Package: installr (via r-universe)

August 20, 2024

Type Package

Title Using R to Install Stuff on Windows OS (Such As: R, 'Rtools', 'RStudio', 'Git', and More!)

Version 0.23.4

Date 2022-11-13

Description R is great for installing software. Through the 'installr' package you can automate the updating of R (on Windows, using updateR()) and install new software. Software installation is initiated through a GUI (just run installr()), or through functions such as: install.Rtools(), install.pandoc(), install.git(), and many more. The updateR() command performs the following: finding the latest R version, downloading it, running the installer, deleting the installation file, copy and updating old packages to the new R installation.

URL `https://talgalili.github.io/installr/`, `https://github.com/talgalili/installr/`, `https://www.r-statistics.com/tag/installr/`

BugReports `https://github.com/talgalili/installr/issues`

Depends R (>= 2.14.0)

Imports stringr, utils

Suggests curl, XML, devtools, rjson, data.table, plyr, ggplot2, sp, tools, pkgbuild (>= 1.1.0), testthat (>= 2.1.0), mockery, R.utils

License GPL-2

RoxygenNote 7.1.1

Encoding UTF-8

Repository https://talgalili.r-universe.dev

RemoteUrl https://github.com/talgalili/installr

RemoteRef HEAD

RemoteSha 0ef36a90a2961551cbf53be267077e03eb413cd0

Contents

`installr-package` *Using R to Install Stuff (Such As: R, Rtools, RStudio, Git, and More!)*

Description

R is great for installing software. Through the 'installr' package you can automate the updating of R (on Windows, using updateR()) and install new software. Software installation is initiated through a GUI (just run installr()), or through functions such as: install.Rtools(), install.pandoc(), install.git(), and many more. The updateR() command performs the following: finding the latest R version, downloading it, running the installer, deleting the installation file, copy and updating old packages to the new R installation.

`add.installr.GUI` *Adds a menu based GUI for updating R within Rgui*

Description

Adds a menu based GUI for updating R within Rgui.

Usage

```
add.installr.GUI()
```

Details

This function is used during .onLoad to load the menus for the installr package in Rgui.

Value

Returns invisible TRUE/FALSE if menus were added or not.

Author(s)

Tal Galili, Dason

References

My thanks goes to Yihui and Dason, for the idea and help with implementation. See also: `https://stackoverflow.com/questions/15250487/how-to-add-a-menu-item-to-rgui/`

Examples

```
## Not run:
add.installr.GUI()

## End(Not run)
```

`add_load_installr_on_startup_menu` *Add menu item for having installr load on startup*

Description

Add menu item for having installr load on startup

Usage

```
add_load_installr_on_startup_menu(...)
```

Arguments

`...`	not used. (but good for future backward compatibility)

`add_remove_installr_from_startup_menu` *Add menu item for having installr NOT load on startup*

Description

Add menu item for having installr NOT load on startup

Usage

```
add_remove_installr_from_startup_menu(...)
```

Arguments

`...`	not used. (but good for future backward compatibility)

add_to_.First_in_Rprofile.site

Add a code line to Rprofile.site .First

Description

Goes through Rprofile.site text, finds the .First function - and add a line of code to the beginning of it.

Usage

```
add_to_.First_in_Rprofile.site(code, indent = "\t", ...)
```

Arguments

`code`	A character scalar with code to add at the beginning of the .First function in Rprofile.site
`indent`	a character scalar indicating the text to be added before code. Default is a tab.
`...`	not used.

References

https://stackoverflow.com/questions/1395301/how-to-get-r-to-recognize-your-working-directory-as-it
https://stackoverflow.com/questions/1189759/expert-r-users-whats-in-your-rprofile
http://www.noamross.net/archives/2012-11-02-rprofile/ https://www.statmethods.net/interface/customizing.html

Examples

```
## Not run:
is_in_.First_in_Rprofile.site("suppressMessages(library(installr))") # FALSE
add_to_.First_in_Rprofile.site("suppressMessages(library(installr))")
is_in_.First_in_Rprofile.site("suppressMessages(library(installr))") # TRUE
remove_from_.First_in_Rprofile.site("suppressMessages(library(installr))")
is_in_.First_in_Rprofile.site("suppressMessages(library(installr))") # FALSE
# this would still leave .First

## End(Not run)
```

`ask.user.for.a.row` *Asks the user for a row number from a data.frame table*

Description

The function gets a data.frame and asks the user to choose a row number. Once choosen, that row number is returned from the function.

Usage

```
ask.user.for.a.row(
  TABLE,
  header_text = "Possible versions to download (choose one)",
  questions_text
)
```

Arguments

`TABLE`	a data.frame table with rows from which we wish the user to choose a row. If TABLE is not a data.frame, it will be coerced into one.
`header_text`	the text the users sees (often a question) as a title for the printed table - explaining which row they should choose from
`questions_text`	the question the users see after the printing of the table - explaining which row they should choose from. (the default is: "Please review the table of versions from above, and enter the row number of the file-version you'd like to install: ")

Details

This function is used in `installr` when we are not sure what version of the software to download, or when various actions are available for the user to choose from. If the user doesn't give a valid row number, the function repeats its questions until a valid row number is chosen (or the user escapes)

Value

The row number the user has choosen from the data.frame table.

Source

On how to ask the user for input:
`https://stackoverflow.com/questions/5974967/what-is-the-correct-way-to-ask-for-user-input-in-an-r-`

Examples

```
## Not run:
version_table <- data.frame(versions = c("devel", "V 1.0.0", "V 2.0.0"))
installr:::ask.user.for.a.row(version_table)

## End(Not run)
```

`ask.user.yn.question` *Asks the user for one yes/no question.*

Description

Asks the user for one yes/no question. If the users replies with a "yes" (or Y, or y) the function returns TRUE. Otherwise, FALSE. (also exists as the function devtools::yesno)

Usage

```
ask.user.yn.question(question, GUI = TRUE, add_lines_before = TRUE)
```

Arguments

`question`	a character string with a question to the user.
`GUI`	a logical indicating whether a graphics menu should be used if available. If TRUE, and on Windows, it will use winDialog, otherwise it will use menu.
`add_lines_before`	if to add a line before asking the question. Default is TRUE.

Value

TRUE/FALSE - if the user answers yes or no.

References

https://stackoverflow.com/questions/15250487/how-to-add-a-menu-item-to-rgui (my thanks goes to Dason for his answer and help)

See Also

menu in the utils package, yesno in the devtools package.

Examples

```
## Not run:
ask.user.yn.question("Do you love R?")
ask.user.yn.question(question = "Do you love R?", GUI = TRUE) # the same one as before
ask.user.yn.question(question = "Do you love R?", GUI = FALSE)
# reverts to command line questions

ask.user.yn.question("Lorem ipsum dolor sit amet, consectetur adipisicing elit,
sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.
Ut enim ad minim veniam, quis nostrud exercitation

  ullamco laboris nisi ut aliquip
  ex ea commodo consequat. Do \n you \n love R?")
   # checking how it deals with multi lines, and a lot of text (very good actually)
```

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

`barplot_package_users_per_day`
barplot for the number of users installation of a package

Description

This function is a first template for creating a barplot of the number of downloads a package had in a time period. This function is based on some other functions, have a look at the example for more details.

Usage

```
barplot_package_users_per_day(pkg_name, dataset, remove_dups = TRUE, ...)
```

Arguments

`pkg_name`	a string of the package we are interested in checking.
`dataset`	a dataset output from running read_RStudio_CRAN_data.
`remove_dups`	default is TRUE. Should the duplicate user ids (based on their ips) be removed. If TRUE, then the plot is the number of unique users who have downloaded our package everyday.
`...`	not in use.

Details

RStudio maintains its own CRAN mirror, https://cran.rstudio.com/ and offers its log files.

Value

Returns the total number of downloads of the package for that time period.

See Also

download_RStudio_CRAN_data, read_RStudio_CRAN_data,barplot_package_users_per_day

Examples

```
## Not run:
# The first two functions might take a good deal of time to run (depending on the date range)
RStudio_CRAN_data_folder <-
      download_RStudio_CRAN_data(START = '2013-04-02',
                                 END = '2013-04-05')
                                 # around the time R 3.0.0 was released
my_RStudio_CRAN_data <- read_RStudio_CRAN_data(RStudio_CRAN_data_folder)
my_RStudio_CRAN_data <- format_RStudio_CRAN_data(my_RStudio_CRAN_data)
head(my_RStudio_CRAN_data)
lineplot_package_downloads(pkg_names = c("ggplot2", "reshape", "plyr", "installr"),
                           dataset = my_RStudio_CRAN_data)

# older plots:
# barplots: (more functions can easily be added in the future)
barplot_package_users_per_day("installr", my_RStudio_CRAN_data)
barplot_package_users_per_day("plyr", my_RStudio_CRAN_data)

## End(Not run)
```

browse.latest.R.NEWS *See the NEWS file for the latest R release*

Description

Sends the user the the NEWS html file on "https://cran.rstudio.com/bin/windows/base/NEWS.R-3.0.0.html" (URL changes with each version)

Usage

```
browse.latest.R.NEWS(URL = "https://cran.rstudio.com/bin/windows/base/", ...)
```

Arguments

URL	the URL of the page from which R can be downloaded.
...	for future use

Value

invisible(NULL)

Examples

```
## Not run:
browse.latest.R.NEWS()

## End(Not run)
```

check.for.updates.R *Checks if there is a newer version of R*

Description

Fetches the latest (not development!) R version and compares it with your currently installed R version (the version of the R session from which you are running this function).

Usage

```
check.for.updates.R(
  notify_user = TRUE,
  GUI = TRUE,
  page_with_download_url = "https://cran.rstudio.com/bin/windows/base/",
  pat = "R-[0-9.]+.+-win\\.exe"
)
```

Arguments

`notify_user`	if to print to you (the user) what is the latest version and what version you are currently using.
`GUI`	a logical indicating whether a graphics menu should be used if available. If TRUE, and on Windows, it will use winDialog, otherwise it will use cat.
`page_with_download_url`	the URL of the page from which R can be downloaded.
`pat`	pattern to search for when looking for a newer R version

Value

TRUE/FALSE - if there is a newer version of R to install or not.

Examples

```
## Not run:
check.for.updates.R()
# Possible output:
#  There is a newer version of R for you to download!
#  You are using R version:  2.15.0
#  And the latest R version is:  2.15.3
#  [1] TRUE

## End(Not run)
```

`check.integer` *Check if a number is integer*

Description

Returns TRUE/FALSE on whether a number is integer or not.

Usage

```
check.integer(N)
```

Arguments

`N`	A number (if a vector is supplied only the first element is checked - without warning)

Details

Surprising as it may be, R doesn't come with a handy function to check if the number is integer. This function does just this.

Value

TRUE/FALSE on whether a number is integer or not.

Author(s)

VitoshKa

Source

https://stackoverflow.com/questions/3476782/how-to-check-if-the-number-is-integer

Examples

```
check.integer <- installr:::check.integer
check.integer(4) # TRUE
check.integer(3243) #TRUE
check.integer(3243.34) #FALSE
check.integer("sdfds") #FALSE
check.integer(1e4) #TRUE
check.integer(1e6) #TRUE
check.integer(1e600) #FALSE - the function is having a hardtime with Inf...
rm(check.integer)
```

`checkMD5sums2` *Check and Create MD5 Checksum Files*

Description

checkMD5sums checks the files against a file 'MD5'. This extends the default checkMD5sums from package tools by adding a new parameter "md5file"

Usage

```
checkMD5sums2(package, dir, md5file, omit_files, ...)
```

Arguments

`package`	the name of an installed package
`dir`	the path to the top-level directory of an installed package.
`md5file`	the exact path of the md5file to compare the dir with
`omit_files`	a character vector with the files or file directories to not include in the checksums
`...`	not used. (but good for future backward compatibility)

Value

checkMD5sums returns a logical, NA if there is no 'MD5' file to be checked.

See Also

checkMD5sums

Examples

```
## Not run:
checkMD5sums2(dir=R.home()) # doesn't work for R 3.0.0 or R 3.0.1
checkMD5sums2(dir=R.home(), omit_files = c("etc/Rconsole", "etc/Rprofile.site")) # will work!
# tools::md5sum(file.path(R.home(), "MD5"))


## End(Not run)
```

copy.packages.between.libraries

Copies all packages from one library folder to another

Description

Copies all packages from one folder to another. This function is used if we wish to either:

- Upgrade R to a new version - and copy all of the packages from the old R installation to the new one.
- Move to a global library system - and wanting to copy all of packages from the local library folder to the global one

It takes into account that we don't want to copy packages which have "high" importance (such as MASS, boot, graphics, utils, rpart, Matrix and more GREAT packages...) to the new library folder. Also, it assumes that within an R installation, the packages are located inside the "library" folder.

