Package 'exceldata'

April 4, 2022

Type Package

Title Streamline Data Import, Cleaning and Recoding from 'Excel'

Version 0.1.1.2

Description A small group of functions to read in a data dictionary and the corresponding data table from 'Excel' and to automate the cleaning, re-coding and creation of simple calculated variables. This package was designed to be a companion to the macro-enabled 'Excel' template available on the GitHub site, but works with any similarly-formatted 'Excel' data.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.1.2

Imports dplyr, ggplot2, graphics, lubridate, readxl, scales

Suggests knitr, rmarkdown

NeedsCompilation no

Author Lisa Avery [aut, cre, cph] (<https://orcid.org/0000-0002-8431-5143>)

Maintainer Lisa Avery <lisa.avery@uhn.ca>

Repository CRAN

Date/Publication 2022-04-04 18:30:02 UTC

R topics documented:

addFactorVariables 2
checkData 2
createCalculated 4
createCategorisedVar 4
createCombinedVar 5
createRecodedVar 6
createSurvVar 6
importCodes 7
importExcelData 7
plotVariables 9
readDataDict 10
readExcelData 11

Index 13

`addFactorVariables` *Create factor variables from data dictionary*

Description

This function will replace the code and category variables with factors based on the factor levels provided in the data dictionary. The original variables are retained with the suffix '_orig'

Usage

```
addFactorVariables(data, dictionary, keepOriginal = TRUE)
```

Arguments

`data`	A data frame returned by readExcelData
`dictionary`	A data frame returned by readDataDict
`keepOriginal`	Boolean indicating if the original character variables should be kept, default is TRUE with _original appended to variable names

Value

A data frame with the updated factor variables

Examples

```
## Not run:
exampleDataFile <- system.file("extdata", "exampleData.xlsx", package = "exceldata")
dictionary <- readDataDict(exampleDataFile, dictionarySheet = 'DataDictionary')
data <- readExcelData(exampleDataFile,dictionary,dataSheet='DataEntry')
factorData <- addFactorVariables(data,dictionary,keepOriginal = TRUE)

## End(Not run)
```

`checkData` *Check the entered data against the data dictionary*

Description

This function compares the data in the data entry table against the specifications in the dictionary

Usage

```
checkData(dictionary, data, id)
```

Arguments

dictionary	A data frame returned by readDataDict
data	A data frame returned by readExcelData
id	String indicating the ID variable, to display errors by ID instead of row number

Details

Prior to reading in the data, the dictionary must be imported using readDataDict and the data must be imported using readExcelData.

The function will check all variables in the dictionary. If variables are missing from the dictionary an error will occur. If variables are missing from the data table a warning will be shown.

Value

A list with various reports of errors and duplicates

- errors_by_row - A data frame with errors by rownumber, or ID if supplied
- errors_by_variable - A data frame containing all errors by variable
- duplicated_entries - A string containing a list of duplicated entries
- error_dataframe - A data frame containing all the rows and columns with errors and Boolean values indicating if the entry is an error

Examples

```
## Not run:
exampleDataFile <- system.file("extdata", "exampleData.xlsx", package = "exceldata")

dictionary <- readDataDict(exampleDataFile, dictionarySheet = 'DataDictionary')
data <- readExcelData(exampleDataFile,dictionary,dataSheet='DataEntry')
checks <- checkData(dictionary,data,'ID')

exampleDataFile <- system.file("extdata", "exampleData_withErrors.xlsx", package = "exceldata")

dictionary <- readDataDict(exampleDataFile, dictionarySheet = 'DataDictionary')
data <- readExcelData(exampleDataFile,dictionary,dataSheet='DataEntry')
checks <- checkData(dictionary,data,'ID')

checks

## End(Not run)
```

createCalculated *Create calculated variables*

Description

This function will create survival and recoded variables according to the rules in the dictionary.xlsm file. See the Example sheet for an example.

Usage

```
createCalculated(data, dictionary, timeUnit = "month")
```

Arguments

data	A data frame data returned by the importExcelData or readExcelData functions
dictionary	A data frame returned by the importExcelData or readDataDict functions
timeUnit	String containing the desired unit of time for survival variables

Value

A data frame with the calculated variables as specified by the dictionary

Examples

```
## Not run:
exampleDataFile <- system.file("extdata", "exampleData.xlsx", package = "exceldata")
dictionary <- readDataDict(exampleDataFile, dictionarySheet = 'DataDictionary')
data <- readExcelData(exampleDataFile,dictionary,dataSheet='DataEntry')
factorData <- addFactorVariables(data,dictionary,keepOriginal = TRUE)
fullData <- createCalculated(factorData,dictionary,timeUnit='month')

## End(Not run)
```

createCategorisedVar *Create categories from continuous data*

Description

This function will create categories based on the ranges provided in the instructions

Usage

```
createCategorisedVar(data, newVarName, instructions)
```

Arguments

data	is data returned by the importExcelData or readExcelData functions
newVarName	the name of the new variable. Must be empty in data
instructions	category names and bounds

---

createCombinedVar *Create a combined variable from several dichotomous variables*

Description

This function will create a single variable from a set of dichotomous variables, usually checkbox items from a survey. The combined variable may be if there are a small number of popular response patterns. Currently this function only works with dichotomous variables.

