Package 'BrailleR'

August 10, 2018

Type Package

Title Improved Access for Blind Users

Version 0.29.1

Date 2018-08-3

Author A. Jonathan R. Godfrey [aut, cre],
Debra Warren [aut],
Donal Fitzpatrick [ctb],
Duncan Murdoch [ctb],
Greg Snow [ctb],
Henrik Bengtsson [ctb],
James Curtis [ctb],
JooYoung Seo [ctb],
Marshall Flax [ctb],
Paul Murrell [aut],
Timothy Bilton [aut],
Tony Hirst [ctb],
Tsan-Kuang Lee [ctb],
Volker Sorge [aut],
Yihui Xie [ctb]

Maintainer A. Jonathan R. Godfrey <a.j.godfrey@massey.ac.nz>

Description Blind users do not have access to the graphical output from R without printing the content of graphics windows to an embosser of some kind. This is not as immediate as is required for efficient access to statistical output. The functions here are created so that blind people can make even better use of R. This includes the text descriptions of graphs, convenience functions to replace the functionality offered in many GUI front ends, and experimental functionality for optimising graphical content to prepare it for embossing as tactile images.

Repository CRAN

License GPL-2

Depends R (>= 3.4.0)

Imports devtools, dplyr, extrafont, ggplot2, grid, gridGraphics, gridSVG, hunspell, knitr, magrittr, moments, nortest,

reticulate, rmarkdown, roloc, rolocISCCNBS, utils, whisker, XML, xtable

Suggests installr

SystemRequirements Python 2.7 and wxPython 2.8

VignetteBuilder knitr

URL https://github.com/ajrgodfrey/BrailleR

BugReports https://github.com/ajrgodfrey/BrailleR/issues

RoxygenNote 6.0.1

NeedsCompilation no

Date/Publication 2018-08-10 12:40:07 UTC

R topics documented:

`BrailleR-package` *Improved Access for Blind Users*

Description

Blind users do not have access to the graphical output from R without printing the content of graph windows to an embosser of some kind. This is not as immediate as is required for efficient access to statistical output. The functions here are created so that blind people can make even better use of R. This includes the text descriptions of graphs, convenience functions to replace the functionality offered in many GUI front ends, and experimental functionality for optimising graphical content to prepare it for embossing as tactile images.

Details

Package:	BrailleR
Type:	Package
Version:	0.29.1
Date:	2018-08-3
License:	GPL-2

Author(s)

A. Jonathan R. Godfrey Volker Sorge and Timothy P. Bilton with other contributions.

Maintainer: A. Jonathan R. Godfrey <a.j.godfrey@massey.ac.nz>

References

Godfrey, A.J.R. (2013) "Statistical Software from a Blind Person's Perspective: R is the Best, but we can make it better", The R Journal 5(1), pp73-79.

AddXMLInternal — *Internal functions for adding the necessary XML content for accessible graphs in SVG format*

Description

Mostly internal functions for adding XML content to an external file

Details

Not for use as exported functions

Value

Either the filename written to, or NULL

Author(s)

Volker Sorge and A. Jonathan R. Godfrey

AddXMLMethod — *Create XML files to sit alongside SVG files in order to make an accessible graph experience.*

Description

Creates the necessary XML file for a graph object (as long as it has a class assigned) or the current graph window.

Usage

```
AddXML(x, file)
```

Arguments

x	a graph object for which a method exists, or the current graphics device if set to NULL.
file	The XML file to be created.

Details

This is experimental work. At present, the proof of concept is based on a fairly standard histogram from the **graphics** package.

Value

NULL. This function is solely for the purpose of creating XML files in the current working directory or in a path of the user's choosing.

Author(s)

Volker Sorge and A. Jonathan R. Godfrey

References

P. Dengler et al. (2011) Scalable vector graphics (SVG) 1.1, second edition. W3C recommendation, W3C. http://www.w3.org/TR/2011/REC-XML11-20110816/

Examples

```
x=rnorm(1000)
#AddXML(hist(x))
```

AugmentMethod *add additional detail to the stored object for a graph*

Description

Creates the necessary details that feed into the text descriptions in the VI() function and into the descriptions used in the accessible online versions of the graphs.

Usage

```
Augment(x)
```

Arguments

x	a graph object for which a method exists, or the current graphics device if set to NULL.

Details

Ought to be treated as an internal function and not used interactively.

Value

The input object is returned with additions to the object. This does not break the S3 class.

Author(s)

A. Jonathan R. Godfrey and Volker Sorge

Examples

```
x=rnorm(1000)
MyHist=Augment(hist(x))
MyHist
```

AutoSpellCheck *Automatic fixing of typos*

Description

Fix up all those annoying typos that come up far too often.

Usage

```
AutoSpellCheck(file)
```

Arguments

file	A vector of files to be checked

Details

The word list of typos and their corrections is called 'AutoSpellList.csv' and is stored in the user's MyBrailleR folder. The file can be updated to meet the user's specific needs. It should not be over-written by a new installation of BrailleR.

Value

NULL. This function only affects external files.

Author(s)

A. Jonathan R. Godfrey

`boxplot` *Create a standard boxplot with a few extra elements added to the output object*

Description

This function is a wrapper to the standard `boxplot()` function in the **graphics** package. It adds detail to the stored object so that a better text description can be formulated using the `VI()` method in the **BrailleR** package.

Usage

```
boxplot(x, ...)
```

Arguments

`x`	a numeric variable.
`...`	additional arguments passed on to the plotting function.

Details

This function masks the function of the same name in the **graphics** package. The base R implementation does create an object, but does not give it a class attribute, the object does not store all graphical arguments that are passed to the `boxplot()` function. The functionality should be no different at all for anyone who is not using the `VI()` function to gain a more detailed text description of the boxplot. See the help page for the `graphics::boxplot()` function to get a more complete description of boxplot creation.

Value

An object of class boxplot.

Note

I would love to see this function become redundant. This will happen if the extra functionality is included in the `boxplot()` function in the **graphics** package. This should be possible as the user experience will not be any different, no matter if the user is blind or sighted.

Author(s)

A. Jonathan R. Godfrey

References

The problem of not including class attributes for graphs was identified in: Godfrey, A.J.R. (2013) 'Statistical Software from a Blind Person's Perspective: R is the Best, but we can make it better', The R Journal 5(1), pp73-79.

See Also

The base R implementation of the boxplot function should be consulted; see the entry in the **graphics** package

Examples

```
x=rnorm(1000)
op = par(mfcol=c(2,1))
# the stamdard boxplot function returns
MyBoxplot=graphics::boxplot(x, main="Example boxplot (graphics package)", horizontal=TRUE)
MyBoxplot

# while this version returns
MyBoxplot=boxplot(x, main="Example boxplot (BrailleR package)", horizontal=TRUE)
MyBoxplot
par(op)

# The VI() method then uses the extra information stored
VI(MyBoxplot)
```

`BrailleRUsefulLinks` *Open the home page for the BrailleR Project, the Google search engine, the BrailleR in Action book, or the Let's Use R Now (LURN) book in your browser*

Description

Visit the BrailleR Project home page for updates, instructions on how to join the mailing list which we call BlindRUG and get the latest downloads; or the Let's Use R Now (LURN) home page to read an online manual for using R.

Usage

```
BrailleRHome()

BrailleRInAction()

LURN(BlindVersion = getOption("BrailleR.VI"))

Google()
google()
```

Arguments

`BlindVersion`	Use the version of Let's Use R Now tailored to an audience of blind users.

Value

NULL. These functions are for opening a web page and nothing will result in the R session.

Author(s)

A. Jonathan R. Godfrey

`BRLThis` *Convert a graph to a pdf ready for embossing*

Description

The first argument to this function must be a call to create a graph, such as a histogram. Instead of opening a new graphics device, the graph will be created in a pdf file, with all text being presented using a braille font. The function is somewhat experimental as the best braille font is not yet confirmed, and a number of examples need to be tested on a variety of embossers before full confidence in the function is given.

Usage

```
BRLThis(x, file)
```

Arguments

`x`	the call to create a graph
`file`	A character string giving the filename where the image is to be saved.

Details

The user's chosen braille font must be installed. This might include the default font shipped as part of the package.

Value

Nothing within the R session, but a pdf file will be created in the user's working directory.

Author(s)

A. Jonathan R. Godfrey. with contributions from JooYoung Seo and TK Lee.

`BrowseSVG` *Launches a browser tab to explore SVG diagram*

Description

Creates a single HTML file that embeds an SVG diagram and its XML annotations. Then launches a browser tab to allow viewing and interactive exploration of the SVG diagram.