Usage

```
copy.packages.between.libraries(
  from,
  to,
  ask = FALSE,
  keep_old = TRUE,
  do_NOT_override_packages_in_new_R = TRUE
)
```

Arguments

`from`	a character vector for the location of the old library folder FROM which to copy files from.
`to`	a character vector for the location of the old library folder TO which to copy files to.
`ask`	should the user be given the option to choose between which two libraries to copy the packages? If FALSE (default), the folders are copied from the before-newest R installation to the newest R installation. This the overrides "from" and "to" parameters.
`keep_old`	should the packages be COPIED to the new library folder, thus KEEPing the old package as they are? Or should they be removed?
`do_NOT_override_packages_in_new_R`	default TRUE If FALSE, then If a package exists in both the "from" and "to" library folders - it would copy to "to" the version of the package from "from". (this parameter should rarely be FALSE)

Value

TRUE if it copied (moved) packages, and FALSE if it did not.

See Also

get.installed.R.folders

Examples

```
## Not run:
copy.packages.between.libraries(ask = T)
# it will ask you from what R version
# to copy the packages into which R version.
# Since (do_NOT_override_packages_in_new_R = T) the function will
# make sure to NOT override your newer packages.

# copy.packages.between.libraries(ask = T, keep_old = F)
# As before, but this time it will MOVE (instead of COPY) the packages.
#  e.g: erase them from their old location.

## End(Not run)
```

`cranometer` *Measures the speed of downloading from different CRAN mirrors*

Description

Estimates the speed of each CRAN mirror by measuring the time it takes to download the NEWS file.

Usage

```
cranometer(ms = getCRANmirrors(all = FALSE, local.only = FALSE), ...)
```

Arguments

`ms`	- the output of getCRANmirrors. Defaults to using all of the mirrors.
`...`	not in use

Details

It works by downloading the latest NEWS file (288 Kbytes at the moment, so not huge) from each of the mirror sites in the CRAN mirrors list. If you want to test it on a subset then call getCRANmirrors yourself and subset it somehow.

It runs on the full CRAN list and while designing this package I've yet to find a timeout or error so I'm not sure what will happen if download.file fails. It returns a data frame like you get from getCRANmirrors but with an extra 't' column giving the elapsed time to get the NEWS file.

CAVEATS: if your network has any local caching then these results will be wrong, since your computer will probably be getting the locally cached NEWS file and not the one on the server. Especially if you run it twice. Oh, I should have put cacheOK=FALSE in the download.file - but even that might get overruled somewhere. Also, sites may have good days and bad days, good minutes and bad minutes, your network may be congested on a short-term basis, etc etc.

There may also be a difference in reliability, which would not so easily be measured by an individual user.

Later that year, Barry also wrote Cranography. See: https://www.maths.lancs.ac.uk/~rowlings/R/Cranography/.

Value

a data.frame with details on mirror sites and the time it took to download their NEWS file.

Author(s)

Barry Rowlingson <b.rowlingson@lancaster.ac.uk>

See Also

freegeoip, myip, cranometer

Examples

```
## Not run:
# this can take some time
x <- cranometer()

time_order <- order(x$t)

# a quick overview of the fastest mirrors
head(x[time_order,c(1:4, 9)], 20)

# a dotchart of the fastest mirrors
with(x[rev(time_order),],
 dotchart(t, labels =Name,
 cex = .5, xlab = "Timing of CRAN mirror")
 )

# tail(geonames_df)
# tail(x)
require(plyr)
ss <- !(x$Name == "0-Cloud")
gvis_df <- ddply(x[ss,], .(CountryCode), function(xx) {
  ss <- which.min(xx$t)
  if(length(ss) == 0) ss <- 1
  data.frame(time = xx$t[ss], name = xx$Name[ss] )
})
gvis_df <- gvis_df[!is.na(gvis_df$time), ]

require2("googleVis")
Geo<-gvisGeoMap(gvis_df,
              locationvar = "CountryCode",
               numvar="time",
               hovervar = "name",
               options=list(
                            colors='[0xA5EF63,
                             0xFFB581, 0xFF8747]')
              )
# Display chart
plot(Geo)

## End(Not run)
```

create.global.library *Creates a global library folder*

Description

Creates a global library folder (above the folder R is currently installed in)

Usage

```
create.global.library(global_library_folder)
```

Arguments

global_library_folder	the path of the new global library folder to create. If missing, will be set to R_path/R/library. (for example: "C:/Program Files/R/library")

Value

TRUE/FALSE if we created a new folder or not.

Examples

```
## Not run:
create.global.library()

## End(Not run)
```

download_RStudio_CRAN_data

Download RStudio CRAN mirror data files into a folder

Description

This function download these files based on the code from the download page (http://cran-logs.rstudio.com/) into a temporary folder.

Usage

```
download_RStudio_CRAN_data(
  START = as.Date(Sys.time()) - 5,
  END = as.Date(Sys.time()),
  log_folder = tempdir(),
  trunc_END_date_to_today = TRUE,
  override = FALSE,
  message = TRUE,
  ...
)
```

Arguments

`START`	the defaults is 5 days before today. A character string of the START date for files to be downloaded. The date format is "YYYY-MM-DD".
`END`	the defaults is today. A character string of the END date for files to be downloaded. The date format is "YYYY-MM-DD".
`log_folder`	the folder into which we would like the files to be downloaded to. Default is the temporary folder picked by tempdir.
`trunc_END_date_to_today`	default is TRUE. Makes sure that if END date is later then today, the END date will be change to today (since otherwise, we will only get many 404 errors)
`override`	boolean (default is FALSE) - should the function download files that are already available in the temp folder
`message`	boolean (default is TRUE) - should a message be printed in interesting cases.
`...`	not in use.

Details

RStudio maintains its own CRAN mirror, https://cran.rstudio.com/ and offers its log files.

Value

Returns the value of log_folder.

See Also

download_RStudio_CRAN_data, read_RStudio_CRAN_data,barplot_package_users_per_day

Examples

```
## Not run:
# The first two functions might take a good deal of time to run (depending on the date range)
RStudio_CRAN_data_folder <-
      download_RStudio_CRAN_data(START = '2013-04-02',
                                 END = '2013-04-05')
                                 # around the time R 3.0.0 was released
# RStudio_CRAN_data_folder <- download_RStudio_CRAN_data()
my_RStudio_CRAN_data <- read_RStudio_CRAN_data(RStudio_CRAN_data_folder)

# barplots: (more functions can easily be added in the future)
barplot_package_users_per_day("installr", my_RStudio_CRAN_data)
barplot_package_users_per_day("plyr", my_RStudio_CRAN_data)

## End(Not run)
```

fetch_tag_from_Rd   *Access tag elements from R's Rd file*

Description

A function to extract elements from R's help file.

It is useful, for example, for going through a package and discover who are its authors (useful for me to help me give proper credit in the DESCRIPTION file).

Usage

```
fetch_tag_from_Rd(package, tag = "\\author", ...)
```

Arguments

package	a character string of the package we are interested in.
tag	a character vector of tag(s) to get from a package's Rd files.
...	not in use.

Value

a character vector with the tag's content, and the name of the Rd source of the function the tag came from.

Author(s)

Thomas J. Leeper <thosjleeper@gmail.com>

Source

https://stackoverflow.com/questions/17909081/access-elements-from-rs-rd-file

See Also

package_authors

Examples

```
## Not run:
fetch_tag_from_Rd("installr", "\\author")
fetch_tag_from_Rd("knitr", "\\author")
fetch_tag_from_Rd("lubridate", "\\author")

fetch_tag_from_Rd("installr", "\\source")

# get all the authors for this package
unique(unname(fetch_tag_from_Rd("installr", "\\author")))

fetch_tag_from_Rd("installr", "\\author")
package_authors("installr")


## End(Not run)
```

file.name.from.url *Extract the file name from some URL*

Description

Gets a character of link to some file, and returns the name of the file in this link.

Usage

```
file.name.from.url(URL, rm.params = FALSE)
```

Arguments

URL	Some url to a file.
rm.params	(optional, default=FALSE). Whether or not to remove query parameters from URL.

Details

The install.packages.zip must use this function, since it is crucial that the name of the file into which the ZIPPED package is downloaded to the computer, will have the same name as the file which is online.

Value

The name of the file in the URL

See Also

`install.URL`, `install.packages.zip`

Examples

```
## Not run:
url <- "https://cran.r-project.org/bin/windows/base/R-2.15.3-win.exe"
file.name.from.url(url) # returns: "R-2.15.3-win.exe"

url <- "https://bioconductor.org/packages/R/genetics?version=5.01&f=gz"
file.name.from.url(url, rm.params=TRUE)  # returns: "genetics?version=5.01&f=gz"
file.name.from.url(url, rm.params=FALSE) # returns: "genetics"
file.name.from.url(url)                  # returns: "genetics"

## End(Not run)
```

---

format_RStudio_CRAN_data *Format the RStudio CRAN mirror data into the data.table format*

Description

This function makes sure the the RStudio CRAN mirror data object has correct classes for the columns date, package, country. It also adds the columns weekday and week. Lastly, it also sets a key.

Usage

```
format_RStudio_CRAN_data(dataset, ...)
```

Arguments

`dataset`	the RStudio CRAN mirror data object
`...`	not in use.

Details

RStudio maintains its own CRAN mirror, https://cran.rstudio.com/ and offers its log files.

Value

Returns the re-formated data object.

Author(s)

Felix Schonbrodt, Tal Galili

Source

https://www.nicebread.de/finally-tracking-cran-packages-downloads/

See Also

download_RStudio_CRAN_data, read_RStudio_CRAN_data,barplot_package_users_per_day

Examples

```
## Not run:
# The first two functions might take a good deal of time to run (depending on the date range)
RStudio_CRAN_data_folder <-
      download_RStudio_CRAN_data(START = '2013-04-02',
                                 END = '2013-04-05')
                                 # around the time R 3.0.0 was released
my_RStudio_CRAN_data <- read_RStudio_CRAN_data(RStudio_CRAN_data_folder)
my_RStudio_CRAN_data <- format_RStudio_CRAN_data(my_RStudio_CRAN_data)
head(my_RStudio_CRAN_data)
lineplot_package_downloads(pkg_names = c("ggplot2", "reshape", "plyr", "installr"),
                           dataset = my_RStudio_CRAN_data)

# older plots:
# barplots: (more functions can easily be added in the future)
barplot_package_users_per_day("installr", my_RStudio_CRAN_data)
barplot_package_users_per_day("plyr", my_RStudio_CRAN_data)

## End(Not run)
```

`freegeoip` *Geolocate IP addresses in R*

Description

This R function uses the free freegeoip.net geocoding service to resolve an IP address (or a vector of them) into country, region, city, zip, latitude, longitude, area and metro codes.

The function require rjson.

Usage

```
freegeoip(
  ip = myip(),
  format = ifelse(length(ip) == 1, "list", "dataframe"),
  ...
)
```

Arguments

`ip`	a character vector of ips (default is the output from myip)
`format`	format of the output. Either "list" (default) or "data.frame"
`...`	not in use

Value

a list or data.frame with details on your geo location based on the freegeoip.net service.

Author(s)

Heuristic Andrew (see source for details)

Source

`https://heuristically.wordpress.com/2013/05/20/geolocate-ip-addresses-in-r/`.

See Also

freegeoip, myip, cranometer

Examples

```
## Not run:
freegeoip()

## http://www.students.ncl.ac.uk/keith.newman/r/maps-in-r
# install.packages("maps")
# install.packages("mapdata")
library(maps)
library(mapdata)    # Contains the hi-resolution points that mark out the countries.
map('worldHires')
require(installr)
myip_details <- freegeoip(myip())
my_lati <- myip_details$latitude
my_long <- myip_details$longitude
points(my_lati,my_long,col=2,pch=18, cex = 1)
# lines(c(my_lati,0) ,c(my_long, 50), col = 2)#'

## End(Not run)
```

get.installed.R.folders

Returns folder names with R installations

Description

The function finds the folders where there are R installations. This is important for deciding what to uninstall, and where from and to to move libraries. This function ignores installations of R-devel at this point. Also, this function is based on only looking at the folders above the current installation of R. If there are other installations of R outside the above folder, they will not be listed.