Usage

```
createCombinedVar(data, dictionary, newVarName, varsToCombine, responseVal)
```

Arguments

data	is data returned by the importExcelData or readExcelData functions
dictionary	is the data dictionary returned by importExcelData or readDataDict functions
newVarName	the name of the new variable.
varsToCombine	a vector of the first and last variables to combine into the new variable. Note that the variables to be conbined mut be contiguous in the data.
responseVal	the value of the variables to be combined, usually this will be 1 for 0,1 variables or Yes for Yes No or Checked for Checked Unchecked

Details

The instructions are contained in the Levels column of the data dictionary and should be in the format: original_varname,newCode1=oldcode1,oldCode2,...,newCode2=oldCode3,..

For Example: instructions="T_Stage,T0=T0,T1up=T1,T2,T3,T4"

will recode T1-T4 as T1up and retain T0 as is

createRecodedVar *Create survival variables (survival duration + status)*

Description

This function will create survival variables from an existing start variable date of event variable and last date followed variable

Usage

```
createRecodedVar(data, dictionary, newVarName, instructions)
```

Arguments

data	is data returned by the importExcelData or readExcelData functions
dictionary	is the data dictionary returned by importExcelData or readDataDict functions
newVarName	the name of the new variable.
instructions	are from the data dictionary

Details

The instructions are contained in the Levels column of the data dictionary and should be in the format:

original_varname,newCode1=oldcode1,oldCode2,...,newCode2=oldCode3,..

For Example: instructions="T_Stage,T0=T0,T1up=T1,T2,T3,T4"

will recode T1-T4 as T1up and retain T0 as is

createSurvVar *Create survival variables (survival duration + status)*

Description

This function will create survival variables from an existing start variable date of event variable and last date followed variable

Usage

```
createSurvVar(data, newVarName, survVars, timeUnit = "month")
```

Arguments

data	A data frame returned by the importExcelData or readExcelData functions
newVarName	the name of the new survival variable. The status variable will be suffixed with '_status'
survVars	In order the start date, event date and date of last followup
timeUnit	Character, the unit of time to calculate survival variables for (ex: 'day' 'week' 'month' 'year')

---

importCodes *Return A data frame of codes*

Description

Accepts a string input in the form "code1=label1,code2=label2,.." and returns A data frame with a column of codes and a column of labels

Usage

```
importCodes(labelStr, delim = ",")
```

Arguments

labelStr	in the format code1=label1,code2=label2
delim	delimeter separating codes in labelStr, defaults to ','

---

importExcelData *Import Excel Data based on the specifications in a data dictionary*

Description

This function reads in a data dictionary and data entry table and converts code and category variables to factors as outlined in the dictionary. See the examples.

Usage

```
importExcelData(
  excelFile,
  dictionarySheet = "DataDictionary",
  dataSheet = "DataEntry",
  id,
  saveWarnings = TRUE,
  setErrorsMissing = TRUE,
  range,
  colnames,
  origin,
  timeUnit = "month"
)
```

Arguments

excelFile	path and filename of the data file containing the data and dictionary
dictionarySheet	name of the sheet containing the data dictionary, defaults to 'DataDictionary'
dataSheet	name of the data entry sheet within the file, defaults to 'DataEntry'
id	String indicating the ID variable, to display errors by ID instead of row number
saveWarnings	Boolean, if TRUE and there are any warnings then the function will return a list with the data frame and the import warnings
setErrorsMissing	Boolean, if TRUE all values out of range will be set to NA
range	Optional, Range of Excel sheet to restrict import to (ie. range="A1:F6")
colnames	Optional, Column names of the dictionary, defaults to those used in the Excel template: c('VariableName', 'Description (optional)', 'Type', 'Minimum', 'Maximum', 'Levels')
origin	Optional, the date origin of Excel dates, defaults to 30 December 1899
timeUnit	Character specifying the unit of time that should be used when creating survival type variables. Allowed values are from lubridate (ex: 'day' 'week' 'month' 'year')

Details

The exceldata package was designed around the DataDictionary.xlsm template. More documentation and the current downloadable template can be found at:

https://github.com/biostatsPMH/exceldata#readme Note that as of release 0.1.1.1 the log file will give row numbers corresponding to the row number in Excel, as opposed to the row number in the data frame

Warning: If SetErrorsMissing = TRUE then a subsequent call to checkData will not return any errors, because the errors have already been set to missing.