Usage

```
BrowseSVG(file="test", dir=".", key=TRUE, footer = TRUE, view=interactive())
```

Arguments

`file`	the filename for the HTML file; this should correspond to basename of an existing SVG and its XML annotations
`dir`	the folder into which the HTML file is written and where R tries to locate the SVG and XML files
`key`	include key for explorer's keyboard commands in webpage
`footer`	Volker, explain please.
`view`	launch in browser; this is the default when running in an interactive session

Details

An HTML file is written in the given directory and the Javascript library is copied to that location.

Value

NULL. This function exists for its side effects only.

Author(s)

Volker Sorge

References

to add following first demonstration/publication

CleanCSV *clean out unwanted white space from a csv file*

Description

A blind user may not see the white space characters surrounding text or numbers in a csv file. These corrupt analyses and are annoying to fix.

Usage

```
CleanCSV(file)
```

Arguments

file	A vector of files to be checked

Details

Spits out the csv file in clean form, as well as a back up copy of the original file.

Value

NULL. This function only affects external files.

Author(s)

A. Jonathan R. Godfrey

DataViewer *Open a data object in your chosen spreadsheet software*

Description

The chosen data object (data.frame, matrix, or vector) is stored in a temporary csv file. The file is opened while the R terminal/console is suspended until the <enter> key is pressed. This should be done once the spreadsheet software has been closed so that the temporary file is removed and the data object gets updated from the edited csv file. The user must save the csv file if changes are made.

Usage

```
DataViewer(x, Update = FALSE, New = NULL, Filename = NULL)
```

Arguments

x	an object of class data.frame, matrix, or vector. A check is made for this condition.
Update	Logical. Will changes made in the spreadsheet editor be returned to the R session.
New	If Update=TRUE, the original object will be replaced unless a name for a new object is given. N.B. the default is to overwrite the original object.
Filename	Provide a filename if desired. N.B. this does not guarantee that the csv file will be kept. Do not use this function as a shortcut for saving csv files.

Details

The following steps should be taken if the intention is to view and possibly edit a data.frame, matrix, or vector:

There are a few quirks with respect to variables that could be interpreted as factors. The process of updating is to write a csv file using code write.csv() and then read it back using code read.csv() after it has been edited. Users must be aware that factors may not register as such, or that character vectors may be interpreted as factors.

Value

If code Update=TRUE and code New=NULL, the original data object is overwritten. To create a new object the user must specify code New.

Author(s)

A. Jonathan R. Godfrey <a.j.godfrey@massey.ac.nz>

Examples

```
data(airquality)
DataViewer(airquality, Update=TRUE, New="NewAirQuality")
# remove the new object using rm()
```

---

dotplot *create a dotplot using stripchart*

Description

A method for creating dotplots. The functions call the `stripchart` command from the **graphics** package and assign the output to have the class `dotplot`.

Usage

```
dotplot(x, ...)

## S3 method for class 'formula'
dotplot(x, ...)
```

Arguments

x	a vector or formula, where the right hand side of the formula is a factor.
...	other graphical parameters including those passed to title.

Details

This function was created as a result of being unable to assign all graphical parameters that are created when a formula is used in stripchart. Users not intending to use the VI method should use stripchart instead.

Author(s)

A. Jonathan R. Godfrey

See Also

This function is dependent on the stripchart function from the **graphics** package. Consult its help page for more information.

Examples

```
VI(with(airquality, dotplot(Ozone~Month)))
```

Embossers *Prepare BrailleR settings for specific braille embossers*

Description

Convenience functions for setting package options based on experimentation using specific embossers.

Usage

```
Premier100()
```

Details

These functions are only relevant for owners of the specified embossers. Ownership of these models means the user has access to fonts that are licenced to the user.

The Premier 100 embosser uses standard 11 by 11.5 inch fanfold braille paper. Printing in landscape or portrait is possible.

Value

Nothing. The functions are only used to set package options.

Author(s)

A. Jonathan R. Godfrey.

See Also

ChooseEmbosser

Examples

```
#Premier100() #' Specify use of the Premier 100 embosser.
#ChooseEmbosser() #' reset to default: using no embosser.
```

`FindReplace` *Find/Replace text in a file*

Description

Simple wrapper functions to make it easier to replace the text in a file, possibly due to spelling errors, but perhaps to replace default text in a template file.

Usage

```
FindReplace(file, find, replace)

Rnw2Rmd(file)

UseTemplate(file, find=NULL, replace=NULL)
```

Arguments

`file`	The external (text) file or template to be updated.
`find`	The text to remove.
`replace`	The text to insert.

Details

The FindReplace function is purely intended for use on an external file whereas UseTemplate is intended to take a template file from within the BrailleR package and return the updated text to the calling environment.

Rnw2Rmd tries to replace Standard LaTeX commands and Sweave chunk headers with R markdown ones. It is NOT comprehensive, but it does get a long way towards a useful markdown file.

Obviously the specified file must exist for these functions to work.

Value

FindReplace will replace the existing file with the updated version while UseTemplate will return a character string which will usually be pushed out to an R script or R markdown file.

Author(s)

A. Jonathan R. Godfrey

Examples

```
UseTemplate("DTGroupSummary.R")
UseTemplate("DTGroupSummary.R", "DataName", "MyData")
```

`GetExampleText` *extract the example text from a help page*

Description

A cut back version of example() for obtaining the text used in examples on help pages without running those examples. The function is intended to help write markdown/Sweave documents.

Usage

```
GetExampleText(topic, package = NULL, lib.loc = NULL, character.only = FALSE, outFile="")
```

Arguments

`topic`	name or literal character string: the online help topic the examples of which should be run.
`package`	a character vector giving the package names to look into for the topic, or NULL (the default), when all packages on the search path are used.
`lib.loc`	a character vector of directory names of R libraries, or NULL. The default value of NULL corresponds to all libraries currently known. If the default is used, the loaded packages are searched before the libraries.
`character.only`	a logical indicating whether topic can be assumed to be a character string.
`outFile`	an optional filename to save the results into. The default is to use a temporary file.

Details

The example() code was hacked back to form this utility function. It is probably a little heavy for what is needed but it works sufficiently at the time of creation.

Value

a vector of character strings each element being one line of the examples from the corresponding help topic.

Author(s)

The work of Martin Maechler and others got tampered with by Jonathan Godfrey

See Also

You may wish to compare this with example

Examples

```
cat(paste(GetExampleText(mean), collapse="\n"))
```

GetGoing *Set options for using brailleR*

Description

An interactive question-and-answer interface suitable for blind users wanting to set the options for using the BrailleR package.

Usage

```
GetGoing()
```

Details

Defaults are offered for all questions so that pressing <Enter> means no changes are made. Users answer yes/no questions as TRUE or FALSE respectively; the short form T or F is also allowed.

The user can also choose to perform various setup tasks using this interface.

Value

NULL. This function is a tool for executing other functions that will set options and setup the package according to the user's wants.

Author(s)

A. Jonathan R. Godfrey

See Also

All options being set through this function have specific functions that achieve the same ends. For example, see GoSighted for the options that are binary settings, or SetAuthor for options requiring a specific character or numeric value to be chosen.

The setup functionality can be reviewed at MakeBatch.

GetWriteR *Download an executable for Windows users*

Description

Anyone wishing to make use of the WriteR application must have pandoc installed.

Users that do not have Python 2.7 and wxPython installed cannot use the WriteR application file as provided by the BrailleR package. An executable for Windows is available for separate download. In 2018, the WriteR script files were tested for use with Python3; initial testing was successful.

Downloaded files will be saved into the user's MyBrailleR folder.

Usage

```
Get7zip()

GetCygwin(x64 = TRUE)

GetPandoc()

GetPython27()
GetPython3()

GetRStudio()

GetWriteR(UseGitHub = TRUE)

GetWxPython27()
GetWxPython3()
```

Arguments

UseGitHub	Use the latest version found via GitHub or an older version found on the R-Resources.massey.ac.nz webpage.
x64	Use the 64 bit version if appropriate.

Details

This function assumes you have a current internet connection because it downloads a File. For WriteR, it is a zip file which gets unpacked in your current working directory, then The zip file is removed and the executable file is moved to your chosen MyBrailleR folder.

The other functions will download and install the chosen application. The installers are stored in the user's MyBrailleR folder.

Value

NULL. The downloaded file is saved in the user's MyBrailleR folder.

