Package 'ImportExport'

April 3, 2025

Type Package

Title Import and Export Data

Version 1.3.1

Date 2025-04-02

Maintainer Isaac Subirana <isubirana@imim.es>

Description Import and export data from the most common statistical formats by using R functions that guarantee the least loss of the data information, giving special attention to the date variables and the labelled ones.

Depends gdata, Hmisc, chron, RODBC

Imports readxl, writexl, haven, utils

Suggests shiny, shinyBS, shinythemes, compareGroups, foreign

License GPL (>= 2)

NeedsCompilation no

Author Roger Pros [aut],
Isaac Subirana [aut, cre],
Joan Vila [ctb]

Repository CRAN

Date/Publication 2025-04-03 08:10:11 UTC

Contents

`ImportExport-package` *Import and Export Data*

Description

Import and export data from the most common statistical formats by using R functions that guarantee the least loss of the data information, giving special attention to the date variables and the labelled ones.

The package also includes an usefull shiny app called by `ImportExportApp` which uses all the content of the package to import and export databases in a rather easy way.

Details

The DESCRIPTION file:

Package:	ImportExport
Type:	Package
Title:	Import and Export Data
Version:	1.3.1
Date:	2025-04-02
Authors@R:	c(person("Roger", "Pros", role=c("aut")), person("Isaac", "Subirana", role=c("aut", "cre"), email="isubirana@i
Maintainer:	Isaac Subirana <isubirana@imim.es>
Description:	Import and export data from the most common statistical formats by using R functions that guarantee the least
Depends:	gdata, Hmisc, chron, RODBC
Imports:	readxl, writexl, haven, utils
Suggests:	shiny, shinyBS, shinythemes, compareGroups, foreign
License:	GPL (>= 2)
Author:	Roger Pros [aut], Isaac Subirana [aut, cre], Joan Vila [ctb]

Index of help topics:

Author(s)

Roger Pros [aut], Isaac Subirana [aut, cre], Joan Vila [ctb]

Maintainer: Isaac Subirana <isubirana@imim.es>

See Also

ImportExportApp

Examples

```
## Not run:
x<-spss_import("mydata.sav")

## End(Not run)
```

---

access_export *Export multiple R data sets to Microsoft Office Access*

Description

Directly connect (and disconnect at the end) with the Microssoft Office Access database using the **RODBC** package and write one or multiple data sets.

Usage

```
access_export(file,x,tablename=as.character(1:length(x)),uid="",pwd="",...)
```

Arguments

file	The path to the file with .mdb extension.
x	Either a data frame or a list containing multiple data frame to be exported.
tablename	A character or a vector character containing the names that will receive the tables where the data frame is stored. If it is a vector, it must follow the same order as the data frames in x have. All names must be different from each others.
uid	see odbcConnect .
pwd	see odbcConnect .
...	see odbcConnect,sqlSave.

Details

Date variables are exported as an integer, they might be converted to character if a character representation in the access database is wanted.

Value

No value is returned.

Note

This function connects and writes on an existing Microsoft Office Access database, but it can't create a new one.

Examples

```
## Not run:
# x is a data.frame
file<-("mydata.xlsx")
a<- 1:10
b<-rep("b",times=10)
c<-rep(1:2,each=5)
x<-data.frame(a,b,c)
excel_export(x,file,table_names="mydata")
# x is a list
y<-list(x,x[2:3])
excel_export(y,file,table_names=c("mydata1","mydata2"))

## End(Not run)
```

---

access_import *Import tables and queries from Microssoft Office Access(.mdb)*

Description

Directly connect (and disconnect at the end) with the Microssoft Office Access database using the **RODBC** package and read one or multiple data sets. It can read both tables and SQL queries depending on the input instructions. It automatically detects date variables that are stored also with date format in the original data set.

Usage

```
access_import(file,table_names,ntab=length(table_names),
SQL_query=rep(F,times=ntab),where_sql=c(),out.format="d-m-yy",uid="",pwd="",...)
```

Arguments

`file`	The path to the file with .mdb extension.
`table_names`	A single character or a character vector containing either the names of the tables to read or the SQL queries to perform. Each position must contain only one table name or SQL querie.The format of the SQL queries must follow the one described in `sqlQuery`.
`where_sql`	If `table_names` is a vector, `where_sql` must contain the position of the SQL queries within the `table_names` vector. Ex: If the first and the fifth elements of `table_names` are SQL queries (the other ones are table names) the vector `where_sql` should be `where_sql=c(1,5)` .
`out.format`	a character specifying the format for printing the date variables.

ntab	The number of tables to import, equal to the number of table names.
SQL_query	Auxiliar vector to perform the function.
uid	see odbcConnect .
pwd	see odbcConnect .
...	see odbcConnect, sqlFetch, sqlQuery.

