NAs. Default is analysis. Optionally include oder listwise can be used.

Details

Performs a Student's T-Test. The xpssTtest compares the mean by calculating Students-t of the selected distributions.

It is possible to check the samples.

1. against a specific value (one-sample) -> testval
2. in difference of groups (independent-sample)-> groups
3. in difference of variables (paired-sample) -> pairs

Simple statistics will be printed with every t-test. At the one-sample test, the mean difference will be visualized with the statistics
At the independend-sample test, the mean differnce and ANOVA will be visualized with the statistics
At the paired-sample test, the mean diference and a correlation statistic will be visualizied with the statistics

Value

returns a list depending upon the t-test.

All t-test contain:

statistics	simple statistics.
parameter	degrees of freedom.
p.value	significance level.
conf.int	confidence bound.
null.value	value of null hypothesis.
alternative	value of alternative hypothesis.
method	character string with the name of the t-test.
data.name	name of the data.

The independent t-test includes additonally:

anova	anova of the groups.

The paired t-test includes additonally:

corr	correlation of the pairs.

Author(s)

Bastian Wiessner

Examples

```
data(fromXPSS)


xpssTtest(fromXPSS,
                  variables = "V7_2",
                  t_test = "testval",
                  testval= 50,
                  criteria = 0.65)

xpssTtest(fromXPSS,
                  t_test = "pairs",
                  variables=c("V5",
                              "V6",
                              "V7_1",
                              "V7_2"),
                  paired = FALSE,
                  missing = "analysis",
                  criteria = 0.85)

xpssTtest(fromXPSS,
                  variables = "V7_2",
                  t_test = "groups",
                  groupvar = "V3",
                  groups = c(1, 2),
                  missing = "analysis",
                  criteria = 0.99)
```

---

xpssValueLabels *Modifies value labels*

Description

R implementation of the SPSS `VALUE LABEL` function. xpssValueLabels creates value labels for specific variables. The values of the label get stored in attributes of the variable.

Usage

```
xpssValueLabels(x, variables = NULL, values = NULL, labels = NULL,
  datevariables = NULL, datevalues = NULL, datelabels = NULL)
```

Arguments

x	a (non-empty) data.frame or input data of class "xpssFrame".
variables	atomic character or character vector with the names of the variables.
values	atomic numeric or numeric vector containing the values of the variable.
labels	atomic character or character vector containing the variable labels.
datevariables	atomic date or date vector with the names of the date variables.
datevalues	atomic date or date vector containing the value of the date, the values has to be like the old date format.
datelabels	atomic character or character vector containing the date labels.

Details

The SPSS variables are stored at the variable itself.
In contrast to xpssAddValueLabels , xpssValueLabels does erase existing value labels.
If the value label for a specific variable already exists, all value labels for that variable get overwritten.
If the value label for a specific variable does not exist, the value label gets created and all existing value labels for that variable get deleted.

A variable can have the following attributes: value.labels, defined.MIS, MIS, varname, variable.label

Author(s)

Andreas Wygrabek

See Also

xpssAddValueLabels xpssVariableLabels

Examples

```
data(fromXPSS)

temp <- xpssValueLabels(fromXPSS,
                            variables = "V1",
                            value = 1 ,
                            label = "Label1")

fromXPSS <- xpssAddValueLabels(fromXPSS,
                            variables = "V1",
                            values = c("A","B"),
                            labels = c("Label1","Label2"))
```

xpssVariableLabels *Modifies variable labels*

Description

R implementation of the SPSS VARIABLE LABEL function. Changing the label of a variable. In the structure of xpss-data the variable label is an attribute of each variable.

Usage

```
xpssVariableLabels(x, variables = NULL, labels = NULL)
```

Arguments

x	a (non-empty) data.frame or input data of class "xpssFrame".
variables	atomic character or character vector with the names of the variable(s).
labels	atomic character of character vector with labels for the specified variables in variables. The labels are associated in order of appearence of the variables.

Value

Input Data with modified attribute variable label.

Author(s)

Andreas Wygrabek

See Also

attributes attr xpssValueLabels

Examples

```
data(fromXPSS)

daten <- xpssVariableLabels(fromXPSS, c("V4", "V7_1"), c("Label1", "Label2"))
```

xpssVarsToCases *Transforms variables to cases*

Description

Creates a transfromed xpssFrame.

Usage

```
xpssVarsToCases(x, from, idVar = NULL, indexVar = NULL, nullArg = "keep",
countVar = NULL, varLabels = list(id = NULL, index = NULL, count = NULL))
```

Arguments

x	as a (non-empty) data.frame or input data of class "xpssFrame".
from	variable that opens the span.
idVar	determines whether an id-variable should be created.
indexVar	determines whether an index-variable should be created.
nullArg	Can be either "keep" or "drop".
countVar	determines whether a counter should be created?
varLabels	determines whether labels for id-, index- and count variables are set.

Value

Returns the transformed xpssFrame.

Author(s)

Andreas Wygrabek

Examples

```
data(fromXPSS)

xpssVarsToCases(fromXPSS, from = list(c("newVar", "V7_1, V7_2")),
idVar = "myID", indexVar = "myIndex", nullArg = "drop", countVar = "Counter")
```

Index