Note

Use of this function assumes you are happy for a file to be downloaded and saved on your hard drive. You can go to your MyBrailleR folder and delete the executable at any time.

Author(s)

A. Jonathan R. Godfrey, building on the installr package by Tal Galili

---

grep.VIgraph *String manipulation of the output produced by VI.ggplot*

Description

Allows the output from VI.ggplot to be searched and replaced based on a search pattern.

Usage

```
## S3 method for class 'VIgraph'
grep(pattern, x, ...)
## S3 method for class 'VIgraph'
gsub(pattern, replacement, x, ...)
```

Arguments

pattern	Regular expression for matching, as per grep
replacement	Replacement text, as per gsub
x	object returned by VI.ggplot
...	other arguments passed on to grep or gsub to control matching behaviour

Details

The **BrailleR** package redefines the grep and gsub functions as generic functions (that dispatch on the x argument), with base::grep and base::gsub as the default methods. This grep.VIgraph method behaves like base::grep with value=TRUE (i.e., it returns the matched values, not the indices).

Value

Returns a new object of the same type as that returned by VI.ggplot, but with the text component restricted to only those lines that matched the pattern or with the text component replaced.

Author(s)

Debra Warren and Paul Murrell

Examples

```
if (require(ggplot2)) {
    grep("axis", VI(qplot(1,1)))
    gsub("labels", "tick labels", VI(qplot(1,1)))
}
```

---

hist *Create a standard histogram with a few extra elements added to the output object*

Description

This function is a wrapper to the standard `hist()` function in the **graphics** package. It adds detail to the stored object so that a better text description can be formulated using the `VI()` method in the **BrailleR** package.

Usage

```
hist(x, ...)
```

Arguments

`x`	a numeric variable.
`...`	additional arguments passed on to the plotting function.

Details

This function masks the function of the same name in the **graphics** package. Even though the base R implementation does create an object of class histogram, the object does not store all graphical arguments that are passed to the `hist()` function. The functionality should be no different at all for anyone who is not using the `VI()` function to gain a more detailed text description of the histogram. See the help page for the graphics::hist() function to get a more complete description of histogram creation.

Value

An object of class histogram as per the `hist()` function from the **graphics** package, with the addition of any calls to the main title or axis labels.

Author(s)

A. Jonathan R. Godfrey

References

Godfrey, A.J.R. (2013) 'Statistical Software from a Blind Person's Perspective: R is the Best, but we can make it better', The R Journal 5(1), pp73-79.

See Also

The base R implementation of the `hist` function should be consulted, using the entry in the **graphics** package

Examples

```
x=rnorm(1000)
# the stamdard hist function returns
MyHist=graphics::hist(x, xlab="random normal values", main="Example histogram (graphics package)")
#dev.off()
MyHist

# while this version returns
MyHist=hist(x, xlab="random normal values", main="Example histogram (BrailleR package)")
#dev.off()
MyHist

# The VI() method then uses the extra information stored
VI(MyHist)
```

---

history *View the history of the current workspace*

Description

A substitute for the history function from the **utils** package.

Usage

```
history(max.show = 25, reverse = FALSE, pattern, ...)
```

Arguments

`max.show`	The maximum number of lines to show. Inf will give all of the currently available history.
`reverse`	logical. If true, the lines are shown in reverse order. Note: this is not useful when there are continuation lines.
`pattern`	A character string to be matched against the lines of the history. When supplied, only unique matching lines are shown.
`...`	Arguments to be passed to grep when doing the matching.

Details

This function exists because the standard history function in the **utils** package opens the internal pager that cannot be used by a blind person's screen reading software.

Value

Nothing is returned to the workspace. This function exists for the creation and viewing of the temporary file that holds the list of commands issued in the current workspace.

Author(s)

Duncan Murdoch, with testing by A. Jonathan R. Godfrey.

See Also

the original `history` function.

`Internal` *Internal functions for the BrailleR package*

Description

Some functions that probably weren't really necessary, but proved useful at some stage.

Usage

```
FindCSSFile(file)


InQuotes(x)

nNonMissing(x)
```

Arguments

`x`	an object that is of the form needed by the internal function concerned.
`file`	a filename to look for in the user's css folder or locally.

Value

These should be fairly obvious from the name of the function

Author(s)

A. Jonathan R. Godfrey

JoinBlindRUG *Send an email based on a template*

Description

Template email messages have been created to help you send feedback to the package creator or to join the Blind R Users Group email list. The BlindRUG email list is a forum where we can discuss issues about using R as a blind person. List traffic is light. You can join and leave as you see fit.

Usage

```
JoinBlindRUG()

ThankYou()
```

Details

You will get an email prepared in your selected email client. If you're wanting to join the BlindRUG list, just press send without editing that message at all. Pretty soon, the server will send you a message that you will need to reply to in order to finalise the subscription. Once you have joined the list you will be sent an initial welcome message.

If you are sending in feedback, then please do tell me how you heard about BrailleR and where you are studying or working.

Author(s)

A. Jonathan R. Godfrey

MakeAccessibleSVGMethod *Create matched pairs of SVG and XML files to make an accessible graph experience.*

Description

Creates the necessary SVG and XML files for a graph object (as long as it has a class assigned) or the current graph window.

Usage

```
MakeAccessibleSVG(x, file = "test", view = interactive(), ...)
```

Arguments

x	a graph object for which a method exists, or the current graphics device if set to NULL.
file	The shared name for the SVG and XML files to be created.
view	launch in browser; this is the default when in interactive session
...	arguments passed on to other methods/functions

Details

This is experimental work. Having started with the proof of concept based on a fairly standard histogram from the **graphics** package, we are now looking to the boxplots and time series plots.

Value

NULL. This function is solely for the purpose of creating SVG and XML files in the current working directory or in a path of the user's choosing.

Author(s)

A. Jonathan R. Godfrey and Volker Sorge

References

P. Dengler et al. (2011) Scalable vector graphics (SVG) 1.1, second edition. W3C recommendation, W3C. http://www.w3.org/TR/2011/REC-XML11-20110816/

Examples

```
x=rnorm(1000)
#MakeAccessibleSVG(hist(x))
```

MakeAllFormats *Prepare the options for conversion of an R markdown file.*

Description

Make a pandoc options file for use with a named R markdown file.

Usage

```
MakeAllFormats(RmdFile, BibFile = "")
```

Arguments

RmdFile	The name of the R markdown file to be converted.
BibFile	Name of the bibtex database file to include.

Details

The options are based on the current best set of possible outcomes. The two html file options use different representations of any mathematical equations. The Microsoft Word file uses the native equation format which is not accessible for blind people using screen readers.

Value

Nothing in the R console/terminal. The function is used for its side effects in the working directory.

Author(s)

A. Jonathan R. Godfrey.

MakeBatch *Create batch files for processing R scripts and markdown files under Windows*

Description

Convenience function for creating batch files that can be used under Windows to process R scripts and Rmarkdown files. It may also be possible to use them in conjunction with the Open With dialogue in Windows Explorer; this makes use of file associations so that R scripts and Rmd files are (in effect) executable.

Usage

```
MakeBatch(file=NULL)
```

Arguments

`file`	A character string for the file to be processed. The file need not yet exist. The extension must be either R or Rmd and is case sensitive.

Details

These batch files are not for use in an R session. They do need to be created in an interactive session, so that users can process R scripts and Rmarkdown files without needing to open an R session later.

If a file is specified, the function will create a single batch file that will process the file appropriately. Processing an R script will generate a Rout file, while an Rmd file is converted into HTML.

If no file is specified, the function creates various files in the current working directory.

Files starting with the word test are for testing the batch files. An R script and an Rmarkdown file were created as well as the batch files that will process them into a Rout file and an HTML document respectively. Pressing <enter> on these test*.bat files will process the test files appropriately.

The other three batch files (ending in .bat) need to be moved to a folder on the user's path so that they can be called from anywhere. They could also be manually edited to suit the user's needs.

The path.txt file shows the user what folders are already on the path list. The user can review this list and decide to alter the system variable if they so choose. The path.txt file has no value otherwise.

Further instructions

Once the RBatch.bat file has been moved to the desired folder that is included in the path for your system you can follow the following steps to get this function working fully.

1. Open windows explorer, and browse to the folder containing the test files.

2. Select the test.R script.

3. Under the File menu, look for the item Open with... (This might already be a submenu for some users; if so, the last item is Choose default program.)

4. We are going to choose to use a program on our computer. Do not go looking on the internet to see which program we need.