Details

By default, the function gives to each data set the name specified in table_names, so the sql queries data set have probably an inappropriate name. It can be easily renamed using names.

Value

A data frame or a data frame list containing the data requested from the Microsoft Office Access file.

Note

The function don't contribute in the date variables detection, it just process with the **Chron** package the ones who has been automatically detected.

See Also

access_export,var_view sqlFetch, sqlQuery

Examples

```
## Not run:
x<-access_import(file="mydata.mdb",
                    table_names=c("table1","table2",
               "Select * From table1 inner join table2 on table.var1=table2.var2","table3")
                    ,where_sql=c(3))


## End(Not run)
```

---

excel_export *Export multiple R data sets to Excel*

Description

Exports a single data frame or a list of data frames to one or multiple excel sheets using the function write_xlsx frome the **writexl** package. This function can write multiple data frames (passed as a list) with a single command .It can write both .xls and .xlsx files.

Usage

```
excel_export(x,file,table_names=as.character(1:length(x)),...)
```

Arguments

x	Either a data frame or a list containing multiple data frame to be exported.
file	The name of the file we want to create.
table_names	A character or a vector character containing the names that will receive the sheet where the data frame is stored. If it is a vector, it must follow the same order as the data frames in x have. All names must be different from each others.
...	see write_xlsx.

Value

No value is returned.

See Also

read_excel

Examples

```
## Not run:
# x is a data.frame
file<-("mydata.xlsx")
a<- 1:10
b<-rep("b",times=10)
c<-rep(1:2,each=5)
x<-data.frame(a,b,c)
excel_export(x,file,table_names="mydata")
# x is a list
y<-list(x,x[2:3])
excel_export(y,file,table_names=c("mydata1","mydata2"))

## End(Not run)
```

---

format_corrector *Identify and corrects variable formats*

Description

The function creates a loop to compare for each variable the values it have with the usual ones that typical R formats have in order to correct, for example, missing value or dates stored as a character. It also specify for each variable the most appropriate SPSS format that it should have.

Usage

```
format_corrector(table,identif=NULL,force=FALSE,rate.miss.date=0.5)
```

Arguments

table	The data set we want to correct.
identif	The name of the identification variable included in the data frame. It will be used to list the individuals who had any problems during the execution of the function.
force	If TRUE, run format_corrector even if "fixed.formats" attribute is TRUE
rate.miss.date	The maximum rate of missing date fields we want the function to accept.The function details which fields have been lost anyways.

Details

If the date variable don't have chron format it must be in one of the following formats, else the function leaves it as a character:
—-dates separator must be one of the following:("-","/",".").
—-hour separator must be ":".

Value

A single data frame which results from the function.

Note

This function may not be completely optimal so it might have problems when correcting huge data frames.

See Also

spss_export

Examples

```
require(ImportExport)
a<-c(1,NA,3,5,".")
b<-c("19/11/2006","05/10/2011","09/02/1906","22/01/1956","10/10/2010")
c<-101:105
x<-data.frame(a,b,c)
sapply(x,class)
x_corr<-format_corrector(x)
sapply(x_corr,class)
```

ImportExportApp *Runs the shiny app*

Description

Runs a shiny app which uses all the content of the package to import and export databases in a rather easy way.

Usage

```
ImportExportApp(...)
```

Arguments

...	See runApp

Details

It requires a few packages to run the app: **shiny**, **shinyBS**, **shinythemes**, **compareGroups**.

See Also

runApp

Examples

```
## Not run:
ImportExportApp()
## End(Not run)
```

spss_export *Export data to SPSS (.sav) by using runsyntx.exe or pspp.exe*

Description

Export data to txt and syntax to an spss syntax file and then runs runsyntx.exe (located in the SPSS folder) in order to create the final file with .sav extension containing the data frame we wanted to export. Date variables in the original data frame are also identified when reading the .sav file with SPSS.

