

Package ‘WriteXLS’

February 14, 2012

Version 2.1.0

Date 2010-09-18

Title Cross-platform Perl based R function to create Excel 2003 (XLS) files

Description Cross-platform Perl based R function to create Excel 2003 (XLS) files from one or more data frames. Each data frame will be written to a separate named worksheet in the Excel spreadsheet. The worksheet name will be the name of the data frame it contains or can be specified by the user.

Author Marc Schwartz <marc_schwartz@me.com>

Maintainer Marc Schwartz <marc_schwartz@me.com>

License GPL (>= 2)

SystemRequirements Perl, Encode, Text::CSV_XS

URL <http://r-forge.r-project.org/projects/writexls/>

Repository CRAN

Date/Publication 2010-09-19 18:30:07

R topics documented:

testPerl	2
WriteXLS	3
Index	7

`testPerl`*Test Perl installation and required modules for WriteXLS()*

Description

Test Perl installation and required modules for WriteXLS()

Usage

```
testPerl(perl = "perl", verbose = TRUE)
```

Arguments

<code>perl</code>	Name of the perl executable to be called.
<code>verbose</code>	Output test result messages.

Details

This function will test your current system to be sure that Perl is installed and if so, whether or not all of the Perl modules required for WriteXLS() are present.

Success and/or error messages as appropriate will be output.

Value

A boolean value (TRUE or FALSE). TRUE if Perl and ALL required modules are found

Note

Please be sure to read the included INSTALL file (in the main package installation directory) for additional details on meeting the requirements for Perl and the additional Perl modules that are necessary for WriteXLS to work properly. The file includes platform specific recommendations for common scenarios. The path to the package installation directory can be located using `.path.package("WriteXLS")` after using `library(WriteXLS)`.

A working installed version of Perl must be present in the current system searchpath or the exact path of the perl executable must be provided via the `perl` argument. Perl modules required for this function that may not be part of a default Perl installation are included with this package. These modules include:

OLE::Storage_Lite, Parse::RecDescent, Getopt::Long, File::Basename and Spreadsheet::WriteExcel

Note that the required Perl modules Encode and Text::CSV_XS are not included with this package. They are platform specific, requiring local compilation and installation from CPAN or via your OS package manager.

Author(s)

Marc Schwartz <marc_schwartz@me.com>

Many thanks to Prof. Brian Ripley for his assistance in the testing of this package.

See Also[WriteXLS](#)

WriteXLS*Cross-platform Perl based R function to create Excel 2003 (XLS) files*

Description

Writes one or more R data frames to an Excel 2003 file

Usage

```
WriteXLS(x, ExcelFileName = "R.xls", SheetNames = NULL, perl = "perl",
        verbose = FALSE, Encoding = c("UTF-8", "latin1"),
        row.names = FALSE,
        AdjWidth = FALSE, AutoFilter = FALSE, BoldHeaderRow = FALSE,
        FreezeRow = 0, FreezeCol = 0,
        envir = parent.frame())
```

Arguments

<code>x</code>	A character vector containing either the names of one or more R data frames, or the single name of a list containing one or more R data frames, to be exported to the Excel file.
<code>ExcelFileName</code>	The name of the Excel file to be created. Must be a valid Excel filename. May include an existing path.
<code>SheetNames</code>	A character vector containing the names of each worksheet to be created. If NULL (the default), the names of the dataframes will be used instead. Worksheet names may be up to 31 characters in length and must be unique. If specified, <code>length(SheetNames)</code> must be the same as <code>length(x)</code> . NOTE: The order of the names here must match the order of the data frames as listed in <code>x</code> .
<code>perl</code>	Name of the perl executable to be called.
<code>verbose</code>	Output step-by-step status messages during the creation of the Excel file. Default is FALSE.
<code>Encoding</code>	Define the character encoding to be used for the exported data frames. Defaults to UTF-8.
<code>row.names</code>	If TRUE, the row names of the data frames are included in the Excel file worksheets.
<code>AdjWidth</code>	If TRUE, will adjust the worksheet column widths based upon the longest entry in each column. This is approximate.
<code>AutoFilter</code>	If TRUE, will add autofiltering to each column in each worksheet. Note that not all spreadsheet applications support this feature.
<code>BoldHeaderRow</code>	If TRUE, will apply a bold font to the header row for each worksheet.

FreezeRow	Rows including this row and above this row will be frozen and not scroll. The default value of 0 will scroll the entire sheet. Note that not all spreadsheet applications support this feature.
FreezeCol	Columns including this column and to the left of this column will be frozen and not scroll. The default value of 0 will scroll the entire sheet. Note that not all spreadsheet applications support this feature.
envir	The environment in which to look for the data frames named in <code>x</code> . This defaults to the environment in which <code>WriteXLS</code> was called.

Details

This function takes a character vector containing the names of one or more R data frames and writes them to an Excel 2003 spreadsheet file. Each data frame will be written to a separate worksheet in the Excel file.

The order of the worksheets created in the Excel file will match the order of the entries in `x`.