Usage

```
get.installed.R.folders(sort_by_version = TRUE, add_version_to_name = TRUE)
```

Arguments

`sort_by_version`	should the returned vector be sorted by the version number? (default is yes - so that the first element is of the newest version of R) should the user be given the option to choose between which two libraries to copy the packages? If FALSE (default), the folders are copied from the before-newest R installation to the newest R installation.
`add_version_to_name`	should the version number be added to the vector of folders? (default is yes)

Value

Returns a character vector (possibly named, possibly sorted) of the folders where there are R installations.

See Also

copy.packages.between.libraries

Examples

```
## Not run:
get.installed.R.folders()
# returns the sorted and named vector of
# folder names where R is installed (in different versions).
#  The first element is
# the folder of the newest version of R.

get.installed.R.folders(F, F)
# returns the folder names where R is
# installed (in different versions) - no sorting of
# the folder names was performed
```

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

`get_latest_r_version` *Fetches latest R version from CRAN*

Description

Returns the latest version of R available on CRAN as an R system version object.

Usage

```
get_latest_r_version(cran_mirror = "https://cran.rstudio.com/")
```

Arguments

`cran_mirror`	The CRAN mirror to use

References

`https://cran.r-project.org/bin/windows/base/`

See Also

getRversion

`get_pid` *Find the pid of a process by name*

Description

Returns a vector with the process ID (pid) for all processes with a particular name.

Usage

```
get_pid(process, exact = FALSE, ...)
```

Arguments

`process`	a character vector of process names.
`exact`	logical (FALSE). should we get exact match to process name, or can we use just partial matching.
`...`	not used.

Value

an integer vector with the process ID (pid) of the processes.

References

tasklist details from microsoft homepage: https://learn.microsoft.com/en-us/previous-versions/windows/it-pro/windows-xp/bb491010(v=technet.10)

See Also

get_tasklist, get_Rscript_PID, get_pid, kill_pid, kill_all_Rscript_s, pskill

Examples

```
## Not run:
get_pid("rsession") # finds it
get_pid("rsession", exact = TRUE) # doesn't find it
get_pid("rsession.exe", exact = TRUE) # finds it
get_pid(c("wininit", "winlogon"), exact = TRUE) # doesn't find it
get_pid(c("wininit", "winlogon")) # finds it

## End(Not run)
```

---

get_Rscript_PID *Get the running "Rscript" processes PID*

Description

Returns a vector with the process ID (pid) of the "Rscript" processes which are currently running.

Usage

```
get_Rscript_PID(...)
```

Arguments

...	not used.

Value

an integer vector with the process ID (pid) of the "Rscript" processes.

References

tasklist details from microsoft homepage: https://learn.microsoft.com/en-us/previous-versions/windows/it-pro/windows-xp/bb491010(v=technet.10)

See Also

get_tasklist, get_Rscript_PID, get_pid, kill_pid, kill_all_Rscript_s, pskill

Examples

```
## Not run:
# create several running processes of Rscript (to shitdown)
system("Rscript -e repeat{2+2}", wait = FALSE) # this process should be stuck
system("Rscript -e repeat{2+2}", wait = FALSE) # this process should be stuck
# here are there pid numbers:
get_Rscript_PID()
# let's kill them:
kill_all_Rscript_s()
# they are gone...
get_Rscript_PID() # we no longer have Rscripts running

## End(Not run)
```

`get_tasklist` *Get the running processes in windows task manager*

Description

Returns a data.frame with the current running processes (Windows only).

Usage

```
get_tasklist(...)
```

Arguments

`...`	not used.

Value

a data.frame with the current running processes.

References

tasklist details from microsoft homepage: `https://learn.microsoft.com/en-us/previous-versions/windows/it-pro/windows-xp/bb491010(v=technet.10)`

See Also

`get_tasklist`, `get_Rscript_PID`, `get_pid`, `kill_pid`, `kill_all_Rscript_s`, pskill `kill_pid`, `kill_all_Rscript_s`

Examples

```
## Not run:
# create several running processes of Rscript (to shitdown)
system("Rscript -e repeat{2+2}", wait = FALSE) # this process should be stuck
system("Rscript -e repeat{2+2}", wait = FALSE) # this process should be stuck
# here are there pid numbers:
get_Rscript_PID()
# let's kill them:
kill_all_Rscript_s()
# they are gone...
get_Rscript_PID() # we no longer have Rscripts running

## End(Not run)
```

`install.7zip` *Downloads and installs 7-Zip for windows*

Description

Allows the user to downloads and install the latest version of 7-Zip for Windows.

Usage

```
install.7zip(
  page_with_download_url = "https://www.7-zip.org/download.html",
  ...
)
```

Arguments

`page_with_download_url`	the URL of the 7-Zip download page.
`...`	extra parameters to pass to install.URL

Details

7-Zip is open source software. Most of the source code is under the GNU LGPL license. The unRAR code is under a mixed license: GNU LGPL + unRAR restrictions. Check license information here: 7-Zip license. You can use 7-Zip on any computer, including a computer in a commercial organization. You don't need to register or pay for 7-Zip. *The main features of 7-Zip *High compression ratio in 7z format with LZMA and LZMA2 compression *Supported formats: **Packing / unpacking: 7z, XZ, BZIP2, GZIP, TAR, ZIP and WIM **Unpacking only: ARJ, CAB, CHM, CPIO, CramFS, DEB, DMG, FAT, HFS, ISO, LZH, LZMA, MBR, MSI, NSIS, NTFS, RAR, RPM, SquashFS, UDF, VHD, WIM, XAR and Z. For ZIP and GZIP formats, 7-Zip provides a compression ratio that is 2-10 *Strong AES-256 encryption in 7z and ZIP formats *Self-extracting capability for 7z format *Integration with Windows Shell *Powerful File Manager *Powerful command line version *Plugin for FAR Manager *Localizations for 79 languages

Value

TRUE/FALSE - was the installation successful or not.

References

- 7-zip homepage: https://www.7-zip.org/

Examples

```
## Not run:
install.7zip() # installs the latest version of 7-Zip

## End(Not run)
```

install.CMake *Downloads and installs CMake for windows*

Description

Allows the user to downloads and install the latest version of CMake for Windows.

Usage

```
install.CMake(
  URL = "https://github.com/Kitware/CMake/releases",
  cmake_version = NULL,
  ...
)
```

Arguments

URL	the URL of the CMake download page.
cmake_version	NULL by default, but a user can supply the version as a character.
...	extra parameters to pass to install.URL

Details

CMake is a family of tools designed to build, test and package software. CMake is used to control the software compilation process using simple platform and compiler independent configuration files. CMake generates native makefiles and workspaces that can be used in the compiler environment of your choice. If run NOT on Windows (i.e.: max/linux), this would download the file into the working directory. (getwd).

Value

TRUE/FALSE - was the installation successful or not.

References

- CMake homepage: https://cmake.org/download/

Examples

```
## Not run:
install.CMake() # installs the latest version of CMake

## End(Not run)
```

install.conda *Downloads and installs miniconda*

Description

Downloads and installs the latest version of miniconda for Windows.

Usage

```
install.conda(version = 3, bitNo = "auto", ...)
```

Arguments

`version`	2 or 3. Default is 3
`bitNo`	32 or 64. Defaults is "auto" to check system.
`...`	extra parameters to pass to install.URL

Details

Miniconda is minimal version of anaconda for python.

Value

TRUE/FALSE - was the installation successful or not.

Author(s)

Tal Galili and A. Jonathan R. Godfrey and Chanyub Park

Examples

```
## Not run:
install.conda()
install.conda(version = 3)
install.conda(3)

## End(Not run)
```

`install.Cygwin` *Downloads and installs Cygwin for windows*

Description

Allows the user to downloads and install the latest version of Cygwin for Windows.

Usage

```
install.Cygwin(bit = 32, ...)
```

Arguments

`bit`	Specify 32 bit or 64 for your particular version of Windows.
`...`	extra parameters to pass to install.URL

Details

Cygwin is a collection of tools which provide a Linux look and feel environment for Windows.

Value

TRUE/FALSE - was the installation successful or not.

References

- Cygwin homepage: https://www.cygwin.com/

Examples

```
## Not run:
install.Cygwin() # installs the latest version of Cygwin

## End(Not run)
```

`install.FFmpeg` *Downloads and installs FFmpeg for windows*

Description

Allows the user to downloads the latest version of FFmpeg for Windows. IMPORTANT NOTE: The user (YOU) are responsible for unpacking the 7zip file into the relevant directory. All that this function does is to download the 7zip file and "run" it.

Usage

```
install.FFmpeg(
  page_with_download_url = "http://ffmpeg.zeranoe.com/builds/",
  ...
)
```

Arguments

`page_with_download_url`	the URL of the FFmpeg download page.
`...`	extra parameters to pass to install.URL

Details

FFmpeg is a complete, cross-platform solution to record, convert and stream audio and video. It includes libavcodec - the leading audio/video codec library. See the documentation for a complete feature list and the Changelog for recent changes. This function downloads current releases and NOT the Development Snapshots. This function is useful for saveVideo() in the animation package.

References

- FFmpeg homepage: `http://FFmpeg.org/`

Examples

```
## Not run:
install.FFmpeg() # installs the latest version of FFmpeg

## End(Not run)
```

`install.git` *Downloads and installs git and git-gui for windows*

Description

Allows the user to downloads and install the latest version of git for Windows.

Usage

```
install.git(URL = "https://git-scm.com/download/win", version = 64, ...)
```

Arguments

`URL`	the URL of the git download page.
`version`	numeric - either 32 or 64 (default)
`...`	extra parameters to pass to install.URL

Details

Git is a distributed revision control and source code management system with an emphasis on speed.

Value

TRUE/FALSE - was the installation successful or not.

References

git homepage: `https://git-scm.com/` git download page: `https://git-scm.com/download/win`

Examples

```
## Not run:
install.git() # installs the latest version of git

## End(Not run)
```

`install.GitHub` *Downloads and installs GitHub for windows*

Description

Allows the user to downloads and install the latest version of GitHub for Windows.

Usage

```
install.GitHub(
  URL = "http://github-windows.s3.amazonaws.com/GitHubSetup.exe",
  ...
)
```

Arguments

`URL`	the URL of the GitHub download page.
`...`	extra parameters to pass to install.URL

Details

"The easiest way to use Git on Windows." (at least so they say...)

Value

TRUE/FALSE - was the installation successful or not.

References

- GitHub homepage: https://github.com/
- GitHub for windows download page: https://desktop.github.com/

Examples

```
## Not run:
install.GitHub() # installs the latest version of GitHub for windows

## End(Not run)
```

`install.GraphicsMagick` *Downloads and installs GraphicsMagick for windows*

Description

Allows the user to downloads and install the latest version of GraphicsMagick for Windows.

Usage

```
install.GraphicsMagick(
  URL = "https://sourceforge.net/projects/graphicsmagick/files/latest/download",
   ...
)
```

Arguments

`URL`	the URL of the ImageMagick download page.
`...`	deprecated

Details

GraphicsMagick is the swiss army knife of image processing. Comprised of 282K physical lines (according to David A. Wheeler's SLOCCount) of source code in the base package (or 964K including 3rd party libraries) it provides a robust and efficient collection of tools and libraries which support reading, writing, and manipulating an image in over 88 major formats including important formats like DPX, GIF, JPEG, JPEG-2000, PNG, PDF, PNM, and TIFF. This function downloads Win32 dynamic at 16 bits-per-pixel. No direct file is available to run, You'll need to open/run the file that your browser is now downloading.

Value

TRUE/FALSE - was the installation successful or not.

References

- GraphicsMagick homepage: http://www.graphicsmagick.org/

Examples

```
## Not run:
install.GraphicsMagick() # installs the latest version of GraphicsMagick

## End(Not run)
```

`install.ImageMagick` *Downloads and installs ImageMagick for windows*

Description

Allows the user to downloads and install the latest version of ImageMagick for Windows.

Usage

```
install.ImageMagick(
  URL = "https://www.imagemagick.org/script/download.php",
  ...
)
```

Arguments

`URL`	the URL of the ImageMagick download page.
`...`	extra parameters to pass to install.URL

Details

ImageMagick is a software suite to create, edit, compose, or convert bitmap images. It can read and write images in a variety of formats (over 100) including DPX, EXR, GIF, JPEG, JPEG-2000, PDF, PhotoCD, PNG, Postscript, SVG, and TIFF. Use ImageMagick to resize, flip, mirror, rotate, distort, shear and transform images, adjust image colors, apply various special effects, or draw text, lines, polygons, ellipses and Bezier curves. This function downloads Win32 dynamic at 16 bits-per-pixel.