5. You may be able to write a description of the file type. This is an R script but it may not yet be registered as such. Providing this detail is optional.

6. When given the chance to browse for the program to open the test.R script, browse to the folder where you placed Rbatch.bat and select it.

7. When you select OK, the test.R script will be processed by RBatch.bat and a new file test.Rout will be created.

8. Open test.Rout in any text editor you like. This file has the appearance of an R session window except for some processing time detail at the end. You will be able to read the commands that were originally in test.R as well as the output from these commands.

Author(s)

A. Jonathan R. Godfrey with testing by JooYoung Seo

MakeReadable *Convert line breaks in vignette documentation*

Description

The Rnw files used for vignettes use Linux style line breaks that make reading vignette source files difficult for Windows users. A Python script is called which converts the line breaks and saves the vignette source in the user's MyBrailleR folder.

Usage

```
MakeReadable(pkg)
```

Arguments

pkg	The package to investigate for vignette source files.

Details

Must have Python 2.7 installed for this function to work.

Value

Nothing in the workspace. All files are stored in a vignettes folder within MyBrailleR.

Author(s)

A. Jonathan R. Godfrey

`MakeRmdFiles` *Work flow convenience functions*

Description

Time-saving functions that help create files in more useful formats for later use.

Usage

```
History2Rmd(file = "History.Rmd")

R2Rmd(ScriptFile)

ProcessAllMd(dir =".")
ProcessAllRmd(dir =".", method = "render")
```

Arguments

`dir`	The directory to find files.
`file`	the name of the file to be created.
`method`	The method used to process Rmd files; one of "render" or "knit2html".
`ScriptFile`	the R script to be processed into the Rmarkdown file.

Details

The History2Rmd() function was intended for turning a short interactive R session into an R markdown file. Lines of code are all separated into distinct code chunks in the Rmd file. the resulting file will need to be edited if commands have spanned more than one line.

The R2Rmd() function does try to limit the number of blank lines copied from the R script into the Rmarkdown file. The Rmd file may need some editing.

Once all Rmd files have been edited, the user can have all the Rmd files in a folder processed using ProcessAllRmd(). A similar function ProcessAllMd() exists to process any plain markdown files which do not need knitting.

Value

NULL. These functions are for creating files in the current working directory.

Author(s)

A. Jonathan R. Godfrey

See Also

These functions were inspired by the spin functionality of the **knitr** package. You may wish to move onto using it for more features.

MakeRprofile *Load BrailleR on Startup in Current Working Directory*

Description

Writes the single command "library(BrailleR)" to a .First() function in .Rprofile in the current working directory. This forces the BrailleR package to be automatically loaded when R is opened in this working directory.

Usage

```
MakeRprofile(Overwrite = FALSE)
```

Arguments

Overwrite	Logical: Should an existing .Rprofile file be overwritten?

Value

Nothing. This function is used for its side effect of creation of a file in the current working directory. A warning message is created if the file exists and Overwrite=FALSE.

Author(s)

A. Jonathan R. Godfrey.

MakeSlideShow *Turn a set of Rmd files into an HTML slide show*

Description

Takes a set of Rmd files in alphabetical order and makes them a set of linked HTML files or a single slidy presentation suitable for delivering a presentation., or a single plain HTML file suitable for distribution

Usage

```
MakeAllInOneSlide(Folder, Style = getOption("BrailleR.SlideStyle"), file=NULL)
MakeSlidy(Folder, file = NULL)
MakeSlideShow(Folder, Style = getOption("BrailleR.SlideStyle"), ContentsSlide = TRUE)
```

Arguments

`Folder`	name of the folder in the current working directory that contains the set of Rmd files.
`Style`	Choose the cascading style sheet file to be applied. The file is looked for in three places: Your MyBrailleR folder, the current working directory, and finally the folder with the Rmd files. The last one found is the one that gets used if multiple copies exist.
`file`	The filename to use for the Rmd and HTML files. These files will be overwritten if they already exist.
`ContentsSlide`	Do you want links for the extra slide that has links to all slides? The file will be called 00_Contents.html

Details

For MakeSlideShow: The files are temporarily moved up to the current working directory where the links for the back, next and optionally the contents slides are added, before these new files are knitted to HTML. The user then opens the first file in the set and has a series of next links to work with that gets them through the slide show. All temporary files are then deleted.

For MakeAllInOneSlide: The process is similar but the links are not added and only one HTML file is created.

Ultimately, the user should move to using the slidy presentation.

Value

A printout of the slides made in the order they will appear. The HTML files are in the current working directory.

Note

The official CRAN policies tell package developers that we cannot write files to the user's hard drive without their permission or notifying them that this is happening. Well, you've just been told. If you use this function, you will get files written to the hard drive on your computer, but you did want them.

Author(s)

A. Jonathan R. Godfrey

NewFunction *Create a template for a new function*

Description

An R script for a new function is created in the working directory. It includes Roxygen commented lines for the documentation with sensible defaults where possible.

Usage

```
NewFunction(FunctionName, args = NULL, NArgs = 0)
```

Arguments

FunctionName	The name of the function and file to create.
args	a vector of argument names
NArgs	an integer number of arguments to assign to the function.

Details

A file is saved in the current working directory that has a template for a function with a set of arguments (if supplied). The file still needs serious editing before insertion into a package.

Value

No objects are created in the workspace. The only outcome is the template file and a message to let the user know the job was completed.

Author(s)

A. Jonathan R. Godfrey.

Examples

```
NewFunction("MyTest")
```

`OneFactor` *Analysis for a continuous response for one group factor*

Description

A convenience function that creates an analysis for a continuous response variable with one grouping factor. The function creates a number of graphs and tables relevant for the analysis.

Usage

```
OneFactor(Response, Factor, Data = NULL, HSD = TRUE, AlphaE = 0.05,
Filename = NULL, Folder = NULL,
VI = getOption("BrailleR.VI"), Latex = getOption("BrailleR.Latex"),
View = getOption("BrailleR.View"))
```

Arguments

`Response`	Name of the continuous response variable.
`Factor`	The grouping factor.
`Data`	The data.frame that contains both the response and the factor.
`HSD`	Logical: Should Tukey's HSD be evaluated for the data?
`AlphaE`	The family-wise Type I error rate for Tukey's HSD calculations.
`Filename`	Name of the Rmarkdown and HTML files to be created. A default will be created that uses the names of the variables if this is left set to NULL.
`Folder`	Name of the folder to store graph and LaTeX files. A default will be created based on the name of the data.frame being used.
`VI`	Logical: Should the VI method for blind users be employed?
`Latex`	Logical: Should the tabulated sections be saved in LaTeX format?
`View`	Logical: Should the HTML file be opened for inspection?

Details

This function writes an R markdown file that is knitted into HTML and purled into an R script. All graphs are saved in subdirectories in png, eps, pdf and svg formats. Tabulated results are stored in files suitable for importing into LaTeX documents.

Value

This function is used for creation of the files saved in the working directory and a few of its subdirectories.

Author(s)

A. Jonathan R. Godfrey and Timothy P. Bilton

See Also

Other convenience functions can be investigated via their help pages. See UniDesc, OnePredictor, and TwoFactors

Examples

```
data(airquality)

# the following line  returns an error:
## OneFactor("Ozone", "Month", airquality, View=FALSE)
# so we make a copy of the data.frame, and fix that:

airquality2 = airquality
airquality2$Month = as.factor(airquality$Month)
# and now all is good to try:
OneFactor("Ozone", "Month", airquality2)
# N.B. Various files and a folder were created in the working directory.
# Please investigate them to see how this function worked.
```

`OnePredictor` *Exploration of the relationship between a response and a single predictor*

Description

A convenience function for generating exploratory graphs and numeric summaries, regression analysis output, and the residual analysis of the simple linear model.

Usage

```
OnePredictor(Response, Predictor, Data = NULL,
    Filename = NULL, Folder = NULL,
    VI = getOption("BrailleR.VI"), Latex = getOption("BrailleR.Latex"),
   View = getOption("BrailleR.View"))
```

Arguments

`Response`	Name of the continuous response variable.
`Predictor`	Name of the continuous response variable.
`Data`	The data.frame that contains both the response and the factor.
`Filename`	Name of the Rmarkdown and HTML files to be created. A default will be created that uses the names of the variables if this is left set to NULL.
`Folder`	Name of the folder to store graph and LaTeX files. A default will be created based on the name of the data.frame being used.
`VI`	Logical: Should the VI method for blind users be employed?
`Latex`	Logical: Should the tabulated sections be saved in LaTeX format?
`View`	Logical: Should the HTML file be opened for inspection?

