Package ‘inpdfr’

August 24, 2023

Type Package

Title Analyse Text Documents Using Ecological Tools

Version 0.1.12

Author Rebaudo Francois (IRD, UMR EGCE, IRD, CNRS, Univ. ParisSaclay)

Maintainer Rebaudo Francois <francois.rebaudo@ird.fr>

Description A set of functions to analyse and compare texts, using classical text mining functions, as well as those from theoretical ecology.

License GPL-2

LazyData TRUE

Depends R (>= 3.5.0)

Imports wordcloud (>= 2.5), RColorBrewer (>= 1.1-2), tm (>= 0.6-2), SnowballC (>= 0.5.1), cluster (>= 2.0.1), entropart (>= 1.4.1), metacom (>= 1.4.4), parallel (>= 3.1.3), stringi (>= 1.0-1), R.devices (>= 2.14.0)

SystemRequirements XPDF (http://www.foolabs.com/xpdf/download.html)

Suggests knitr, rmarkdown, testthat

VignetteBuilder knitr

RoxygenNote 7.2.3

URL https://github.com/frareb/inpdfr/

BugReports https://github.com/frareb/inpdfr/issues

NeedsCompilation no

Repository CRAN

Date/Publication 2023-08-24 07:50:02 UTC

R topics documented:

`doCluster` *Performs a cluster analysis on the basis of the word-occurrence data.frame.*

Description

Performs a cluster analysis on the basis of the word-occurrence data.frame using `hclust` function.

Usage

```
doCluster(
  wordF,
  myMethod = "ward.D2",
  gp = FALSE,
  nbGp = 5,
  getPlot = TRUE,
  mwidth = 800,
  mheight = 800,
  formatType = "png",
  ...
)
```

Arguments

`wordF`	The data.frame containing word occurrences.
`myMethod`	The method to compute distances, see `dist` function.
`gp`	A logical to specify if groups should be made.
`nbGp`	An intger to specify the number of groups. Ignored if `gp=FALSE`.
`getPlot`	If `TRUE`, save the cluster plot in the RESULTS directory.
`mwidth`	The width of the plot in pixels.
`mheight`	The height of the plot in pixels.
`formatType`	The format for the output file ("eps", "pdf", "png", "svg", "tiff", "jpeg", "bmp").
`...`	Additional arguments from the `hclust` function.

Value

An object of class `hclust`.

Examples

```
data("wordOccuDF")
doCluster(wordF = wordOccuDF, myMethod = "ward.D2", getPlot = FALSE)
```

`doKmeansClust` *Performs a k-means cluster analysis on the basis of the word-occurrence data.frame.*

Description

Performs a k-means cluster analysis on the basis of the word-occurrence data.frame using `kmeans` function.

Usage

```
doKmeansClust(
  wordF,
  nbClust = 4,
  nbIter = 10,
  algo = "Hartigan-Wong",
  getPlot = TRUE,
  mwidth = 800,
  mheight = 800,
  formatType = "png",
  ...
)
```

Arguments

`wordF`	The data.frame containing word occurrences.
`nbClust`	The number of clusters.
`nbIter`	The number of iterations allowed.
`algo`	The algoritm used (see `kmeans`).
`getPlot`	If `TRUE`, save the k-means cluster plot in the RESULTS directory.
`mwidth`	The width of the plot in pixels.
`mheight`	The height of the plot in pixels.
`formatType`	The format for the output file ("eps", "pdf", "png", "svg", "tiff", "jpeg", "bmp").
`...`	Additional arguments from the `kmeans` function.

Value

An object of class kmeans (see `kmeans`).

Examples

```
data("wordOccuDF")
doKmeansClust(wordF = wordOccuDF, nbClust = 2, getPlot = FALSE)
```

`doMetacomEntropart` *Performs an analysis of ecological diversity and structure.*

Description

Uses the `entropart-package` to analyse the word-occurrence data.frame, considering words as species and documents as communities.