Usage

```
spss_export(table,file.dict=NULL,file.save=NULL,var.keep="ALL",
file.runsyntax="C:/Archivos de programa/SPSS/runsyntx.exe",
file.data=NULL,run.spss=TRUE,dec=".")
```

Arguments

table	A data frame to be exported. If it's a matrix, it will be converted into data frame.
file.dict	Spss syntax file containing the variable and value labels.
file.save	The name of the .sav file we want to create.
var.keep	Name of the variables to save. All variables will be saved by default.
file.runsyntax	The path to the file runsyntx.exe or pspp.exe .
file.data	The name of the .txt file containing the data. It will be created as a temp file by default.
run.spss	If true, it runs SPSS and creates the .sav file, else it shows the syntax on the screen.
dec	The string to use for decimal points, it must be a single character.

Details

Both runsyntax.exe (from SPSS) and pspp.exe works the same way.

Value

No value is returned.

Note

If neither SPSS nor PSPP is installed the function can just return the data in a .txt file and the syntax in an SPSS syntax file (.sps).

See Also

spss_import,var_view

Examples

```
## Not run:
table=mydata
file.dict=NULL
file.save="C:\xxx.sav"
var.keep="ALL"
export.SPSS(table=table,file.dict=file.dict,var.keep=var.keep,file.save=file.save)


## End(Not run)
```

spss_import *Import data set from SPSS (.sav)*

Description

Read a labelled data set from SPSS, finding automatically the date variables and keeping the variable and value labels information, by using the information obtained with `spss_varlist()` and the function `spss.get` from the **Hmisc** Package.

Usage

```
spss_import(file, allow="_",out.format="d-m-yy",use.value.labels=F,...)
```

Arguments

`file`	The path to the file with .sav extension.
`allow`	A vector which contains the characters that must be allowed in the variable names.
`out.format`	A character specifying the format for printing the date variables.
`use.value.labels`	If TRUE, replace the labelled variables with their value labels.
`...`	See `spss.get`.

Details

In order to provide the maximum functionallity, if the main code generates an error, the function tries to read the file with the `read_sav` function from the **haven** package, but a warning message appears. The `var_view` function can be used to summarize the contents of the data frame labels.

Value

A data frame or a list containing the data stored in the **SPSS** file.

Note

If the warning message appears and the file has been read using `read_sav` the resulting data frame will be diferent from the expected one (see the **haven** package to learn more about `read_sav`).

Author(s)

Dave MacFarlane, Roger Pros, Isaac Subirana

See Also

`var_view`, `spss.get`, `read_sav`

Examples

```
## Not run:
x <- spss_import("mydata.sav")


## End(Not run)
```

`table_import` *Automatic separator data input*

Description

A small variation of the original `read.table` that most of the time detect automatically the field separator character. It also includes the option to run the `format_corrector` function in order to detect, for example, the date variables included in the original data set. If the function don't recognize any separator, it asks to specify the real one.

Usage

```
table_import(file,sep=F,format_corrector=F,...)
```

Arguments

`file`	The patch to he file which the data are to be read from.
`sep`	The field separator character, see `read.table`.If it is not specified, the function try to detect it automatically.
`format_corrector`	If True, it runs the `format_corrector` function before returning the data frame.
`...`	More arguments from `read.table`.

Details

The `format_corrector` function is a complicated function so it's not recommended to run it unless the data set contains awkward variables like dates.

Value

A data frame containing the data stored in the file.

Note

This function might have problems if any of the fields contain typical separators, so it's always recommended to check the resulting data frame in order to avoid possible errors.

See Also

`read.table`

Examples

```
## Not run:
x <- table_import('mydata.csv',format_corrector=T)


## End(Not run)
```

---

var_view *Summarize variable information*

Description

Creates a table with the name, the description, the value labels and the format for each variable in the data frame. It is similar to the variable view shown in the SPSS.

Usage

```
var_view(x)
```

Arguments

x	The data frame whose variables we want to summarize.

Value

A data frame containing the specified summary.

Note

This function was built in order to summarize imported SPSS labelled data sets using spss_import, but it can also work with other labelled data sets.

See Also

spss_import

Examples

```
require(ImportExport)
a<- 1:10
b<-rep("b",times=10)
c<-rep(1:2,each=5)
x<-data.frame(a,b,c)
attr(x$a,"label")<- "descr1"
attr(x$b,"label")<- NULL
attr(x$c,"label")<- "descr3"
attr(x$c,"value.labels")<-list("1"="Yes","2"="No")
var_view(x)
```

Index