Value

TRUE/FALSE - was the installation successful or not.

References

- ImageMagick homepage: `http://www.imagemagick.org/script/index.php`

Examples

```
## Not run:
install.ImageMagick() # installs the latest version of ImageMagick

## End(Not run)
```

`install.inno` *Downloads and installs Inno Setup*

Description

Downloads and installs Inno Setup's stable release

Usage

```
install.inno(quick_start_pack = FALSE, encryption_module = TRUE, ...)
```

Arguments

`quick_start_pack`	logical (default is FALSE) - The Inno Setup QuickStart Pack includes Inno Setup and Inno Script Studio script editor. See Third-Party Files page for more information.
`encryption_module`	logical (default is TRUE) - Inno Setup's Encryption Module
`...`	extra parameters to pass to install.URL

Details

Inno Setup is a free installer for Windows programs. First introduced in 1997, it currently rivals many commercial installers in feature set and stability.

See Features for more information.

Value

TRUE/FALSE - was the installation successful or not.

Author(s)

Tal Galili and Jonathan M. Hill

Examples

```
## Not run:
install.inno()
install.inno(quick_start_pack = TRUE)

## End(Not run)
```

install.java *Install Java - downloads and set path openjdk*

Description

Downloads and set path the latest version of openjdk for Windows.

Usage

```
install.java(
  version = 11,
  page_with_download_url = "https://jdk.java.net/java-se-ri/",
  path = "C:/java"
)

install.Java(
  version = 11,
  page_with_download_url = "https://jdk.java.net/java-se-ri/",
  path = "C:/java"
)

install.jdk(
  version = 11,
  page_with_download_url = "https://jdk.java.net/java-se-ri/",
  path = "C:/java"
)

install.Jdk(
  version = 11,
  page_with_download_url = "https://jdk.java.net/java-se-ri/",
  path = "C:/java"
)

install.openjdk(
  version = 11,
  page_with_download_url = "https://jdk.java.net/java-se-ri/",
  path = "C:/java"
)

install.OpenJdk(
  version = 11,
  page_with_download_url = "https://jdk.java.net/java-se-ri/",
  path = "C:/java"
)
```

Arguments

`version`	9 or 10 is passible. Default is 11.

`page_with_download_url`	where to download. Default is `https://jdk.java.net/java-se-ri/11`
`path`	where to set java. Default path is C:/java

Details

install openjdk 9 or 10 version for windows.

Value

TRUE/FALSE - was the installation successful or not.

Author(s)

Chan-Yub Park And Tal Galili

Examples

```
## Not run:
install.java()
install.java(version = 11)
install.java(11)

## End(Not run)
```

`install.LaTeX2RTF` *Downloads and installs LaTeX2RTF for windows*

Description

Allows the user to downloads and install the latest version of LaTeX2RTF for Windows.

Usage

```
install.LaTeX2RTF(
  page_with_download_url = "http://sourceforge.net/projects/latex2rtf/",
  ...
)
```

Arguments

`page_with_download_url`	the URL of the SWFTools download page.
`...`	extra parameters to pass to install.URL

Details

Latex2rtf tries to convert your LaTeX file into a RTF file for opening in Microsoft Word. The general idea is to try and get the things that computers are good at correct: character conversion, graphic conversion, etc. Page layout suffers because control in RTF is pretty pathetic compared to TeX. Consequently, it is likely that manual reformatting will be needed.

Value

TRUE/FALSE - was the installation successful or not.

References

- SWFTools homepage: https://latex2rtf.sourceforge.net/

Examples

```
## Not run:
install.LaTeX2RTF() # installs the latest version of LaTeX2RTF

## End(Not run)
```

`install.LyX` *Downloads and installs LyX for windows*

Description

Allows the user to downloads and install the latest version of LyX for Windows.

Usage

```
install.LyX(
  page_with_download_url = "http://www.lyx.org/Download",
  new_installation,
  ...
)
```

Arguments

`page_with_download_url`	the URL of the LyX download page.
`new_installation`	boolean. TRUE means we should make a new installation of LyX. FALSE means to update an existing installation. Missing - prompts the user to decide.
`...`	extra parameters to pass to install.URL

Details

LyX is an advanced open source document processor running on Linux/Unix, Windows, and Mac OS X. It is called a "document processor", because unlike standard word processors, LyX encourages an approach to writing based on the structure of your documents, not their appearance.#' LyX lets you concentrate on writing, leaving details of visual layout to the software. LyX automates formatting according to predefined rule sets, yielding consistency throughout even the most complex documents. LyX produces high quality, professional output - using LaTeX, an open source, industrial strength typesetting engine, in the background.

Value

TRUE/FALSE - was the installation successful or not.

References

- LyX homepage: `http://www.lyx.org/`

Examples

```
## Not run:
install.LyX() # installs the latest version of LyX

## End(Not run)
```

`install.MikTeX` *Downloads and installs MikTeX for windows*

Description

Allows the user to downloads and install the latest version of MikTeX for Windows.

Usage

```
install.MikTeX(page_with_download_url = "https://miktex.org/download", ...)
```

Arguments

`page_with_download_url`	the URL of the MikTeX download page.
`...`	extra parameters to pass to install.URL

Details

MiKTeX is a typesetting system for Microsoft Windows that is developed by Christian Schenk. It consists of an implementation of TeX and a set of related programs. MiKTeX provides the tools necessary to prepare documents using the TeX/LaTeX markup language, as well a simple tex editor (TeXworks).

MiKTeX is essential for using Sweave, knitr, and creating Vignette for R packages.

Value

TRUE/FALSE - was the installation successful or not.

References

MikTeX homepage: `http://miktex.org/` MikTeX download page: `http://miktex.org/download`

Examples

```
## Not run:
install.MikTeX() # installs the latest version of MikTeX 62 bit

## End(Not run)
```

`install.nodejs` *Downloads and installs nodejs LTS or Current*

Description

Downloads and installs the latest version of nodejs LTS or Current for Windows.

Usage

```
install.nodejs(
  page_with_download_url = "https://nodejs.org/en/download/",
  version_number = "LTS",
  ...
)
```

Arguments

`page_with_download_url`	a link to the list of download links for Nodejs
`version_number`	Either LTS or Current. Version LTS will lead to download of v6.11.X
`...`	extra parameters to pass to install.URL

Details

Nodejs is a programming language which has two versions under active development. Make sure you know which version is required for the code you have to run, or alternatively, make sure you are developing code that is fit for your chosen version of Nodejs.

Value

TRUE/FALSE - was the installation successful or not.

Author(s)

Tal Galili and A. Jonathan R. Godfrey and Chanyub Park

Examples

```
## Not run:
install.nodejs()
install.nodejs(,"Current")
install.nodejs(,"LTS")

## End(Not run)
```

install.notepadpp *Downloads and installs Notepad++ for windows*

Description

Allows the user to downloads and install the latest version of Notepad++ for Windows.

Usage

```
install.notepadpp(
  page_with_download_url = "https://notepad-plus-plus.org/downloads/",
  ...
)
```

Arguments

`page_with_download_url`	the URL of the Notepad++ download page.
`...`	extra parameters to pass to install.URL

Details

Notepad++ is a free (as in "free speech" and also as in "free beer") source code editor and Notepad replacement that supports several languages. Running in the MS Windows environment, its use is governed by GPL License. Based on the powerful editing component Scintilla, Notepad++ is written in C++ and uses pure Win32 API and STL which ensures a higher execution speed and smaller program size. By optimizing as many routines as possible without losing user friendliness, Notepad++ is trying to reduce the world carbon dioxide emissions. When using less CPU power, the PC can throttle down and reduce power consumption, resulting in a greener environment.

Value

invisible TRUE/FALSE - was the installation successful or not.

References

homepage: https://notepad-plus-plus.org/ download page: https://notepad-plus-plus.org/downloads/

Examples

```
## Not run:
install.notepadpp() # installs the latest version of Notepad++

## End(Not run)
```

`install.npptor` *Downloads and installs NppToR for windows*

Description

Allows the user to downloads and install the latest version of NppToR extension for Notepad++ for Windows.

Usage

```
install.npptor(
  URL = "http://sourceforge.net/projects/npptor/files/npptor%20installer/",
  ...
)
```

Arguments

`URL`	the URL of the Notepad++ download page.
`...`	extra parameters to pass to install.URL

Details

Similar to the windows R gui built in editor, NppToR aims to extend the functionality of code passing to the Notepad++ code editor. In addition to passing to the R gui, NppToR provides optional passing to a PuTTY window for passing to an R instance a remote machine.

NppToR is a companion utility that facilitates communication between R and Notepad++. It provides code passing from Notepad++ into the windows R Gui. NppToR also provides an autocompletion database which is dynamically generated from the users' R library of packages, thanks to an addition by Yihui Xie. Notepad++ provides built it R code highlighting and folding.

Value

invisible TRUE/FALSE - was the installation successful or not.

References

download page: https://sourceforge.net/projects/npptor/

Examples

```
## Not run:
install.npptor() # installs the latest version of NppToR

## End(Not run)
```

---

install.packages.zip *Downloads and installs a ZIP R package Binary (for Windows) from a URL*

Description

Gets a character with a link to an R package Binary, downloads it, and installs it.

Usage

```
install.packages.zip(zip_URL)
```

Arguments

zip_URL	a link to a ZIP R package Binary.

Details

To my knowledge, there is currently three ways to install packages on R: 1. To get the package through a repository (such as CRAN or RForge) through install.packages. 2. To manually download a ZIP file locally to the computer, and use install.packages on it. 3. To get the package from github, by using devtools (but this will require you to first install RTools, and not everyone wishes to do it for just some package). This function aims to combine option 1 and 2, by automatically downloading the ZIP file locally and then running install.packages on it. After being downloaded and installed, the binary is erased from the computer.

Value

Invisible NULL

See Also

install.packages, installPackages

Examples

```
## Not run:
install.packages.zip("https://cran.r-project.org/bin/windows/contrib/r-release/devtools_1.1.zip")

## End(Not run)
```

`install.pandoc` *Downloads and installs pandoc*

Description

Downloads and installs the latest version of pandoc for Windows.

Usage

```
install.pandoc(
  URL = "https://github.com/jgm/pandoc/releases",
  use_regex = TRUE,
  to_restart,
  ...
)
```

Arguments

`URL`	a link to the list of download links of pandoc
`use_regex`	(default TRUE) - deprecated (kept for legacy purposes).
`to_restart`	boolean. Should the computer be restarted after pandoc is installed? (if missing then the user is prompted for a decision)
`...`	extra parameters to pass to install.URL

Details

pandoc is a free open source software for converting documents from many filetypes to many filetypes. For details, see `https://johnmacfarlane.net/pandoc/`.

Credit: the code in this function is based on GERGELY DAROCZIs coding in his answer on the Q&A forum StackOverflow, and also G. Grothendieck for the non-XML addition to the function. I thank them both!

Value

TRUE/FALSE - was the installation successful or not.

Author(s)

GERGELY DAROCZI, G. Grothendieck, Tal Galili

Source

`https://stackoverflow.com/questions/15071957/is-it-possible-to-install-pandoc-on-windows-using-an-`

Examples

```
## Not run:
install.pandoc()

## End(Not run)
```

install.python *Downloads and installs python 2 or 3*

Description

Downloads and installs the latest version of python 2 or 3 for Windows.

Usage

```
install.python(
  page_with_download_url = "https://www.python.org/downloads/windows/",
  version_number = 3,
  x64 = TRUE,
  ...
)
```

Arguments

`page_with_download_url`	a link to the list of download links for Python
`version_number`	Either 2 or 3. Version 2/3 will lead to download of v2.7.xx/3.6.xx respectively.
`x64`	logical: fetch a 64 bit version. default is TRUE; used to check architecture of current R session.
`...`	extra parameters to pass to install.URL

Details

Python is a programming language which has two versions under active development. Make sure you know which version is required for the code you have to run, or alternatively, make sure you are developing code that is fit for your chosen version of Python. In addition, the Python installers are specific to 32 or 64 bit windows architectures.

Value

TRUE/FALSE - was the installation successful or not.