Usage

```
doMetacomEntropart(
  wordF,
  getPlot = c(TRUE, TRUE, TRUE, TRUE),
  getTextSink = c(TRUE, TRUE, TRUE, TRUE),
  mwidth = 800,
  mheight = 800,
  formatType = "png"
)
```

Arguments

`wordF`	The data.frame containing word occurrences.
`getPlot`	A vector with four logical values. If `getPlot[1]==TRUE`, the `MetaCommunity` object is plotted and saved in the RESULTS directory. If `getPlot[2]==TRUE`, the `DivPart` analisis is plotted and saved in the RESULTS directory. If `getPlot[3]==TRUE`, the `DivEst` analisis is plotted and saved in the RESULTS directory. If `getPlot[4]==TRUE`, the `DivProfile` analisis is plotted and saved in the RESULTS directory.
`getTextSink`	A vector with four logical values. If `getTextSink[1]==TRUE`, the `MetaCommunity` object is saved in the RESULTS directory. If `getTextSink[2]==TRUE`, the `DivPart` analisis is saved in the RESULTS directory. If `getTextSink[3]==TRUE`, the `DivEst` analisis is saved in the RESULTS directory. If `getTextSink[4]==TRUE`, the `DivProfile` analisis is saved in the RESULTS directory.
`mwidth`	The width of the plot in pixels.
`mheight`	The height of the plot in pixels.
`formatType`	The format for the output file ("eps", "pdf", "png", "svg", "tiff", "jpeg", "bmp").

Value

A `MetaCommunity` object (see `entropart-package`).

Examples

```
## Not run:
data("wordOccuDF")
doMetacomEntropart(wordF = wordOccuDF)

## End(Not run)
```

`doMetacomMetacom` *Performs a metacomunity analysis.*

Description

Use the package `Metacommunity` to analyse the word-occurrence data.frame, considering words as species and documents as communities.

Usage

```
doMetacomMetacom(
  wordF,
  numSim = 10,
  limit = "Inf",
  getPlot = TRUE,
  getTextSink = TRUE,
  mwidth = 800,
  mheight = 800,
  formatType = "png"
)
```

Arguments

`wordF`	The data.frame containing word occurrences.
`numSim`	Number of simulated null matrices, see `Metacommunity`.
`limit`	An integer to limit the number of words to use in the analysis.
`getPlot`	If `TRUE`, save the plot in the RESULTS directory.
`getTextSink`	If `TRUE`, save the console output in the RESULTS directory.
`mwidth`	The width of the plot in pixels.
`mheight`	The height of the plot in pixels.
`formatType`	The format for the output file ("eps", "pdf", "png", "svg", "tiff", "jpeg", "bmp").

Value

An object of class `Metacommunity`.

Examples

```
data("wordOccuDF")
doMetacomMetacom(wordF = wordOccuDF, getPlot = FALSE, getTextSink = FALSE)
```

`excludeStopWords` *Exclude StopWords form the word-occurrence data.frame.*

Description

Exclude StopWords form the word occurrences data.frame. `excludeStopWords` uses `parallel` to perform parallel computation.

Usage

```
excludeStopWords(wordF, lang = "English")
```

Arguments

`wordF`	The data.frame containing word occurrences.
`lang`	The language used ("French", "English", "Spanish").

Value

The word-occurrence data.frame.

Examples

```
## Not run:
excludeStopWords(wordF = myDF, lang = "French")

## End(Not run)
```

exclusionList_FR *Stop words in French.*

Description

A vector containing stop words in French.

Usage

```
exclusionList_FR
```

Format

A vector with 173 elements (character), with UTF-8 characters escaped using `stringi::stri_escape_unicode(exclusion`

Source

Adapted from www.ranks.nl/stopwords.

exclusionList_SP *Stop words in Spanish.*

Description

A vector containing stop words in Spanish

Usage

```
exclusionList_SP
```

Format

A vector with 190 elements (character), with UTF-8 characters escaped using `stringi::stri_escape_unicode(exclusion`

Source

Adapted from www.ranks.nl/stopwords.

`exclusionList_UK`	*Stop words in English.*

Description

A vector containing stop words in English.

Usage

```
exclusionList_UK
```

Format

A vector with 542 elements (character).

Source

Adapted from www.ranks.nl/stopwords.

`getAllAnalysis`	*A quick way to compute a set of analysis from the word-occurrence data.frame.*

Description

A quick way to compute a set of analysis from the word-occurrence data.frame.

Usage

```
getAllAnalysis(
  dataset,
  wcloud = TRUE,
  sumStats = TRUE,
  freqW = TRUE,
  clust = TRUE,
  metacom = TRUE
)
```

Arguments

`dataset`	A single word-occurrrence data.frame.
`wcloud`	A logical to for word cloud analysis.
`sumStats`	A logical to for summary statistics analysis.
`freqW`	A logical to for word frequency analysis.
`clust`	A logical to for cluster analysis.
`metacom`	A logical to for metacommunity analysis.

Value

A set of analyses available from the `inpdfr` package.

Examples

```
## Not run:
data("wordOccuDF")
getAllAnalysis(dataset = wordOccuDF, wcloud = FALSE, sumStats = FALSE)

## End(Not run)
```

`getListFiles` *List files in a specified directory sorted by extension.*

Description

List files in a specified directory sorted by extension. The function takes into account .txt and .pdf files based on `strsplit` function.