Details

This function writes an R markdown file that is knitted into HTML and purled into an R script. All graphs are saved in subdirectories in png, eps, pdf and svg formats. Tabulated results are stored in files suitable for importing into LaTeX documents.

Value

This function is used for creation of the files saved in the working directory and a few of its subdirectories.

Author(s)

A. Jonathan R. Godfrey and Timothy P. Bilton

See Also

Other convenience functions can be investigated via their help pages. See `UniDesc`, `OneFactor`, and `TwoFactors`

Examples

```
data(airquality)
OnePredictor("Ozone", "Wind", airquality, View=FALSE)
# N.B. Various files and a folder were created in the working directory.
# Please investigate them to see how this function worked.
```

`Options` *Set package options*

Description

A set of convenience functions to alter the settings that control how much output is generated and displayed by other **BrailleR** functions.

Usage

```
GoBlind()
GoSighted()

LatexOn()
LatexOff()

ViewOn()
ViewOff()
```

Details

The function names should be fairly self explanatory. `GoBlind()` and `GoSighted()` control use of the `VI()` method which provides extra information about graphical objects for the assistance of blind users; `ViewOn()` and `ViewOff()` are for the automatic opening of `HTML` pages created by **BrailleR** functions; and `LatexOn()` and `LatexOff()` control the production of tables into LaTeX via the **xtable** package.

Value

Nothing is returned. These functions are only used for their side effects.

Author(s)

A. Jonathan R. Godfrey

See Also

See these settings applied as default arguments to `UniDesc`

---

pdf2html *Convert a pdf file to html*

Description

A blind user often has difficulty reading the content provided in pdf files. This tool is quite experimental at this stage.

Usage

```
pdf2html(pdffile, htmlfile=sub(".pdf", ".html", pdffile), HeadingLevels=4, PageTag="h6")
```

Arguments

`pdffile`	A pdf file to be converted.
`htmlfile`	The filename for the resulting html; default is to change the file extension.
`HeadingLevels`	The depth of heading level to include; tags h1, h2, h3... are used.
`PageTag`	Which tag to use for replacing page numbers; default is h6, but any tag could be used.

Details

A Python 2.7 module is the basis for the conversion. Some post-processing can be done to further enhance the readability of the resulting html file.

Value

Logical: Has the conversion completed. Note that this does not mean the result is totally useful as this will depend on the quality of the input file.

Author(s)

A. Jonathan R. Godfrey

`R2txtJG` *Save a transcript of commands and/or output to a text file.*

Description

These functions save a transcript of your commands and their output to a script file.

They work as combinations of `sink` and `history` with a couple of extra bells and whistles.

Usage

```
txtStart(file, commands=TRUE, results=TRUE, append=FALSE, cmdfile,
          visible.only=TRUE)

txtOut(Filename=NULL)

txtStop()

txtComment(txt,cmdtxt)

txtSkip(expr)
```

Arguments

`file`	Text file to save transcript in
`Filename`	A filename to be given for the `txtOut` command. If this is not specified, the user will be prompted for a filename. If the user presses the enter key, a filename will be automatically generated that is based on the current date and time.
`commands`	Logical, should the commands be echoed to the transcript file
`results`	Logical, should the results be saved in the transcript file
`append`	Logical, should we append to `file` or replace it
`cmdfile`	A filename to store commands such that it can be `sourced` or copied and pasted from
`visible.only`	Should non-printed output be included, not currently implemented.
`txt`	Text of a comment to be inserted into `file`
`cmdtxt`	Text of a comment to be inserted into `cmdfile`
`expr`	An expression to be executed without being included in `file` or `cmdfile`

Details

These functions are used to create transcript/command files of your R session. In the original **TeachingDemos** package from which the functions were obtained, there are 3 sets of functions. Those starting with "txt",those starting with "etxt", and those starting with wdtxt.

The "txt" functions create a plain text transcript while the "etxt" functions create a text file with extra escapes and commands so that it can be post processed with enscript (an external program) to create a postscript file and can include graphics as well. The postscript file can be converted to pdf or other format file. The "wdtxt" functions will insert the commands and results into a Microsoft Word document.

Users wishing to have the additional functionality that the "etxt" and "wdtxt" functions provide are advised to make use of the **TeachingDemos** package.

If `results` is TRUE and `commands` is FALSE then the result is similar to the results of `sink`. If `commands` is true as well then the results will show both the commands and results similar to the output on the screen. If both `commands` and `results` are FALSE then pretty much the only thing these functions will accomplish is to waste some computing time.

If `cmdfile` is specified then an additional file is created with the commands used (similar to the `history` command), this file can be used with `source` or copied and pasted to the terminal.

The Start function specifies the file/directory to create and starts the transcript, The prompts are changed to remind you that the commands/results are being copied to the transcript. The Stop function stops the recording and resets the prompts.

The txtOut function is a short cut for the txtStart command that uses the current date and time in the filenames for the transcript and command files. This function is not part of the **TeachingDemos** package.

The R parser strips comments and does some reformatting so the transcript file may not match exactly with the terminal output. Use the `txtComment` functions to add a comment. This will show up as a line offset by whitespace in the transcript file. If `cmdtxt` is specified then that line will be inserted into `cmdfile` preceded by a \# so it will be skipped if sourced or copied.

The `txtSkip` function will run the code in `expr` but will not include the commands or results in the transcript file (this can be used for side computations, or requests for help, etc.).

Value

Most of these commands do not return anything of use. The exception is:

`txtSkip` returns the value of `expr`.

Note

These commands do not do any fancy formatting of output, just what you see in the regular terminal window. If you want more formatted output then you should look into `Sweave` or the use of markdown documents..

Do not use these functions in combination with the **R2HTML** package or `sink`.

Author(s)

Greg Snow, <greg.snow@imail.org> is the original author, but Jonathan Godfrey <a.j.godfrey@massey.ac.nz> is responsible for the implementation in the **BrailleR** package (including the `txtOut()` function), and should therefore be your first point of contact with any problems. If you find the functions useful, you may wish to send a vote of thanks in Greg's direction.

See Also

`sink`, `history`, `Sweave`, the odfWeave package, the R2HTML package, the R2wd package

Examples

```
## Not run:
txtStart()
txtComment('This is todays transcript')
date()
x <- rnorm(25)
summary(x)
stem(x)
txtSkip(?hist)
hist(x)
Sys.Date()
Sys.time()

## End(Not run)
```

Require *Load a package by installing it if necessary*

Description

It is easier to run a script if we know the packages will be installed if this additional step is necessary. Installation uses the RStudio mirror of CRAN.

Usage

```
Require(pkg)
```

Arguments

pkg	the package to be loaded/installed.

Value

logical: to say that the package has been successfully loaded (invisible)

Author(s)

A. Jonathan R. Godfrey

See Also

require from the base package

`ScatterPlot` *Create a standard scatter plot with a few extra elements added to the output object*

Description

This function is a wrapper to the standard `plott()` function in the **graphics** package. It is tailored to generating a scatter plot, and adds detail to the stored object so that a better text description can be formulated using the `VI()` method in the **BrailleR** package.

Usage

```
ScatterPlot(x, y, ...)

FittedLinePlot(x, y, line.col=2, ...)
```

Arguments

`x,y`	numeric variables.
`line.col`	colour to be used for the fitted line; col is used to modify the colour of the points.
`...`	additional arguments passed on to the plotting function.

Details

These wrapper functions will draw the graphics plots for a set of points. The only difference is that the fitted line is added for the FittedLinePlot()

Value

An object of class scatterplot of fittedlineplot, with the addition of any calls to the main title or axis labels being explicitly stored even if a zero length character string.

Author(s)

A. Jonathan R. Godfrey

References

Godfrey, A.J.R. (2013) 'Statistical Software from a Blind Person's Perspective: R is the Best, but we can make it better', The R Journal 5(1), pp73-79.

Examples

```
attach(airquality)
op = par(mfcol=c(3,2))
plot(Wind, Ozone, pch=4)
test1 = ScatterPlot(Wind, Ozone, pch=4)
test1 #does the plot method work?
plot(Wind, Ozone)
abline(coef(lm(Ozone~Wind)), col=4)
test2 = FittedLinePlot(Wind, Ozone, line.col=4)
test2 #does the plot method work?
par(op)
detach(airquality)
rm(test1); rm(test2); rm(op)
```

SetOptions *Functions for setting package options.*

Description

Some package options have arguments which need validation. Setting the default significance level for analyses was the first function of this kind. Setting the name of the user to be inserted into documents was the second. Others are detailed further below.