Author(s)

Tal Galili and A. Jonathan R. Godfrey

Examples

```
## Not run:
install.python()
install.python(,3)
install.python(,2)

## End(Not run)
```

---

install.R *Downloads and installs the latest R version*

Description

Fetches the latest (not development!) R version

Usage

```
install.R(
  page_with_download_url = "https://cran.rstudio.com/bin/windows/base/",
  pat = "R-[0-9.]+.+-win\\.exe",
  to_checkMD5sums = TRUE,
  keep_install_file = FALSE,
  download_dir = tempdir(),
  silent = FALSE,
  ...
)
```

Arguments

`page_with_download_url`	URL from which the latest stable version of R can be downloaded from.
`pat`	the pattern of R .exe file to download
`to_checkMD5sums`	Should we check that the new R installation has the files we expect it to (by checking the MD5 sums)? default is TRUE. It assumes that the R which was installed is the latest R version.
`keep_install_file`	If TRUE - the installer file will not be erased after it is downloaded and run.
`download_dir`	A character of the directory into which to download the file. (default is tempdir())
`silent`	If TRUE - enables silent installation mode.
`...`	extra parameters to pass to install.URL

Details

If you are not sure if you need to update R or not, It is better to use updateR for updating R, since it includes more options. But in case you wish to only install R, with no other steps taken (for example, taking care of your old packages), then you can use install.R()

See the install.Rdevel function for installing the latest R-devel version.

Value

TRUE/FALSE - was the installation of R successful or not.

References

https://cran.r-project.org/bin/windows/base/

See Also

uninstall.R, install.Rdevel, updateR, system

Examples

```
## Not run:
install.R()

## End(Not run)
```

`install.Rdevel` *Downloads and installs the latest Rdevel version*

Description

Fetches the latest (development!) R version

Usage

```
install.Rdevel(
  exe_URL = "https://cran.rstudio.com/bin/windows/base/R-devel-win.exe",
  ...
)
```

Arguments

`exe_URL`	A character with a link to an installer file (with the .exe file extension)
`...`	extra parameters to pass to install.URL

Details

This is a development version of R. It likely contains bugs, so be careful if you use it. Please don't report bugs in this version through the usual R bug reporting system, please report them on the r-devel mailing list —but only if they persist for a few days.

Value

TRUE/FALSE - was the installation of R successful or not.

References

https://cran.r-project.org/bin/windows/base/rdevel.html

See Also

install.R, updateR

Examples

```
## Not run:
install.Rdevel()

## End(Not run)
```

`install.RStudio` *Downloads and installs RStudio for windows*

Description

Allows the user to downloads and install the latest version of RStudio for Windows.

Usage

```
install.RStudio(page_with_download_url, ...)
```

Arguments

`page_with_download_url`	the URL of the RStudio download page.
`...`	extra parameters to pass to install.URL

Details

RStudio is a free and open source integrated development environment (IDE) for R, a programming language for statistical computing and graphics.

Value

TRUE/FALSE - was the installation successful or not.

References

- RStudio homepage: https://posit.co/

Examples

```
### devtools::source_url
## Not run:
install.RStudio() # installs the latest version of RStudio

## End(Not run)
```

`install.Rtools` *Downloads and installs Rtools*

Description

Install compatible version of Rtools for Windows. By default, the function searches if a compatible Rtools is installed, if not, it offers to install the latest compatible version.

Usage

```
install.Rtools(check = TRUE, check_r_update = TRUE, GUI = TRUE, ...)
```

Arguments

`check`	checks if we need to install Rtools or not.
`check_r_update`	checks if there is an R update available (ignores patch versions), if so, asks if user wants to install the R update first. (defaults to TRUE)
`GUI`	Should a GUI be used when asking the user questions? (defaults to TRUE)
`...`	extra parameters to pass to install.URL

Details

Rtools is a collection of software for building packages for R under Microsoft Windows, or for building R itself (version 1.9.0 or later). The original collection was put together by Prof. Brian Ripley; it is currently being maintained by Duncan Murdoch.

Value

invisible(TRUE/FALSE) - was the installation successful or not.

References

Rtools homepage: `https://cran.r-project.org/bin/windows/Rtools/`

Examples

```
## Not run:
# installs the latest compatible version of Rtools if a compatible version is not yet installed
install.Rtools()
# (re)installs the latest compatible version of Rtools
install.Rtools(check = F)

# skip R version check
install.Rtools(checkRupdate = F)

## End(Not run)
```

`install.SWFTools` *Downloads and installs SWFTools for windows*

Description

Allows the user to downloads and install the latest version of SWFTools for Windows.

Usage

```
install.SWFTools(
  page_with_download_url = "http://swftools.org/download.html",
  ...
)
```

Arguments

`page_with_download_url`	the URL of the SWFTools download page.
`...`	extra parameters to pass to install.URL

Details

SWFTools is a collection of utilities for working with Adobe Flash files (SWF files). The tool collection includes programs for reading SWF files, combining them, and creating them from other content (like images, sound files, videos or sourcecode). SWFTools is released under the GPL. This function downloads current releases and NOT the Development Snapshots. This function is useful for saveSWF() in the animation package.

Value

TRUE/FALSE - was the installation successful or not.

References

- SWFTools homepage: `http://swftools.org/`

Examples

```
## Not run:
install.SWFTools() # installs the latest version of SWFTools

## End(Not run)
```

`install.Texmaker` *Downloads and installs Texmaker for windows*

Description

Allows the user to downloads and install the latest version of Texmaker for Windows.

Usage

```
install.Texmaker(
  URL = "http://www.xm1math.net/texmaker/texmakerwin32_install.exe",
  ...
)
```

Arguments

`URL`	the URL of the GitHub download page.
`...`	extra parameters to pass to install.URL

Details

Texmaker is a free, modern and cross-platform LaTeX editor for linux, macosx and windows systems that integrates many tools needed to develop documents with LaTeX, in just one application.

Value

logical - was the installation successful or not.

References

- Texmaker homepage: `http://www.xm1math.net/texmaker/`

Examples

```
## Not run:
install.Texmaker() # installs the latest version of Texmaker for windows

## End(Not run)
```

install.URL — *Downloads and runs a .exe installer file for some software from a URL*

Description

Gets a character with a link to an installer file, downloads it, runs it, and then erases it.

Usage

```
install.URL(
  exe_URL,
  keep_install_file = FALSE,
  wait = TRUE,
  download_dir = tempdir(),
  message = TRUE,
  installer_option = NULL,
  download_fun = download.file,
  ...
)
```

Arguments

`exe_URL`	A character with a link to an installer file (with the .exe file extension)
`keep_install_file`	If TRUE - the installer file will not be erased after it is downloaded and run.
`wait`	should the R interpreter wait for the command to finish? The default is to NOT wait.
`download_dir`	A character of the directory into which to download the file. (default is tempdir())
`message`	boolean. Should a message on the file be printed or not (default is TRUE)
`installer_option`	A character of the command line arguments
`download_fun`	a function to use for downloading. Default is download.file. We can also use curl_download (but it doesn't give as good of an output while downloading the file).
`...`	parameters passed to 'shell'

Details

This function is used by many functions in the installr package. The .exe file is downloaded into a temporary directory, where it is erased after installation has started (by default - though this can be changed)

Value

invisible(TRUE/FALSE) - was the installation successful or not. (this is based on the output of shell of running the command being either 0 or 1/2. 0 means the file was successfully installed, while 1 or 2 means there was a failure in running the installer.)

Author(s)

GERGELY DAROCZI, Tal Galili

See Also

shell

Examples

```
## Not run:
install.URL("adfadf") # shows the error produced when the URL is not valid.

## End(Not run)
```

---

installr *Installing software from R*

Description

Gives the user the option to download software from within R.

Usage

```
installr(GUI = TRUE, ...)
```

Arguments

GUI	a logical indicating whether a graphics menu should be used if available. If TRUE, and on Windows, it will use winDialog, otherwise it will use menu.
...	not in use

Value

TRUE/FALSE - if the software was installed successfully or not.

See Also

updateR, install.R, install.RStudio, install.Rtools, install.pandoc, install.MikTeX, install.git, install.git, install.GraphicsMagick, install.ImageMagick, check.for.updates.R, install.URL, install.packages.zip,

Examples

```
## Not run:
installr()

## End(Not run)
```

`is.empty` *Checks if an object is empty (e.g: of zero length)*

Description

Checks if an object is empty (e.g: of zero length) and returns TRUE/FALSE

Usage

```
is.empty(x, mode = NULL, ...)
```

Arguments

`x`	an object
`mode`	is the object an empty (zero length) object of this mode (can be "integer", "numeric", and so on...)
`...`	none are available.

Details

Uses identical and avoids any attribute problems by using the fact that it is the empty set of that class of object and combine it with an element of that class.

Value

Returns TRUE/FALSE if the object is empty or not.

Author(s)

James (`https://stackoverflow.com/users/269476/james`)

Source

`https://stackoverflow.com/questions/6451152/how-to-catch-integer0`

See Also

integer, identical

Examples

```
is.empty(integer(0)) #TRUE
is.empty(0L)         #FALSE
is.empty(numeric(0)) #TRUE
is.empty(NA) # FALSE
is.empty(FALSE) # FALSE
is.empty(NULL) # FALSE (with a warning)

a <- which(1:3 == 5)
b <- numeric(0)
is.empty(a)
is.empty(a,"numeric")
is.empty(b)
is.empty(b,"integer")
```

`is.exe.installed` *Checks if some .exe is available in on the Windows machine search PATH*

Description

Checks the existence of an .exe extention in the search path for executable files

Usage

```
is.exe.installed(exe_file)
```

Arguments

`exe_file`	a character with the name of the executable to be looked for

Value

A boolean vector indicating the existence of each program's availability on the system.

Examples

```
## Not run:
is.exe.installed(c("zip.exe", "Rgui.exe", "blablabla")) # [1]  TRUE  TRUE FALSE
is.exe.installed("7z")

## End(Not run)
```

is.Rgui *Checks if the R session is running within Rgui (Windows OS)*

Description

Returns TRUE/FALSE if the R session is running within Rgui or not.

Usage

```
is.Rgui(...)
```

Arguments

`...`	none are available.

Details

This function is used in order to check if a GUI can be added to the session or not.

Value

Returns TRUE/FALSE if the R session is running within Rgui or not.

See Also

is.RStudio, is.windows

Examples

```
## Not run:
is.Rgui()

## End(Not run)
```

is.RStudio *Checks if the R session is running within RStudio*

Description

Returns TRUE/FALSE if the R session is running within RStudio or not.

Usage

```
is.RStudio(...)
```

Arguments

`...`	none are available.

Details

This function is used in order to check if a GUI can be added to the session or not.

Value

Returns TRUE/FALSE if the R session is running within RStudio or not.

Examples

```
## Not run:
is.RStudio()

## End(Not run)
```

`is.windows` *Checks if the running OS is windows*

Description

Returns TRUE/FALSE if the R session is running on Windows or not.

Usage

```
is.windows(...)
```

Arguments

`...`	none are available.

Details

This function is run when the 'installr' package is first loaded in order to check if the current running OS is Windows. If you are running a different OS, then the installr package (at its current form) does not have much to offer you.

Value

Returns TRUE/FALSE if the R session is running on Windows or not.

Examples

```
## Not run:
is.windows() # returns TRUE on my machine.

## End(Not run)
```

`is.x64` *Checks if the running OS is x64*

Description

Returns TRUE/FALSE if the R session is running on Windows 64-bit or not.

Usage

```
is.x64(...)
```

Arguments

`...`	none are available.

Details

This function is run when the 'installr' package is first loaded in order to check if the current running OS is Windows 64-bit. If you are running a different OS, then the installr package (at its current form) does not have much to offer you.

Value

Returns TRUE/FALSE if the R session is running on Windows 64-bit or not.

Examples

```
is.x64() # returns TRUE on my machine.
```

`is_in_.First_in_Rprofile.site` *Remove a code line from Rprofile.site .First*

Description

Goes through

Usage

```
is_in_.First_in_Rprofile.site(code, fixed = TRUE, ...)
```

Arguments

`code`	A character scalar with code to add at the beginning of the .First function in Rprofile.site
`fixed`	passed to grep
`...`	passed to grep

Value

logical, if code is in Rprofile.site or not.