The actual creation of the Excel file is performed by a Perl script called `WriteXLS.pl`, which is included with this package.

Note that the named Excel file, if it already exists, will be overwritten and no warning is given. In addition, if the file exists and is open by another application (eg. Excel, OO.org, etc.) you will likely get an error message regarding the inability to open the file and/or that the file is already in use by another application or user. Errors can also occur if the file has been marked as read-only or if your access rights do not allow you to overwrite the file or write to the folder you have indicated in the path to the file.

There is an intermediate step, where the R data frames are first written to CSV files using `write.table` before being written to the Excel file by the Perl script. `tempdir` is used to determine the current R session temporary directory and a new sub-directory called "WriteXLS" will be created there. The CSV files will be written to that directory and both the files and the directory will be deleted prior to the function terminating normally using `on.exit`. It is possible that these will remain in place if this function terminates abnormally or is aborted prior to completion.

As `write.table` is used to write the data frames to CSV files, the data types supported by `write.table` will be exported to their character representation correctly. For other data types, it is recommended that you first coerce them to character columns formatted as you require and then use `WriteXLS` to create the Excel file.

All of the CSV files will be created prior to the creation of the Excel file as the Perl script will loop over them as part of the process. Thus, sufficient free disk space must be available for these files and the Excel file at the same time.

A text file called "SheetNames.txt" will be created in the same temporary directory as the CSV files. This file will contain the sheet names, one per line and will be used by the Perl script to name the worksheets in the Excel file.

Each worksheet will be named using either the names in `SheetNames`, the names of the data frames in `x` or the names of the list elements if `x` is a list (up to the first 31 characters, which is an Excel limitation). If any the data frame names specified in `x` are longer than 31 characters, they will be truncated to 31 characters. `SheetNames` if specified, will be checked to make sure that all of the entries are less than or equal to 31 characters. If not, an error message will be displayed.

Note that the order of the names in `SheetNames` MUST match the order of the data frames named in `x`.

Note that the worksheets must have unique names. Thus, if `SheetNames` is `NULL`, the data frame names will be checked to be sure that they are unique up through the first 31 characters. If `SheetNames` is specified, the entries will be checked to be sure that they are unique. If not, an error message will be displayed.

Note that the following characters are not allowed for Excel worksheet names: `[]:*\?/\`

The data frame column names will be exported "as is" and will be the first row in the corresponding worksheet.

UTF-8 encoded content in the data frame should be properly exported using the Perl Encode module by default. If you are operating in a 'latin1' based locale (also known as iso-8859-1), set `Encoding` to 'latin1'.

Note that arguments `AdjWidth`, `AutoFilter`, `BoldHeaderRow`, `FreezeRow` and `FreezeCol` will apply to ALL worksheets exported.

Value

TRUE if the Excel file was successfully created. FALSE if any errors occurred.

Note

Please be sure to read the included `INSTALL` file (in the main package installation directory) for additional details on meeting the requirements for Perl and the additional Perl modules that are necessary for this function to work properly. The file includes platform specific recommendations for common scenarios. The path to the package installation directory can be located using `.path.package("WriteXLS")` after using `library(WriteXLS)`.

A working installed version of Perl must be present in the current system searchpath or the exact path of the perl executable must be provided via the `perl` argument. Perl modules required for this function that may not be part of a default Perl installation are included with this package. These modules include:

`OLE::Storage_Lite`, `Parse::RecDescent`, `Getopt::Long`, `File::Basename` and `Spreadsheet::WriteExcel`

Note that the required Perl modules `Encode` and `Text::CSV_XS` are not included with this package. They are platform specific, contain C source code requiring compilation and installation from CPAN or via your OS/Perl package manager to utilize pre-packaged binary versions.

To test your Perl installation and verify that all required Perl modules are available, use the `testPerl` function provided in this package.

Author(s)

Marc Schwartz <marc_schwartz@me.com>

Many thanks to Prof. Brian Ripley for his assistance in the testing of this package.

References

Spreadsheet::WriteExcel Perl Module <http://search.cpan.org/dist/Spreadsheet-WriteExcel>

Excel 2003 Specifications and Limitations <http://office.microsoft.com/en-us/excel/HP051992911033.aspx>

For Perl Unicode Issues <http://www.ahinea.com/en/tech/perl-unicode-struggle.html>

See Also

[write.table](#) and [testPerl](#)

Examples

```
# Only run the examples if Perl and all modules are present
if (testPerl(verbose = FALSE))
{
  # Examples using built-in data frames
  WriteXLS("iris", "iris.xls")

  WriteXLS(c("iris", "infert", "esoph"), "Example.xls")

  iris.split <- split(iris, iris$Species)
  WriteXLS("iris.split", "irissplit.xls")

  # Clean up and delete XLS files
  rm(iris.split)
  unlink("iris.xls")
  unlink("Example.xls")
  unlink("irissplit.xls")
}
```

Index

*Topic **file**

testPerl, [2](#)

WriteXLS, [3](#)

testPerl, [2](#), [6](#)

write.table, [6](#)

WriteXLS, [3](#), [3](#)