Usage

```
ChooseEmbosser(Embosser = "none", Permanent = interactive(), Local = interactive())

ChooseStyle(css = "BrailleR.css", Permanent = interactive(), Local = interactive())
ChooseSlideStyle(css = "JGSlides.css", Permanent = interactive(), Local = interactive())

ResetDefaults(Local = interactive())

SetAuthor(name = "BrailleR", Permanent = interactive(), Local = interactive())

SetBRLPointSize(pt, Permanent = FALSE, Local = interactive())
SetPaperHeight(Inches, Permanent = FALSE, Local = interactive())
SetPaperWidth(Inches, Permanent = FALSE, Local = interactive())

SetLanguage(Language = "en_us", Permanent = interactive(), Local = interactive())

SetMakeUpper(Upper, Permanent = interactive(), Local = interactive())

SetPValDigits(digits, Permanent = interactive(), Local = interactive())

SetSigLevel(alpha, Permanent = interactive(), Local = interactive())
```

Arguments

alpha	The level of alpha to be used for analyses. Must be between zero and one or a warning is given and the option is not changed.
css	a cascading style sheet file to be inserted in HTML documents created by convenience functions. The file must be placed in the css folder within the MyBrailleR folder created by the user when prompted, for this to work.
digits	The number of decimal places to display. Must be an integer greater than one or a warning is given and the option is not changed.
Embosser	the name of the embosser to be used for tactile images. Not all embossers will be immediately supported by the package. The supported embossers are listed in the relevant section below. Please contact the package maintainer to introduce a new embosser to the list of supported models.
Inches	The size of the area to use for embossing. This should be the size of the embossed area not the actual size of the paper itself.
Language	The character string for the language files to be used in spell checking.
Local	Should the local copy of BrailleROptions be updated?
name	a character string to be used for author details in various file writing functions.
Permanent	Should the change be made permanent? Set to FALSE for a temporary change.
pt	The point size of the chosen braille font.
Upper	Should the initial letter of variable names be capitalised in captions and filenames? Logical, initially set to TRUE.

Details

More convenience functions for BrailleR users. Most are self explanatory, but the following details should be noted.

The Choose...() functions are used for establishing default parameters for other details. The ChooseStyle() command can be used to alter the appearance of HTML output by way of cascading style sheets. You can create your own css file and add it to your user folder called MyBrailleR before calling this function.

The ChooseEmbosser() will look for the default settings recommended for particular types of embosser. Initial testing was done on a Tiger Premier 100 embosser manufactured by ViewPlus Inc. The default paper size is 11 by 11.5 inches, but the recommended embossing area for graphics is 10 by 10 inches. Please submit your preferences for any embosser to the package maintainer.

The Set..() commands will let the user specify any desired value for the options as long as it is valid: Options assumed to be character strings are checked to be so, integers must be integers and a proportion must be between zero and one.

SetPaperHeight and SetPaperWidth are temporary changes by default because some types of images are not meant to use the maximum area set down by the original default settings for an embosser. Careful experimentation may be required to get optimal results. If permanent changes are desired, then please contact the package maintainer to explain why you have made these changes so that we can help other users get the best from a wide range of embossers.

SetPValDigits() is used for rounding purposes to avoid the use of scientific notation. It is not used for determining significance.

Author(s)

A. Jonathan R. Godfrey

Examples

```
# SetSigLevel(5) # not a valid alpha
SetSigLevel(0.05, Local=FALSE) # valid alpha value
SetAuthor(Local=FALSE)
SetAuthor("Jonathan Godfrey", Local=FALSE)
```

SetupBrailleR *Establish the BrailleR folder for the user*

Description

Creates a permanent folder if the user agrees.

Usage

```
SetupBrailleR()
```

Details

The user can establish a permanent folder if the BrailleR package is loaded in an interactive session. If only used in batch mode, the user will only be able to use the default BrailleR settings.

Value

The path to the folder is returned invisibly.

Author(s)

A. Jonathan R. Godfrey with suggestions from Henrik Bengtsson and Brian Ripley.

---

sort.VIgraph *Sort VI.ggplot points list*

Description

Allows the list of data points listed by VI.ggplot to be sorted by x or y values, ascending or descending. Currently only implemented for geom_points. This function is experimental and has not been extensively tested.

Usage

```
## S3 method for class 'VIgraph'
sort(x, decreasing = FALSE, by = "x", ...)
```

Arguments

x	object returned by VI.ggplot
decreasing	logical: should the sort be decreasing
by	value on which to sort, "x" or "y"
...	further arguments passed to base::sort

Value

Returns a new object of the same type as that returned by VI.ggplot, but with data re-ordered.

Author(s)

Debra Warren and Paul Murrell

Examples

```
if (require(ggplot2)) {
    sort(VI(qplot(x=1:5, y=c(2,5,1,4,3))), decreasing=TRUE, by="y")
}
```

SpellCheck *A spell checking interface*

Description

An terminal-based interface for dealing with words that appear to be misspelled in the specified files. Files for lists of words to ignore at local and global levels are augmented, or replacement text can be supplied.

Usage

```
SpellCheck(file)
```

Arguments

file	A vector of files to be checked

Details

A backup file is created before any changes are made. This file is not overwritten each time the spell checking is done so it may end up becoming out of date. Users can delete the backup file anytime.

Value

NULL. This function is intended to affect only external files.

Author(s)

A. Jonathan R. Godfrey

`SpellCheckFiles` *Spell checking a file or all files within a specified folder*

Description

Check spelling using hunspell. A new print method is also used.

Usage

```
SpellCheckFiles(file = ".", ignore = character(),
    local.ignore = TRUE, global.ignore = TRUE)

## S3 method for class 'wordlist'
print(x, ...)
```

Arguments

`file`	The filename of an individual file, or an individual folder.
`ignore`	The character vector of words to be ignored by hunspell
`local.ignore`	Use a local file of words to be ignored. This file has the same name as the file with .ignore.txt tacked on the end and is colocated with the file being checked. If the file argument is set to a folder then the local.ignore can be set to the name of a file in the current working directory.
`global.ignore`	Use the global word list called words.ignore.txt found in the MyBrailleR folder
`x`	the object to be printed
`...`	other parameters pass to the print method

Details

The global list of words to be ignored needs to be saved in the user's MyBrailleR folder. It can be updated as often as the user likes. It should have one word per line, and contain no space, tab or punctuation characters.

Value

A list object with each item of the list being the findings from spell checking each file. The words not found in the dictionary are given as well as the line numbers where they were found.

Author(s)

A. Jonathan R. Godfrey wrote these functions but leaned heavily on functions found in the devtools package.

See Also

The hunspell package and functions therein.

SVGThis *Save commonly used graphs as structured SVG files.*

Description

Converts a graph object (as long as it has a class assigned) or the current graph window to an SVG file that can be viewed using a browser (not IE) or the Tiger Player software available from ViewPlus Inc. At present, the SVG needs manual editing using the Tiger Transformer software before viewing in the Tiger Player.

Usage

```
SVGThis(x, file = "test.svg", ...)
```

Arguments

x	a graph object for which a method exists, or the current graphics device if set to NULL.
file	The SVG file to be created.
...	Arguments to be passed to the methods.

Details

The Cairo SVG device found in the **gr.devices** package does not create a structured SVG file that includes the semantics of the graphic being displayed. The SVG created by the **gridSVG** package does meet this need, but only works on graphs drawn using the **grid** package. Any graph created using functions from the more common **graphics** package can be converted to the **grid** package system using the **gridGraphics** package.

The `SVGThis.ggplot` method accepts argument `createDevice` (default TRUE) - if true the method creates its own null pdf device to draw the graph on, if false it draws on the current device.

Value

NULL. This function is solely for the purpose of creating SVG files in the current working directory or in a path of the user's choosing.

Author(s)

A. Jonathan R. Godfrey and Paul Murrell

References

P. Murrell and S. Potter (2014) "The **gridSVG** package" The R Journal 6/1, pp. 133-143. http://journal.r-project.org/archive/2014-1/RJournal_2014-1_murrell-potter.pdf

P. Murrell (2015) "The **gridGraphics** package", The R Journal 7/1 pp. 151-162. http://journal.r-project.org/archive/2015-1/murrell.pdf

P. Dengler et al. (2011) Scalable vector graphics (SVG) 1.1, second edition. W3C recommendation, W3C. http://www.w3.org/TR/2011/REC-SVG11-20110816/

Examples

```
x=rnorm(1000)
#SVGThis(hist(x))
```

`ThreeFactors` *A convenience function for a Three-way analysis*

Description

Prepares an analysis of a data set with one response and three predictors that are all factors. Interactions between the Three factors are also allowed for. The function creates a number of graphs and tables relevant for the analysis.