References

https://stackoverflow.com/questions/1395301/how-to-get-r-to-recognize-your-working-directory-as-it
https://stackoverflow.com/questions/1189759/expert-r-users-whats-in-your-rprofile
http://www.noamross.net/archives/2012-11-02-rprofile/ https://www.statmethods.net/interface/customizing.html

Examples

```
## Not run:
is_in_.First_in_Rprofile.site("suppressMessages(library(installr))") # FALSE
add_to_.First_in_Rprofile.site("suppressMessages(library(installr))")
is_in_.First_in_Rprofile.site("suppressMessages(library(installr))") # TRUE
remove_from_.First_in_Rprofile.site("suppressMessages(library(installr))")
is_in_.First_in_Rprofile.site("suppressMessages(library(installr))") # FALSE
# this would still leave .First

## End(Not run)
```

---

kill_all_Rscript_s *kill (i.e.: stop) all running "Rscript" processes*

Description

kill (i.e.: stop) all running "Rscript" processes based on their process ID (pid)

Usage

```
kill_all_Rscript_s(s = 0, m = 0, h = 0, ...)
```

Arguments

s	numeric. number of seconds to wait before killing the processes
m	numeric. number of minutes to wait before killing the processes
h	numeric. number of hours to wait before killing the processes
...	not used.

Value

an integer vector with the process ID (pid) of the "Rscript" processes.

References

tasklist details from microsoft homepage: https://learn.microsoft.com/en-us/previous-versions/windows/it-pro/windows-xp/bb491010(v=technet.10) pskill details from microsoft homepage: https://learn.microsoft.com/en-us/sysinternals/downloads/pskill

See Also

get_tasklist, get_Rscript_PID, get_pid, kill_pid, kill_all_Rscript_s, pskill

Examples

```
## Not run:
# create several running processes of Rscript (to shitdown)
system("Rscript -e repeat{2+2}", wait = FALSE) # this process should be stuck
system("Rscript -e repeat{2+2}", wait = FALSE) # this process should be stuck
# here are there pid numbers:
get_Rscript_PID()
# let's kill them:
kill_all_Rscript_s()
# they are gone...
get_Rscript_PID() # we no longer have Rscripts running

## End(Not run)
```

---

kill_pid — *kill (i.e.: stop) running processes by there pid*

Description

kill (i.e.: stop) running processes by there pid. It spawns a new Rscript which runs pskill on the pid-s

Usage

```
kill_pid(pid, s = 0, m = 0, h = 0, ...)
```

Arguments

pid	an integer vector with process id numbers (i.e.: can kill several pid at once!)
s	numeric. number of seconds to wait before killing the processes
m	numeric. number of minutes to wait before killing the processes
h	numeric. number of hours to wait before killing the processes
...	not used.

Value

output from system

References

tasklist details from microsoft homepage: https://learn.microsoft.com/en-us/previous-versions/windows/it-pro/windows-xp/bb491010(v=technet.10) pskill details from microsoft homepage: https://learn.microsoft.com/en-us/sysinternals/downloads/pskill

See Also

get_tasklist, get_Rscript_PID, get_pid, kill_pid, kill_all_Rscript_s, pskill

Examples

```
## Not run:
# create several running processes of Rscript (to shitdown)
system("Rscript -e repeat{2+2}", wait = FALSE) # this process should be stuck
system("Rscript -e repeat{2+2}", wait = FALSE) # this process should be stuck
# here are there pid numbers:
get_Rscript_PID()
# let's kill them:
kill_pid(get_Rscript_PID())
# they are gone...
get_Rscript_PID() # we no longer have Rscripts running

## End(Not run)
```

---

kill_process *kill (i.e.: stop) running processes by there process name*

Description

kill (i.e.: stop) running processes by there process name It spawns a new Rscript which runs pskill on the pid-s per process name.

Usage

```
kill_process(process, s = 0, m = 0, h = 0, exact = FALSE, ...)
```

Arguments

process	a character vector of process names.
s	numeric. number of seconds to wait before killing the processes
m	numeric. number of minutes to wait before killing the processes
h	numeric. number of hours to wait before killing the processes
exact	logical (FALSE). should we get exact match to process name, or can we use just partial matching.
...	not used.

Value

output from system

References

tasklist details from microsoft homepage: https://learn.microsoft.com/en-us/previous-versions/windows/it-pro/windows-xp/bb491010(v=technet.10) pskill details from microsoft homepage: https://learn.microsoft.com/en-us/sysinternals/downloads/pskill

See Also

get_tasklist, get_Rscript_PID, get_pid, kill_pid, kill_all_Rscript_s, pskill

Examples

```
## Not run:
# create several running processes of Rscript (to shitdown)
system("Rscript -e repeat{2+2}", wait = FALSE) # this process should be stuck
system("Rscript -e repeat{2+2}", wait = FALSE) # this process should be stuck
# here are there pid numbers:
get_Rscript_PID()
# let's kill them:
kill_process("Rscript")
# they are gone...
get_Rscript_PID() # we no longer have Rscripts running

## End(Not run)
```

`lineplot_package_downloads`
barplot for the number of users installation of a package

Description

This function gets a vector of package names, and returns a line plot of number of downloads for these packages per week.

Usage

```
lineplot_package_downloads(
  pkg_names,
  dataset,
  by_time = c("date", "week"),
  ...
)
```

Arguments

`pkg_names`	a character vector of packages we are interested in checking.
`dataset`	a dataset output from running read_RStudio_CRAN_data, after going through format_RStudio_CRAN_data.

`by_time`	by what time frame should packages be plotted? defaults to "date", but can also be "week"
`...`	not in use.

Details

RStudio maintains its own CRAN mirror, https://cran.rstudio.com/ and offers its log files.

Value

invisible aggregated data that was used for the plot

Author(s)

Felix Schonbrodt, Tal Galili

Source

https://www.nicebread.de/finally-tracking-cran-packages-downloads/

See Also

download_RStudio_CRAN_data, read_RStudio_CRAN_data,barplot_package_users_per_day

Examples

```
## Not run:
# The first two functions might take a good deal of time to run (depending on the date range)
RStudio_CRAN_data_folder <-
      download_RStudio_CRAN_data(START = '2013-04-02',
                                 END = '2013-04-05')
                                 # around the time R 3.0.0 was released
my_RStudio_CRAN_data <- read_RStudio_CRAN_data(RStudio_CRAN_data_folder)
my_RStudio_CRAN_data <- format_RStudio_CRAN_data(my_RStudio_CRAN_data)
head(my_RStudio_CRAN_data)
lineplot_package_downloads(pkg_names = c("ggplot2", "reshape", "plyr", "installr"),
                           dataset = my_RStudio_CRAN_data)

# older plots:
# barplots: (more functions can easily be added in the future)
barplot_package_users_per_day("installr", my_RStudio_CRAN_data)
barplot_package_users_per_day("plyr", my_RStudio_CRAN_data)

## End(Not run)
```

`load_installr_on_startup` *Have the installr package load on startup*

Description

Load installr on startup.

Usage

```
load_installr_on_startup(...)
```

Arguments

`...`	not used. (but good for future backward compatibility)

Value

invisible(NULL)

References

https://stackoverflow.com/questions/1395301/how-to-get-r-to-recognize-your-working-directory-as-it
https://stackoverflow.com/questions/1189759/expert-r-users-whats-in-your-rprofile
http://www.noamross.net/archives/2012-11-02-rprofile/ https://www.statmethods.net/interface/customizing.html

Examples

```
## Not run:
load_installr_on_startup()

## End(Not run)
```

`most_downloaded_packages` *Most downloaded packages*

Description

Gives the top "x" most downloaded packages.

Usage

```
most_downloaded_packages(dataset, n = 6L, ...)
```

Arguments

dataset	a dataset output from running read_RStudio_CRAN_data, after going through format_RStudio_CRAN_data.
n	the number of top packages to show.
...	not in use.

Details

RStudio maintains its own CRAN mirror, https://cran.rstudio.com/ and offers its log files.

Value

a table of top packages by downloads (a numeric vector with packages as names)

Source

https://www.nicebread.de/finally-tracking-cran-packages-downloads/

See Also

download_RStudio_CRAN_data, read_RStudio_CRAN_data,barplot_package_users_per_day

Examples

```
## Not run:
# The first two functions might take a good deal of time to run (depending on the date range)
RStudio_CRAN_data_folder <-
      download_RStudio_CRAN_data(START = '2013-04-02',
                                 END = '2013-04-05')
                                 # around the time R 3.0.0 was released
my_RStudio_CRAN_data <- read_RStudio_CRAN_data(RStudio_CRAN_data_folder)
my_RStudio_CRAN_data <- format_RStudio_CRAN_data(my_RStudio_CRAN_data)
head(my_RStudio_CRAN_data)
most_downloaded_packages(my_RStudio_CRAN_data)

top_packages <- names(most_downloaded_packages(my_RStudio_CRAN_data))
lineplot_package_downloads(pkg_names = top_packages, dataset = my_RStudio_CRAN_data)


## End(Not run)
```

`myip` *What is my IP*

Description

Retrieving your public IP via `https://api.ipify.org`. (old solution used: http://api.exip.org/ based on http://stackoverflow.com/questions/3097589/getting-my-public-ip-via-api)

Usage

```
myip(...)
```

Arguments

`...`	not in use

Value

your current ip (character string)

Source

`https://api.ipify.org`

See Also

freegeoip, myip, cranometer

Examples

```
## Not run:
myip() # "37.132.25.15"

## End(Not run)
```

`os.hibernate` *Hibernate the operating system (Windows) through a shell command*

Description

This Hibernates Windows after set amount of time.

Usage

```
os.hibernate(s = 0, m = 0, h = 0, first_turn_hibernate_on = TRUE)
```

Arguments

`s`	time to wait before shutting down (in seconds), added to m and h; passed to `Sys.sleep`
`m`	time to wait before shutting down (in minutes), added to s and h; passed to `Sys.sleep`
`h`	time to wait before shutting down (in hours), added to s and m; passed to `Sys.sleep`
`first_turn_hibernate_on`	default is TRUE. This runs "powercfg -hibernate on" in order to turn hibernate on, in cases where it was off.

Value

The status code of `shell`.

Author(s)

Tal Galili

References

https://superuser.com/questions/42124/how-can-i-put-the-computer-to-sleep-from-command-prompt-run-
, https://www.howtogeek.com/howto/windows-vista/quick-tip-create-shutdown-restart-lock-icons-in-win

See Also

`system`,`shell`, `Sys.sleep`, `is.windows`, `os.shutdown`, `os.sleep`, `os.hibernate`, `os.lock`, `os.restart`

Examples

```
## Not run:
## when your code is extremely time-consuming,
# you may need this function to run at the
# end of the simulation.
os.hibernate()

## End(Not run)
```

`os.lock` *Locks the operating system (Windows) through a shell command*

Description

This locks Windows after set amount of time.

Usage

```
os.lock(s = 0, m = 0, h = 0)
```

Arguments

`s`	time to wait before shutting down (in seconds), added to m and h; passed to `Sys.sleep`
`m`	time to wait before shutting down (in minutes), added to s and h; passed to `Sys.sleep`
`h`	time to wait before shutting down (in hours), added to s and m; passed to `Sys.sleep`

Value

The status code of `shell`.

Author(s)

Tal Galili

References

https://superuser.com/questions/42124/how-can-i-put-the-computer-to-sleep-from-command-prompt-run-
, https://www.howtogeek.com/howto/windows-vista/quick-tip-create-shutdown-restart-lock-icons-in-win

See Also

`system`,`shell`, `Sys.sleep`, `is.windows`, `os.shutdown`, `os.sleep`, `os.hibernate`, `os.lock`, `os.restart`

Examples

```
## Not run:
## when your code is extremely time-consuming,
# you may need this function to run at the
# end of the simulation.
os.lock()

## End(Not run)
```

`os.manage` *Gives managing option to the current OS (shutdown, restart, sleep, hibernate, etc...)*

Description

A central function to run functions for shutting down, restarting, sleeping (etc.) your computer. This will run these functions immediately.

Usage

```
os.manage(GUI = TRUE, ask = TRUE, ...)
```

Arguments

`GUI`	a logical indicating whether a graphics menu should be used if available. If TRUE, and on Windows, it will use winDialog, otherwise it will use menu.
`ask`	a logical indicating whether to ask the user for the number of minutes in which to perform the operation.
`...`	not in use

Value

The status code of `system`.

References

`https://superuser.com/questions/42124/how-can-i-put-the-computer-to-sleep-from-command-prompt-run-`
, `https://www.howtogeek.com/howto/windows-vista/quick-tip-create-shutdown-restart-lock-icons-in-win`

See Also

`system`,`shell`, `Sys.sleep`, `is.windows`, `os.shutdown`, `os.sleep`, `os.hibernate`, `os.lock`, `os.restart`

Examples

```
## Not run:
## when your code is extremely time-consuming,
# you may need this function;
# e.g. you wish to go to sleep,
# while keeping R running with a long computation...
# complex graphics... and long long computation...
# at last,
os.manage()
## the next day you wake up, "thank you, R" :)

## End(Not run)
```

`os.restart` *Restarts the operating system (Windows) through a shell command*

Description

This restarts Windows after set amount of time.