Usage

```
getListFiles(mywd)
```

Arguments

`mywd`	A string containing the working directory.

Value

A list of length 2 with file names sorted by extension (pdf and txt).

Examples

```
getListFiles(mywd = getwd())
```

`getMostFreqWord` *Returns most frequent words.*

Description

Returns most frequent words and plots their frequencies per document.

Usage

```
getMostFreqWord(
  wordF,
  numWords,
  getPlot = TRUE,
  mwidth = 1024,
  mheight = 800,
  formatType = "png"
)
```

Arguments

`wordF`	The data.frame containing word occurrences.
`numWords`	The number of words to be returned.
`getPlot`	If `TRUE`, save a scatter plot in the RESULTS directory.
`mwidth`	The width of the plot in pixels.
`mheight`	The height of the plot in pixels.
`formatType`	The format for the output file ("eps", "pdf", "png", "svg", "tiff", "jpeg", "bmp").

Value

The `numWords` most frequent words.

Examples

```
data("wordOccuDF")
getMostFreqWord(wordF = wordOccuDF, numWords = 5, getPlot = FALSE)
```

`getMostFreqWordCor` *Test for correlation between the most frequent words.*

Description

Test for correlation between the most frequent words.

Usage

```
getMostFreqWordCor(
  wordF,
  numWords,
  getPlot = c(TRUE, TRUE),
  getTextSink = TRUE,
  mwidth = 1024,
  mheight = 1024,
  formatType = "png"
)
```

Arguments

`wordF`	The data.frame containing word occurrences.
`numWords`	The number of words to be returned.
`getPlot`	A vector with two logical values. If `plots[1]==TRUE`, an image of the correlation matrix is saved in the RESULTS directory. If `plots[2]==TRUE`, the image of the p-value matrix associated with the correlation is saved in the RESULTS directory.
`getTextSink`	If `TRUE`, save the correlation matrix and the associated p-values in a text file in the RESULTS directory.
`mwidth`	The width of the plot in pixels.
`mheight`	The height of the plot in pixels.
`formatType`	The format for the output file ("eps", "pdf", "png", "svg", "tiff", "jpeg", "bmp").

Value

A list with the correlation matrix and the p-value matrix.

Examples

```
data("wordOccuDF")
getMostFreqWordCor(
  wordF = wordOccuDF,
  numWords = 5,
  getPlot = c(FALSE, FALSE),
  getTextSink = FALSE)
```

getPDF — *Extract text from PDF files and return a word-occurrence data.frame.*

Description

getPDF returns a word-occurrence data.frame from PDF files. It needs XPDF in order to run (http://www.foolabs.com/xpdf/dov
and uses parallel to perform parallel computation.

Usage

```
getPDF(
  myPDFs,
  minword = 1,
  maxword = 20,
  minFreqWord = 1,
  pathToPdftotext = ""
)
```

Arguments

myPDFs	A character vector containing PDF file names.
minword	An integer specifying the minimum number of letters per word into the returned data.frame.
maxword	An integer to specifying the maximum number of letters per word into the returned data.frame.
minFreqWord	An integer specifying the minimum word frequency into the returned data.frame.
pathToPdftotext	A character containing an alternative path to XPDF pdftotext function, see Details section.

Details

getPDF uses XPDF pdftotext function to extract the content of PDF files into a TXT file. If pdftotext is not in the PATH, an alternative is to provide the full path of the program into the pathToPdftotext parameter.

Value

A list of list with word-occurrence data.frame and file name.

Examples

```
## Not run:
getPDF(myPDFs = "mypdf.pdf")

## End(Not run)
```

getStopWords *Load a list of stopwords.*

Description

getStopWords returns a list of stopwords.

Usage

```
getStopWords()
```

Value

A list of vectors with stopwords for French, English, and Spanish languages.

Examples

```
getStopWords()
```

getSummaryStatsBARPLOT *Perform a barplot with the number of unique words per document*

Description

Perform a barplot with the number of unique words per document using barplot function.

Usage

```
getSummaryStatsBARPLOT(
  wordF,
  getPlot = TRUE,
  mwidth = 480,
  mheight = 480,
  formatType = "png",
  ...
)
```

Arguments

wordF	The data.frame containing word occurrences.
getPlot	If TRUE, save the bar plot in the RESULTS directory.
mwidth	The width of the plot in pixels.
mheight	The height of the plot in pixels.
formatType	The format for the output file ("eps", "pdf", "png", "svg", "tiff", "jpeg", "bmp").
...	Additional arguments from barplot function.