Usage

```
ThreeFactors(Response, Factor1, Factor2, Factor3, Data = NULL, Filename = NULL,
   Folder = NULL, VI = getOption("BrailleR.VI"), Latex = getOption("BrailleR.Latex"),
    View = getOption("BrailleR.View"))
```

Arguments

`Response`	Name of the continuous response variable.
`Factor1, Factor2, Factor3`	Name the three factors to be included.
`Data`	Name the data.frame that includes the three variables of interest.
`Filename`	Name of the Rmarkdown and HTML files to be created. A default will be created that uses the names of the variables if this is left set to NULL.
`Folder`	Name of the folder to store graph and LaTeX files. A default will be created based on the name of the data.frame being used.
`VI`	Logical: Should the VI method for blind users be employed?
`Latex`	Logical: Should the tabulated sections be saved in LaTeX format?
`View`	Logical: Should the HTML file be opened for inspection?

Details

to complete

Value

to complete

Author(s)

A. Jonathan R. Godfrey and Timothy P. Bilton

See Also

The `OneFactor` script was the basis for this function;.

Examples

```
TestData=data.frame(Resp=sample(54), expand.grid(F1=c("a","b","c"),
    F2=c("d","e","f"), F3=c("g","h","i"), rep=c(1,2)))
attach(TestData)
ThreeFactors(Resp,F1,F2,F3)
detach(TestData)
rm(TestData)
```

TSPlot — *Create a standard time series plot with a few extra elements added to the output object*

Description

This function is a wrapper to the standard `plott()` function in the **graphics** package. It is tailored to generating a time series plot, and adds detail to the stored object so that a better text description can be formulated using the `VI()` method in the **BrailleR** package.

Usage

```
TimeSeriesPlot(x, ...)
```

Arguments

`x`	a numeric variable.
`...`	additional arguments passed on to the plotting function.

Details

This wrapper will draw the base graphics plot for a time series. The saved object can be plotted later with a call to `plot`.

Value

An object of class tsplot, with the addition of any calls to the main title or axis labels being explicitly stored even if a zero length character string.

Author(s)

A. Jonathan R. Godfrey

References

Godfrey, A.J.R. (2013) 'Statistical Software from a Blind Person's Perspective: R is the Best, but we can make it better', The R Journal 5(1), pp73-79.

Examples

```
attach(airquality)
op = par(mfcol=c(3,2))
plot(as.ts(Wind), ylab="Wind", col=4)
test1 = TimeSeriesPlot(Wind, col=4)
test1 #does the plot method work?
plot(as.ts(Ozone), ylab="Ozone")
test2 = TimeSeriesPlot(Ozone)
test2 # does the plot method work?
par(op)
detach(airquality)
rm(test1); rm(test2); rm(op)
```

TwoFactors *A convenience function for a two-way analysis*

Description

Prepares an analysis of a data set with one response and two predictors that are both factors. An interaction between the two factors is also allowed for. The function creates a number of graphs and tables relevant for the analysis.

Usage

```
TwoFactors(Response, Factor1, Factor2, Inter = FALSE, HSD = TRUE,
   AlphaE = getOption("BrailleR.SigLevel"), Data = NULL, Filename = NULL,
   Folder = NULL, VI = getOption("BrailleR.VI"), Latex = getOption("BrailleR.Latex"),
    View = getOption("BrailleR.View"))
```

Arguments

Response	Name of the continuous response variable.
Factor1, Factor2	Name the two factors to be included.
Inter	Logical: Should the interaction of the two factors be included?
HSD	Logical: Should Tukey's HSD be evaluated for the data?
AlphaE	The family-wise Type I error rate for Tukey's HSD calculations.
Data	Name the data.frame that includes the three variables of interest.
Filename	Name of the Rmarkdown and HTML files to be created. A default will be created that uses the names of the variables if this is left set to NULL.

Folder	Name of the folder to store graph and LaTeX files. A default will be created based on the name of the data.frame being used.
VI	Logical: Should the VI method for blind users be employed?
Latex	Logical: Should the tabulated sections be saved in LaTeX format?
View	Logical: Should the HTML file be opened for inspection?

Details

to complete

Value

to complete

Author(s)

Timothy P. Bilton and A. Jonathan R. Godfrey

See Also

The OneFactor script was the basis for this function;.

Examples

```
TG <- ToothGrowth
TG$dose <- as.factor(TG$dose)

# Without interaction
TwoFactors('len','supp','dose',Data=TG, Inter=FALSE)

# With two-way interaction
TwoFactors('len', 'supp', 'dose', Data=TG, Inter=TRUE)

# For unbalanced data
TG_Unb <- TG[-c(53:60),]
TwoFactors('len', 'supp', 'dose', Data=TG_Unb, Inter=TRUE)
rm(TG); rm(TG_Unb)
```

---

unfinished *Unfinished Methods to help vision impaired useRs*

Description

A set of methods that will (once coded) extract the most relevant information from a graphical object (or implied set of graphical objects) and display the interpreted results in text form.

The method includes representations of summary methods that are more suitable for blind useRs. For example, the method for a data.frame uses a single line for each variable instead of the normal column layout used by the summary method.

Details

This is the help page for the VI() functions that are not fully functional or below par in some way.

Value

This will vary according to the needs of vision impaired useRs and the specific objects that need to be interpreted.

In general, the output is a series of text strings printed in the console/terminal window in addition to the embedded command's normal functionality.

These functions do not create objects as do many R commands. Manipulations on the objects created by regular R expressions will need those regular expressions issued in addition to those of the VI family of functions.

Author(s)

Jonathan Godfrey

`UniDesc` *Descriptive statistics and graphs for univariate data*

Description

This function is a convenience function for analyzing univariate data. It provides histograms, boxplots and tabulated results for normality tests as well as those for skewness and kurtosis. The intended use of this function is principally for a blind user of R who also has the advantage of retrieving textual descriptions of the graphs created along the way, via the `VI()` methods.

Usage

```
UniDesc(Response = NULL, ResponseName = as.character(match.call()$Response),
    Basic = TRUE, Graphs = TRUE, Normality = TRUE, Tests = TRUE,
    Title = NULL, Filename = NULL, Folder = ResponseName, Process = TRUE,
VI = getOption("BrailleR.VI"), Latex = getOption("BrailleR.Latex"),
View = getOption("BrailleR.View"), PValDigits=getOption("BrailleR.PValDigits"))
```

Arguments

`Response`	The numeric vector to be analyzed. This must be specified as a variable that is directly available in the workspace, not as a data.frame$variable construct.
`ResponseName`	This is the same as Response but use quote marks around it. Exactly one of Response or ResponseName must be specified.
`Basic`	logical, asking for basic numeric summary measures
`Graphs`	logical, indicating if the graphs are to be created. These will be eps files suitable for insertion in LaTeX documents, pdf files for more general use, and SVG for easier use by blind users.

Normality	logical, asking if the various normality tests offered in the **nortest** package should be used
Tests	logical, should skewness and kurtosis tests be performed.
Title	the title of the R markdown document being created. NULL leads to a default string being chosen.
Filename	Specify the name of the R markdown and html files (without extensions).
Folder	the folder where results and graph files will be saved.
Process	logical, should the R markdown file be processed?
VI	logical, should the VI method be used to give added text descriptions of graphs? This is most easily set via the package options.
Latex	logical, Should the **xtable** package be used to convert the tabulated results into LaTeX tables? This is most easily set via the package options.
View	logical, should the resulting HTML file be opened in a browser? This is most easily set via the package options.
PValDigits	Integer. The number of decimal places to use for p values. This is most easily set via the package options.

Value

Saves an R markdown file, (and then if Process=TRUE) an R script file, and an html file (which may be opened automatically) in the current working folder. Graphs are saved in png, eps, pdf, and SVG formats (if requested) in (optionally) a subfolder of the current working directory.

Author(s)

A. Jonathan R. Godfrey <a.j.godfrey@massey.ac.nz>

Examples

```
Ozone=airquality$Ozone
UniDesc(Ozone, View=FALSE)
rm(Ozone)
# N.B. Various files and a folder were created in the working directory.
# Please investigate them to see how this function worked.
```

UpdateGraph *extract or alter graph parameters*

Description

Either grabs the specified label or sets it to a newly specified value. In this case the graph is re-drawn and the graph object is updated.