Usage

```
os.restart(s = 0, m = 0, h = 0)
```

Arguments

`s`	time to wait before shutting down (in seconds), added to m and h; passed to `Sys.sleep`
`m`	time to wait before shutting down (in minutes), added to s and h; passed to `Sys.sleep`
`h`	time to wait before shutting down (in hours), added to s and m; passed to `Sys.sleep`

Value

The status code of `shell`.

Author(s)

Tal Galili

References

`https://superuser.com/questions/42124/how-can-i-put-the-computer-to-sleep-from-command-prompt-run-`
, `https://www.howtogeek.com/howto/windows-vista/quick-tip-create-shutdown-restart-lock-icons-in-win`

See Also

`system`,`shell`, `Sys.sleep`, `is.windows`, `os.shutdown`, `os.sleep`, `os.hibernate`, `os.lock`, `os.restart`

Examples

```
## Not run:
os.restart()

## End(Not run)
```

`os.shutdown` *Shut down the operating system with the command ‘shutdown’*

Description

There is a command `shutdown` in both Windows and Linux, and this function uses it to shut down a computer.

After the time `wait` has passed, R will execute `shutdown -s -t 0` (for Windows) or `shutdown -h now` to shut down the computer.

This function is a modified version of Yihui’s shutdown function from the fun package.

Usage

```
os.shutdown(s = 0, m = 0, h = 0)
```

Arguments

s	time to wait before shutting down (in seconds), added to m and h; passed to Sys.sleep
m	time to wait before shutting down (in minutes), added to s and h; passed to Sys.sleep
h	time to wait before shutting down (in hours), added to s and m; passed to Sys.sleep

Value

The status code of system.

Author(s)

Yihui Xie <https://yihui.org/>, and Tal Galili

References

https://github.com/yihui/fun/blob/master/R/shutdown.R

See Also

system,shell, Sys.sleep, is.windows, os.shutdown, os.sleep, os.hibernate, os.lock, os.restart

Examples

```
## Not run:
## when your code is extremely time-consuming,
# you may need this function;
# e.g. you wish to go to sleep, while keeping R running long computation...

os.shutdown()
## the next day you wake up, "thank you, R" :)

## End(Not run)
```

`os.sleep` *Sleeps the operating system (Windows) through a shell command*

Description

This sleeps Windows after set amount of time.

Usage

```
os.sleep(s = 0, m = 0, h = 0, first_turn_hibernate_off = TRUE)
```

Arguments

s	time to wait before shutting down (in seconds), added to m and h; passed to `Sys.sleep`
m	time to wait before shutting down (in minutes), added to s and h; passed to `Sys.sleep`
h	time to wait before shutting down (in hours), added to s and m; passed to `Sys.sleep`
first_turn_hibernate_off	The command rundll32.exe powrprof.dll,SetSuspendState 0,1,0 for sleep is correct - however, it will hibernate instead of sleep if you don't turn the hibernation off. I'm not sure this is true, but that's what is explained in the linke (see bellow)

Value

The status code of `shell`.

Author(s)

Tal Galili

References

https://superuser.com/questions/42124/how-can-i-put-the-computer-to-sleep-from-command-prompt-run-
, https://www.howtogeek.com/howto/windows-vista/quick-tip-create-shutdown-restart-lock-icons-in-win
https://superuser.com/a/135450/28536

See Also

`system`,`shell`, `Sys.sleep`, `is.windows`, `os.shutdown`, `os.sleep`, `os.hibernate`, `os.lock`, `os.restart`

Examples

```
## Not run:
## when your code is extremely time-consuming,
# you may need this function to run at the end of
# the simulation.
os.sleep()

## End(Not run)
```

`package_authors` *Access (and clean) author elements from R's Rd file*

Description

Find authors.

Usage

```
package_authors(
  package,
  to_strsplit = TRUE,
  split = c(",|and"),
  to_table = FALSE,
  ...
)
```

Arguments

`package`	a character string of the package we are interested in.
`to_strsplit`	logical (TRUE). Should the authors strings be split (in cases of a "and" or a comma ",")?
`split`	a character scalar to be passed to strsplit split parameter. default is c(",|and)
`to_table`	logical (FALSE). Should the authors strings be listed in a table - showing a count of how many .Rd files they were listed in? If not - a unique list is produced.
`...`	not used.

Details

List authors for a package from its "author" tag elements from its Rd files. The function also separate lists of authors, and cleans the output a bit (from spaces at the beginning of the strings).

Value

a character vector with a package authors (as extracted from the author tag in the .Rd files)

References

Useful for updating your DESCRIPTION file:

`https://cran.r-project.org/doc/manuals/R-exts.html#The-DESCRIPTION-file`

See Also

fetch_tag_from_Rd

Examples

```
## Not run:

# before:
fetch_tag_from_Rd("installr", "\\author")
# after:
package_authors("installr")
sort(package_authors("installr")) # sorted name list...


## From the top R packages list:
## https://www.r-statistics.com/2013/06/top-100-r-packages-for-2013-jan-may/
package_authors("plyr")
package_authors("digest")
package_authors("ggplot2")
package_authors("colorspace")
package_authors("stringr") # empty string.

package_authors("knitr")
package_authors("MASS")
package_authors("rpart")
package_authors("Rcpp")



## End(Not run)
```

`pkgDNLs_worldmapcolor` *Worldmap colored by the number of downloads for a given package*

Description

Plots a worldmap colored by the number of users installation for a given package

Usage

```
pkgDNLs_worldmapcolor(pkg_name, dataset, remove_dups = TRUE, ...)
```

Arguments

`pkg_name`	a character string of the package we are interested in.
`dataset`	a dataset output from running read_RStudio_CRAN_data.
`remove_dups`	logical (default is TRUE). Should the duplicate user ids (based on their ips) be removed.
`...`	not in use.

Details

RStudio maintains its own CRAN mirror, https://cran.rstudio.com/ and offers its log files.

Value

a ggplot object

Author(s)

Boris Hejblum

Source

`https://www.nicebread.de/finally-tracking-cran-packages-downloads/`

See Also

download_RStudio_CRAN_data, read_RStudio_CRAN_data, barplot_package_users_per_day, ggplot

Examples

```
## Not run:
# The first two functions might take a good deal of time to run (depending on the date range)
RStudio_CRAN_data_folder <-
      download_RStudio_CRAN_data(START = '2013-04-02',
                                 END = '2013-04-05')
                                 # around the time R 3.0.0 was released
my_RStudio_CRAN_data <- read_RStudio_CRAN_data(RStudio_CRAN_data_folder)
head(my_RStudio_CRAN_data)

wm <- pkgDNLs_worldmapcolor(pkg_name="installr", dataset = my_RStudio_CRAN_data)
wm


## End(Not run)
```

---

read_RStudio_CRAN_data *Reads RStudio CRAN mirror data files from a folder*

Description

This function reads files downloaded from the download page (`http://cran-logs.rstudio.com/`).

This function relies on data.table to run faster. WARNING: this function can be quite slow...

Usage

```
read_RStudio_CRAN_data(
  log_folder = tempdir(),
  use_data_table = TRUE,
  packages,
  ...
)
```

Arguments

`log_folder`	the folder which contains the RStudio CRAN log files that were downloaded to. Default is the temporary folder picked by tempdir.
`use_data_table`	default is TRUE. A switch for wether or not to use the data.table package in order to merge the log files using rbindlist. This function is MUCH faster then the alternative.
`packages`	a character vector containing the names of packages for which information is extracted. If not specified, all packages are included, but this can cause out-of-memory problems if there are many log files.
`...`	not in use.

Details

RStudio maintains its own CRAN mirror, https://cran.rstudio.com/ and offers its log files.

Value

Returns the combined data file.

Author(s)

Felix Schonbrodt, Tal Galili

Source

`https://www.nicebread.de/finally-tracking-cran-packages-downloads/`

See Also

download_RStudio_CRAN_data, read_RStudio_CRAN_data,barplot_package_users_per_day

Examples

```
## Not run:
# The first two functions might take a good deal of time to run (depending on the date range)
RStudio_CRAN_data_folder <-
      download_RStudio_CRAN_data(START = '2013-04-02',
                                 END = '2013-04-05')
                                 # around the time R 3.0.0 was released
my_RStudio_CRAN_data <- read_RStudio_CRAN_data(RStudio_CRAN_data_folder)
```

```
# barplots: (more functions can easily be added in the future)
barplot_package_users_per_day("installr", my_RStudio_CRAN_data)
barplot_package_users_per_day("plyr", my_RStudio_CRAN_data)

## End(Not run)
```

`remove.installr.GUI` *Removes the menu based GUI for updating R within Rgui*

Description

Removes the menu based GUI for updating R within Rgui.

Usage

```
remove.installr.GUI()
```

Details

This function is used during .Last.lib to remove the menus for the installr package in Rgui.

Value

invisible(NULL)

Examples

```
## Not run:
add.installr.GUI()  # add menus
remove.installr.GUI() # remove them

## End(Not run)
```

`remove_from_.First_in_Rprofile.site` *Remove a code line from Rprofile.site .First*

Description

Goes through Rprofile.site text, finds a line of code - and removes it.

Usage

```
remove_from_.First_in_Rprofile.site(code, fixed = TRUE, ...)
```

Arguments

`code`	A character scalar with code to add at the beginning of the .First function in Rprofile.site
`fixed`	passed to grep
`...`	passed to grep

Value

logical. Did we remove that line or not (in case it was not there)

References

https://stackoverflow.com/questions/1395301/how-to-get-r-to-recognize-your-working-directory-as-it
https://stackoverflow.com/questions/1189759/expert-r-users-whats-in-your-rprofile
http://www.noamross.net/archives/2012-11-02-rprofile/ https://www.statmethods.net/interface/customizing.html

Examples

```
## Not run:
is_in_.First_in_Rprofile.site("suppressMessages(library(installr))") # FALSE
add_to_.First_in_Rprofile.site("suppressMessages(library(installr))")
is_in_.First_in_Rprofile.site("suppressMessages(library(installr))") # TRUE
remove_from_.First_in_Rprofile.site("suppressMessages(library(installr))")
is_in_.First_in_Rprofile.site("suppressMessages(library(installr))") # FALSE
# this would still leave .First

## End(Not run)
```

`rename_r_to_R` *Rename files' extensions in a folder from .r to .R*

Description

Rename files' extensions in a folder from .r to .R.

Usage

```
rename_r_to_R(
  subdir = ".",
  recursive = FALSE,
  message = TRUE,
  text_to_find = "\\.r$",
  new_extension = ".R",
  ...
)
```

Arguments

`subdir`	(character) sub folder from the current working directory in which the files should be changed. Default is "".
`recursive`	(logical) FALSE. Should the function keep going into folders and check them as well?
`message`	(logical) should we output how many files were changed. (default is FALSE)
`text_to_find`	old file extension (should have $ at the end!)
`new_extension`	new file extension...
`...`	not used.

Details

This came after a discussion with Hadley, JJ, and Martin leading to the realization that since we are using the R language (and not the r language), the standard is to use .R files instead of .r

Be careful when using the recursive argument. And remember that source("miao.r") and source("miao.R") Are NOT the same...

Value

(integer) the number of files changed

Examples

```
## Not run:
rename_r_to_R() # changes only .r in the current wd
rename_r_to_R("R") # fixing the file ending inside a package directory
rename_r_to_R(recursive = TRUE) # Changes
rename_r_to_R(recursive = TRUE, message = FALSE) # Changes
# ALL of the .r files underneath the current
# working directory

# source: http://stackoverflow.com/questions/52950/how-to-make-git-ignore-changes-in-case
# First run the following in git bash:
# git config core.ignorecase false
rename_r_to_R(recursive = TRUE, text_to_find="\\.R$", new_extension = ".b")

# mmm, since it does not work nicely, you'd need to run the following:
# and commit between the two.
rename_r_to_R(recursive = TRUE, text_to_find="\\.r$", new_extension = ".b")
# commit!
rename_r_to_R(recursive = TRUE, text_to_find="\\.b$", new_extension = ".R")


## End(Not run)
```

require2 — *Loading Packages (and Installing them if they are missing)*

Description

require2 load add-on packages by passing it to require. However, if the package is not available on the system, it will first install it (through install.packages), and only then try to load it again.