NOTE: This function will only read in those columns present in the DataDictionary

Value

A list containing two data frames: the data dictionary and the data table

- dictionary - A data frame with entries for each variable
- data - A data frame containing the imported data

Examples

```
exampleDataFile <- system.file("extdata", "exampleData.xlsx", package = "exceldata")
import <- importExcelData(exampleDataFile,
dictionarySheet = 'DataDictionary',dataSheet = 'DataEntry')

# The imported data dictionary
dictionary <- import$dictionary
head(dictionary)
```

```
# The imported data, with calculated variables
data <- import$data
head(data)

# Simple univariate plots with outliers
plots <- plotVariables(data=data,dictionary=dictionary,IDvar = 'ID')
```

---

plotVariables *Return a list of univariate ggplots for each non-character variable*

Description

This function should be run as the final step after the data have been imported, checked and the factor variables created.

Usage

```
plotVariables(data, dictionary, IDvar, vars, showOutliers = TRUE)
```

Arguments

data	A data frame containing the variables to be plotted
dictionary	Optional, the data dictionary returned by importExcelData or readDataDict functions to provide plot titles
IDvar	Optional string indicating the name of an identifying variable to highlight outliers
vars	Optional, vector of the names of variables to plot
showOutliers	Boolean, Defaults to TRUE. Should outliers be labelled? Outliers are defined by the 1.5xIQR rule (as with boxplots)

Value

A list of plots with one plot for each variable

Examples

```
## Not run:
exampleDataFile <- system.file("extdata", "exampleData.xlsx", package = "exceldata")
import <- importExcelData(exampleDataFile,
dictionarySheet = 'DataDictionary',dataSheet = 'DataEntry')
dictionary <- import$dictionary
data <- import$data

# Simple univariate plots with outliers
plots <- plotVariables(data=data,dictionary=dictionary,IDvar = 'ID')
plots
```

```
## End(Not run)
```

`readDataDict` *Read in the data dictionary*

Description

This function reads in a data dictionary from an Excel file, based on the [DataDictionary.xlsm template](https://github.com/biostatsPMH/exceldata#readme)

Usage

```
readDataDict(
  excelFile,
  dictionarySheet = "DataDictionary",
  range,
  colnames,
  origin
)
```

Arguments

`excelFile`	Character, Path and Name of the data file
`dictionarySheet`	Character, Name of the dictionary sheet within the file, defaults to 'DataDictionary'
`range`	Optional, Range of Excel sheet to restrict import to (ie. range="A1:F6")
`colnames`	Optional, Column names of the dictionary, defaults to those used in the Excel template: c('VariableName', 'Description (optional)', 'Type', 'Minimum', 'Maximum', 'Levels')
`origin`	Optional, the date origin of Excel dates, defaults to 30 December 1899

Details

It assumes that the columns names have not been altered and are: c('VariableName', 'Description (optional)', 'Type', 'Minimum', 'Maximum', 'Levels')

To override these column names specify colnames as an argument, ensuring that the content of the columns is in the above order. As of the time of writing, the origin date in Excel is 30 December 1899. To override this specify origin="yyy-mm-dd"

To read in only part of the excel sheet specify the desired range (ie range="A1:F6")

Value

A data frame with an entry for each variable to be imported

Examples

```
## Not run:
exampleDataFile <- system.file("extdata", "exampleData.xlsx", package = "exceldata")
dictionary <- readDataDict(exampleDataFile, dictionarySheet = 'DataDictionary')

## End(Not run)
```

`readExcelData` *Read Excel Data*

Description

This function reads in an excel data table created by the dictionary.xlsm template file according to the specifications in the dictionary

Usage

```
readExcelData(
  excelFile,
  dictionary,
  dataSheet = "DataEntry",
  saveWarnings = FALSE,
  setErrorsMissing = FALSE,
  range,
  origin
)
```

Arguments

`excelFile`	path and filename of the data file
`dictionary`	A data frame returned by readDataDict
`dataSheet`	name of the data entry sheet within the file, defaults to 'DataEntry'
`saveWarnings`	Boolean, if TRUE and there are any warnings then the function will return a list with the data frame and the import warnings
`setErrorsMissing`	Boolean, if TRUE all values out of range will be set to NA
`range`	Optional, Range of Excel sheet to restrict import to (ie. range="A1:F6")
`origin`	Optional, the date origin of Excel dates, defaults to 30 December 1899

Details

Prior to reading in the data, the dictionary file must be imported using readDataDict.

Warning: If SetErrorsMissing = TRUE then a subsequent call to checkData will not return any errors, because the errors have been set to missing.

NOTE: This function will only read in those columns present in the dictionary

Value

A data frame containing the imported data

Examples

```
## Not run:
exampleDataFile <- system.file("extdata", "exampleData.xlsx", package = "exceldata")
dictionary <- readDataDict(exampleDataFile, dictionarySheet = 'DataDictionary')
data <- readExcelData(exampleDataFile,dictionary,dataSheet='DataEntry')

## End(Not run)
```

Index

addFactorVariables, 2

checkData, 2
createCalculated, 4
createCategorisedVar, 4
createCombinedVar, 5
createRecodedVar, 6
createSurvVar, 6

importCodes, 7
importExcelData, 7

plotVariables, 9

readDataDict, 10
readExcelData, 11