Usage

```
UpdateGraph(object, ...)

main(graph, label = NULL)

xlab(graph, label = NULL)

ylab(graph, label = NULL)
```

Arguments

graph,object	The graph object to be updated.
label	the text to be used in place of the current text label. Use of the default NULL leads to the extraction of the current value and no updating is done.
...	the set of parameters to be altered.

Details

Specify the label to be an empty text string if the desire is to delete the current label.

Value

The graph object will be updated in the global environment if a new value is assigned.

Author(s)

A. Jonathan R. Godfrey

Examples

```
attach(airquality)
op = par(mfcol=c(3,2))
test1 = TimeSeriesPlot(Wind, col=4)
xlab(test1, "Day")
# check the change is permanent by doing another change
test1

test2 = TimeSeriesPlot(Ozone)
# using the update method
update(test2, main="important title", sub="subtitles aren't always present", ylab="Ozone (ppb)")
# finally, change the graph to use different plotting characters/line types
update(test2) # to fix
par(op)
detach(airquality)
rm(test1); rm(test2); rm(op)
```

VI *Methods to help vision impaired users*

Description

A set of methods that extract the most relevant information from a graphical object (or implied set of graphical objects) and display the interpreted results in text or HTML form.

The method includes representations of summary methods that are more suitable for blind users. For example, the method for a data.frame uses a single line for each variable instead of the normal column layout used by the `summary` method.

Usage

```
Describe(x, VI=FALSE, ...)
VI(x, Describe=FALSE, ...)

## S3 method for class 'histogram'
VI(x, Describe=FALSE, ...)
## S3 method for class 'histogram'
Describe(x, VI=FALSE, ...)

## S3 method for class 'aov'
VI(x, Describe=FALSE, ...)

## S3 method for class 'lm'
VI(x, Describe=FALSE, ...)
```

Arguments

`x`	any R object
`Describe,VI`	Should the other function be called at the same time
`...`	other arguments, currently ignored

Details

This is the general help page for the VI() functionality. Specific help pages will be created if the ability to alter the outcome through user input warrants. See below for more detail on these.

Describe() is for explaining how a given type of graph appears to the sighted world and is intended for use by blind people who do not know how that graph looks. There is room to add hints for displaying the graphs in a more visually appealing manner. In contrast, VI() is intended to extract the specific details for the particular graph or output concerned.

Further methods can be written by users (blind or sighted). Please submit to the package maintainer for possible inclusion in subsequent releases of the package.

Value

This will vary according to the needs of vision impaired users and the specific objects that need to be interpreted.

In general, the output from VI() is a series of text strings printed in the console/terminal window in addition to the embedded command's normal functionality. The `VI.lm()` method is the first to move away from this idea and use a process that builds on the `UniDesc()` function. In this case, the method creates an R markdown file and compiles it into HTML. The HTML document is opened if the R session is interactive.

The VI() functions do not create objects as do many R commands. Manipulations on the objects created by regular R expressions will need those regular expressions issued in addition to those of the VI family of functions. The `VI.lm()` method does create objects in the current workspace and then deletes them once the HTML document is compiled.

The Describe() family of functions do create an object for passing onto other functions.

Note

The `VI.lm` method fails if you use the one line `VI(lm(...))` even if the model is named using `VI(Model1 <- lm(...))`. It does work if two explicit commands are used. For example `Model1 = lm(...)` followed by `VI(Model1)`.

Author(s)

A. Jonathan R. Godfrey and Timothy P. Bilton

Examples

```
RandomX=rnorm(500)
PlottedFig=hist(RandomX)
rm(RandomX)
VI(PlottedFig)
Describe(PlottedFig)
rm(PlottedFig)
```

VI.ggplot *VI for graphs created using ggplot2*

Description

Prints a textual description of a graph produced by ggplot or qplot.

Usage

```
## S3 method for class 'ggplot'
VI(x, Describe=FALSE, threshold=10,
   template=system.file("whisker/VIdefault.txt", package="BrailleR"),
   ...)
```

Arguments

x	an object created by ggplot() or qplot() from the ggplot2 package which therefore has class gg or ggplot.
Describe	Should the Describe function be called at the same time. Not currently implemented for ggplot objects.
threshold	Maximum number of data items that should be individually listed in the output.
template	Template file, in mustache format, to be used in creating the text
...	other arguments, currently ignored

Value

Returns a structure containing a hierarchical representation of the graph as well as the text description as a character vector. When run interactively, the text description is printed.

Note

This function is experimental. The ggplot2 package produces many different types of graphs and offers many options for modifying the graph appearance. Not all options have been fully catered for, so text descriptions of these graphs may still be less complete than those created using base graphics, or may be misleading in some circumstances.

Author(s)

Debra Warren, Tony Hirst and A. Jonathan R. Godfrey

Examples

```
if(require(ggplot2)){
g = ggplot(economics_long, aes(date, value01, colour = variable))
g = g + geom_line() + ggtitle('dummy title')
VI(g)
g
}
```

ViewSVG *Create the necessary files to allow interactive viewing of SVG*

Description

Copy files from the package folders to the current working directory and write a markdown file that contains the links to all accessible SVG files in the current directory.

Usage

```
ViewSVG(dir=".")
```

Arguments

dir	the directory to search for content; currently limited to the current working directory

Details

Two files are copied from the package folders and an index file is written that contains the necessary links to all accessible SVG files.

Value

NULL. This function exists for its side effects only.

Author(s)

A. Jonathan R. Godfrey

References

Godfrey, A.J.R. (2013) 'Statistical Software from a Blind Person's Perspective: R is the Best, but we can make it better', The R Journal 5(1), pp73-79.

WhereXY *Count points in a scatter plot*

Description

count the number of points that fall into various sized subparts of a scatter plot. The graphing region can be split into cells based on a uniform or normal marginal distribution separately for the x and y variables.

Usage

```
WhereXY(x, y = NULL, grid = c(4, 4), xDist = "uniform",
       yDist = xDist, addmargins=TRUE)
```

Arguments

x,y	vectors of x coordinates. If y is not specified, the function expects x to be a two-column matrix with x and y values in columns 1 and 2 respectively.
grid	pair of values to specify the way the graph is to be split into parts. Specify x and then y.
xDist,yDist	the distribution the variables might be expected to follow. The default is to consider uniformly distributed but any alternative text will lead to an assumption of both margins being normally distributed.
addmargins	logical: should sums be added to both rows and columns.

Value

A text description of the number of points in each subregion of the scatter plot. The table of counts can then be compared to the expected number of points in each subregion.

Author(s)

A. Jonathan R. Godfrey

Examples

```
x=rnorm(50)
y=rnorm(50)
WhereXY(x,y)
WhereXY(x,y, c(3,4))
WhereXY(x,y, xDist="other")
```

`WriteR` *Getting started with WriteR*

Description

The WriteR application was written in wxPython so that blind users can process Rmarkdown documents. This functionality is offered because RStudio is not an accessible application for screen reader users.

Usage

```
WriteR(file = NULL, math = c("webTeX", "MathJax"))

PrepareWriteR(Author = getOption("BrailleR.Author"))
```

Arguments

`Author`	Your name as you want it to appear in the default text that starts your R markdown documents.
`file`	The filename you want started. Not implemented yet.
`math`	The style for mathematical content in HTML documents created using LaTeX input. Not yet implemented.

Details

WriteR needs python 2.7 and wxPython 2.8 to run. Error handling for users without these packages installed is not yet incorporated in the functions.

The `PrepareWriteR()` function writes the settings file (called WriteROptions) for WriteR and copies the files that were part of the BrailleR installation into the current working directory. You will be able to run the WriteR application from there, or move to a folder of your choosing.

Value

NULL. This function is for your convenience and not for doing any work inside an R session.

Note

You must have Python 2.7 and the associated wxPython installation on your system to use the WriteR application.

Author(s)

A. Jonathan R. Godfrey

`WTF` *What's this figure?*

Description

Determine what the current graphics device has on it so the blind user can be sure they have something they want, or find out what it might be that is contained in a graphics device.

Usage

```
WTF()
```

Author(s)

A. Jonathan R. Godfrey and Paul Murrell.

Examples

```
attach(airquality)
hist(Ozone)
WTF()
plot(Ozone~Wind)
WTF()
detach(airquality)
```

Index