Usage

```
require2(package, ask = FALSE, character.only = FALSE, min_version = 0)
```

Arguments

`package`	A character of the name of a package (can also be without quotes).
`ask`	Should the user be asked to install the require packaged, in case it is missing? (default is FALSE)
`character.only`	logical (FALSE) - a logical indicating whether package or help can be assumed to be character strings. Passed to require.
`min_version`	Minimum version of package

Value

returns (invisibly) a logical indicating whether the required package is available.

Examples

```
## Not run:
require2("devtools")
require2(geonames)
require2(pkgbuild, min_version = "1.1.0")
data_table_loaded <- require2("data.table")

## End(Not run)
```

restart_RGui — *Restart RGui from RGui*

Description

Start a new RGui session and then quits the current one.

This is a Windows only function.

Usage

```
restart_RGui(...)
```

Arguments

`...`	passed to q()

Value

q(...)

Examples

```
## Not run:
restart_RGui()

## End(Not run)
```

---

rm_installr_from_startup

Remove installr from startup

Description

Have the installr package NOT load on startup

Usage

```
rm_installr_from_startup(...)
```

Arguments

`...`	not used. (but good for future backward compatibility)

Value

invisible(NULL)

References

https://stackoverflow.com/questions/1395301/how-to-get-r-to-recognize-your-working-directory-as-it
https://stackoverflow.com/questions/1189759/expert-r-users-whats-in-your-rprofile
http://www.noamross.net/archives/2012-11-02-rprofile/ https://www.statmethods.net/interface/customizing.html

Examples

```
## Not run:
load_installr_on_startup()
rm_installr_from_startup()

## End(Not run)
```

`R_version_in_a_folder` *Get the version of the R installed in a folder*

Description

Get the version of the R installed in a folder based on the structure of the filename README.R-... (where ... is a version number for R). This function helps detect the version number of an R installation even if the name of the folder is not standard. If multiple versions were installed, overwriting each other, the most recent is selected.

Usage

```
R_version_in_a_folder(folder)
```

Arguments

`folder`	The folder for which we wish to know the R version.

Value

Returns a character vector of the R version (or NA, if this is not an R installation folder)

See Also

get.installed.R.folders

Examples

```
## Not run:
R_version_in_a_folder(folder = R.home())
# returns the version of the current R installation

## End(Not run)
```

`source.https` *Read R Code from a File in an https URL*

Description

source.https causes R to accept its input from a File in an https URL. Input is read and parsed from that file until the end of the file is reached, then the parsed expressions are evaluated sequentially in the chosen environment.

Usage

```
source.https(URL, ..., remove_r_file = T)
```

Arguments

`URL`	the URL of the .r file to download and source.
`...`	parameters to pass to source
`remove_r_file`	if to remove the .r file after it was sourced.

Details

"The easiest way to use Git on Windows." (at least so they say...)

Value

Nothing.

References

Other solutions to the source.https problem:

- Using RCurl
- devtools::source_url
- A relevant (OLD) discussion: https://stackoverflow.com/questions/7715723/sourcing-r-script-over-https

See Also

source

Examples

```
## Not run:
source.https("https://raw.github.com/talgalili/installr/master/R/install.r")

## End(Not run)
```

`system.PATH` *Returns the search path for executable files*

Description

Returns the search path for executable files based on

Usage

```
system.PATH()
```

Value

A character vector with the search path for executable files

References

http://www.microsoft.com/resources/documentation/windows/xp/all/proddocs/en-us/ntcmds_shelloverview.mspx?mfr=true

Examples

```
## Not run:
system.PATH() #

## End(Not run)
```

`turn.number.version` *Turns a vector of version-numbers back to version-character*

Description

Version Num to char

Usage

```
turn.number.version(number_to_dots)
```

Arguments

`number_to_dots`	A numeric vector - of the number-version of R

Value

A vector of "numbers" representing the versions (for example: 2015002). The names of the vector is the original version character.

Examples

```
## Not run:
turn.number.version(turn.version.to.number(c("2.15.2", "2.15.2")))
turn.number.version(2015011) # "2.15.11"

## End(Not run)
```

`turn.version.to.number` *Turns version to number (for a vector of values)*

Description

Turns version to number (for a vector of values)

Usage

```
turn.version.to.number(version_with_dots)
```

Arguments

`version_with_dots`	- A character vector - of the version of R (for example 2.15.2)

Value

A vector of "numbers" representing the versions (for example: 2015002). The names of the vector is the original version character.

Examples

```
## Not run:
turn.version.to.number(c("2.15.2", "2.15.2"))

## End(Not run)
```

`turn.version.to.number1` *Turns version to number (for 1 value only)*

Description

Turns version to number (for 1 value only)

Usage

```
turn.version.to.number1(version_with_dots)
```

Arguments

`version_with_dots`	A character value - of the version of R (for example 2.15.2)

Value

A "number" representation of the version (for example: 2015002)

See Also

turn.version.to.number

Examples

```
## Not run:
turn.version.to.number1("2.15.2")
turn.version.to.number1("3.0.1")

## End(Not run)
```

`uninstall.packages` *uninstalls (removes) Installed Packages*

Description

A wrapper for remove.packages. Useful since it also works if the package is currently loaded into the workspace.

Usage

```
uninstall.packages(pkgs, lib, warning = TRUE, ...)
```

Arguments

`pkgs`	a character vector with the names of the packages to be removed.
`lib`	a character vector giving the library directories to remove the packages from. If missing, defaults to the first element in .libPaths.
`warning`	boolean (TRUE), should a message be printed in various cases.
`...`	currently ignored.

Value

Invisible NULL

See Also

`install.packages`, `remove.packages`, `install.packages.zip`

Examples

```
## Not run:
install.packages(c("reshape", "plyr"))
require(plyr)
uninstall.packages(c("reshape", "plyr"))
install.packages(c("reshape", "plyr"))

## End(Not run)
```

`uninstall.R` *Uninstall an R version*

Description

Choose an R version to uninstall via a menubar. By default, the function allows the user to pick an R version to uninstall from a list. Also, the function can be called with using "r_version", where multiple R versions can be supplied and all will be uninstalled.

Usage

```
uninstall.R(r_version, GUI = TRUE)
```

Arguments

`r_version`	a character vector for R versions to uninstall (the format is of the style: "2.15.3"). default is empty - resulting in a prompt message asking the user what to do.
`GUI`	If asking the user which R version to uninstall, should the GUI be used? (default is TRUE)

Value

the output of system running the uninstaller

See Also

install.R, updateR, system

Examples

```
## Not run:
uninstall.R() # choose an R version to uninstall
uninstall.R("2.15.3") # will uninstall R 2.15.3
uninstall.R(c("2.15.3", "2.14.0")) # will uninstall two R versions (if both exists)
uninstall.R("10.10.0") # would pop up the menu options (until R 10.10.0 will be released :D )

## End(Not run)
```

`updateR` *Checks for the latest R version, and if there is a newer version of R - downloads and installs it.*

Description

This function performs the following steps:

- Check what is the latest R version. If the current installed R version is up-to-date, the function ends (and returns FALSE)
- If a newer version of R is available, the user is asked if to review the NEWS of the latest R version - in order to decide if to install the newest R or not.
- If the user wishes to - the function will download and install it. (you will need to press the "next" buttons on your own)
- Once the installation is done, you should press "any-key", and the function will proceed with copying all of your packages from your old (well, current) R installation, into your newer R installation.
- You can then erase all of the packages in your old R installation.
- After your packages are moved (and the old ones possibly erased), you will get the option to update all of your packages in the new version of R.
- You will be asked if to open the Rgui of your new R.
- Lastly - you can close the current session of your old R.

Usage

```
updateR(
  fast = FALSE,
  browse_news,
  install_R,
  copy_packages,
  copy_site_files,
  keep_old_packages,
  update_packages,
  start_new_R,
  quit_R,
  print_R_versions = TRUE,
  GUI = TRUE,
  to_checkMD5sums = FALSE,
  keep_install_file = FALSE,
  download_dir = tempdir(),
  silent = FALSE,
  setInternet2 = TRUE,
  cran_mirror = "https://cran.rstudio.com/",
  ...
)
```

Arguments

`fast`	logical (default is FALSE). If TRUE, it overrides other parameters and uses a set of defaults to make the R installation as fast as possible: no news, installr R, copy packages and Rprofile, keep old packages, updated packages, without quitting current R or starting the new R. don't use GUI, check MD5sums, keep installed file in the getwd.
`browse_news`	if TRUE (and if there is a newer version of R) - it opens the browser to the NEWS of the latest version of R, for the user to read through
`install_R`	TRUE/FALSE - if to install a new version of R (if one is available). If missing (this is the default) - the user be asked if to download R or not.Of course the installation part itself (the running of the .exe file) is dependent on the user.
`copy_packages`	TRUE/FALSE - if to copy your packages from the old version of R to the new version of R. If missing (this is the default) - the user will be asked for their preference (he should say yes, unless they are using a global library folder).
`copy_site_files`	logical - if to copy your Rprofile.site and Renviron.site from the old version of R to the new version of R. If missing (this is the default) - the user will be asked for their preference (they should say yes, unless they are using a global library folder).
`keep_old_packages`	- if the keep the packages in the library of the old R installation. If missing (this is the default) - the user will be asked for their preference (they should say yes, unless they are using a global library folder).
`update_packages`	TRUE/FALSE - if to update your packages in the new version of R (all packages will be updated without asking confirmation per package) If missing (this is the default) - the user will be asked for their preference (they should say yes, unless they are using a global library folder). This is done by calling the Rscript in the new R.
`start_new_R`	TRUE/FALSE - if to start the new R (Rgui) after we will quit the old R. Default is TRUE. It will try to start the 64 bit R version, if it does not exist, the 32 bit will be started. This may be less useful for people using RStudio or the likes.
`quit_R`	TRUE/FALSE - if to quit R after the installation and package copying or not. If missing (this is the default) - the user is asked what to do.
`print_R_versions`	if to tell the user what version they have and what is the latest version (default is TRUE)
`GUI`	a logical indicating whether a graphics menu should be used if available. If TRUE, and on Windows, it will use winDialog, otherwise it will use menu.
`to_checkMD5sums`	Should we check that the new R installation has the files we expect it to (by checking the MD5 sums)? default is TRUE. It assumes that the R which was installed is the latest R version. parameter is passed to install.R()
`keep_install_file`	If TRUE - the installer file will not be erased after it is downloaded and run.

`download_dir`	A character of the directory into which to download the file. (default is tempdir())
`silent`	If TRUE - enables silent installation mode.
`setInternet2`	logical. Should setInternet2(TRUE) be run. (only relevant for versions of R before 3.3.0)
`cran_mirror`	URL of your preferred CRAN mirror. (default is https://cran.rstudio.com/)
`...`	Other arguments (this is currently not used in any way)

Details

It is worth noting that the function assumes that you are installing R in the same directory as before. That is, if the old R was on: D:RR-3.0.0 then the new R will be on D:RR-3.0.1.

Value

a TRUE/FALSE value on whether or not R was updated.

See Also

check.for.updates.R, install.R, copy.packages.between.libraries, uninstall.R

Examples

```
## Not run:

updateR(TRUE) # This sets "fast" to be TRUE
# # the fastest/safest upgrade option:
# install R while keeping a copy in the working directory,
# copy packages, keep old packages,
# update packages in the new installation.

updateR() # will ask you what you want at every decision.

## End(Not run)
```

`up_folder` *Performs "up-level" on a folder string*

Description

Gets a character vector of folder strings and returns the same vector after removing the end of the folder path.

Usage

```
up_folder(FOLDER, n = -1, ...)
```

Arguments

`FOLDER`	a character vector of folders
`n`	passed to n in function head
`...`	not used.

Value

The name of the file in the URL

Examples

```
up_folder(FOLDER = c("D:/R/R-3.0.1", "D:/R/R-3.0.2", "D:/R/R-3.0.3"))
```

---

xlsx2csv *Converts xls(x) to csv using VB*

Description

Converts xls(x) to csv using VB script. Not that important now that we have the readxl package.

Usage

```
xlsx2csv(xlsx, csv, path, ...)
```

Arguments

`xlsx`	the (character) name of the xlsx (or xls) file to convert. if xlsx has a full path, it will override the path parameter.
`csv`	the (character) name of the csv file to convert to (default will be the name of the xlsx file)
`path`	the path for the files (default is the working directory).
`...`	ignored.

Source

This is based on the code from plang's answer here: https://stackoverflow.com/questions/1858195/convert-xls-to-csv-on-command-line

Examples

```
## Not run:

xlsx2csv("c:/some_file.xlsx")


## End(Not run)
```

Index