Value

The number of unique words per document.

Examples

```
data("wordOccuDF")
getSummaryStatsBARPLOT(wordF = wordOccuDF, getPlot = FALSE)
```

---

getSummaryStatsHISTO *Plot an histogram with the number of words excluding stop words*

Description

Plot a histogram with the number of words excluding stop words using `hist` function.

Usage

```
getSummaryStatsHISTO(
  wordF,
  getPlot = TRUE,
  mwidth = 800,
  mheight = 800,
  formatType = "png",
  ...
)
```

Arguments

`wordF`	The data.frame containing word occurrences.
`getPlot`	If `TRUE`, save the plot in the RESULTS directory.
`mwidth`	The width of the plot in pixels.
`mheight`	The height of the plot in pixels.
`formatType`	The format for the output file ("eps", "pdf", "png", "svg", "tiff", "jpeg", "bmp").
`...`	Additional arguments from `hist` function.

Examples

```
data("wordOccuDF")
getSummaryStatsHISTO(wordF = wordOccuDF, getPlot = FALSE)
```

getSummaryStatsOCCUR — *Plot a scatter plot with the proportion of documents using similar words.*

Description

Plot a scatter plot with the proportion of documents using similar words.

Usage

```
getSummaryStatsOCCUR(
  wordF,
  getPlot = TRUE,
  mwidth = 800,
  mheight = 800,
  formatType = "png"
)
```

Arguments

`wordF`	The data.frame containing word occurrences.
`getPlot`	If `TRUE`, save the scatter plot in the RESULTS directory.
`mwidth`	The width of the plot in pixels.
`mheight`	The height of the plot in pixels.
`formatType`	The format for the output file ("eps", "pdf", "png", "svg", "tiff", "jpeg", "bmp").

Value

A data.frame containing the proportion of documents and the number of similar words.

Examples

```
## Not run:
getSummaryStatsOCCUR(wordF = myDF)

## End(Not run)
```

`getTXT` *Extract text from TXT files and return a word-occurrence data.frame.*

Description

Extract text from TXT files and return a word-occurrence data.frame.

Usage

```
getTXT(myTXTs)
```

Arguments

`myTXTs`	A character vector containing TXT file names (or complete path to these files).

Value

A list of list with word-occurrence data.frame and file name.

Examples

```
## Not run:
data("loremIpsum")
loremIpsum01 <- loremIpsum[1:100]
loremIpsum02 <- loremIpsum[101:200]
loremIpsum03 <- loremIpsum[201:300]
loremIpsum04 <- loremIpsum[301:400]
loremIpsum05 <- loremIpsum[401:500]
subDir <- "RESULTS"
dir.create(file.path(getwd(), subDir), showWarnings = FALSE)
write(x = loremIpsum01, file = "RESULTS/loremIpsum01.txt")
write(x = loremIpsum02, file = "RESULTS/loremIpsum02.txt")
write(x = loremIpsum03, file = "RESULTS/loremIpsum03.txt")
write(x = loremIpsum04, file = "RESULTS/loremIpsum04.txt")
write(x = loremIpsum05, file = "RESULTS/loremIpsum05.txt")
wordOccuFreq <- getTXT(myTXTs = list.files(path = paste0(getwd(),
  "/RESULTS/"), pattern = "loremIpsum", full.names = TRUE))
file.remove(list.files(full.names = TRUE,
  path = paste0(getwd(), "/RESULTS"), pattern = "loremIpsum"))

## End(Not run)
```

getwordOccuDF — *A quick way to obtain the word-occurrence data.frame from a set of documents.*

Description

A quick way to obtain the word-occurrence data.frame from a set of documents.

Usage

```
getwordOccuDF(mywd, language = "English", excludeSW = TRUE)
```

Arguments

`mywd`	A character variable containing the working directory.
`language`	The language used ("French", "English", "Spanish").
`excludeSW`	A logical to exclude stop words.

Value

A single word-occurrrence data.frame.

Examples

```
## Not run:
data("loremIpsum")
loremIpsum01 <- loremIpsum[1:100]
loremIpsum02 <- loremIpsum[101:200]
loremIpsum03 <- loremIpsum[201:300]
loremIpsum04 <- loremIpsum[301:400]
loremIpsum05 <- loremIpsum[401:500]
subDir <- "RESULTS"
dir.create(file.path(getwd(), subDir), showWarnings = FALSE)
write(x = loremIpsum01, file = "RESULTS/loremIpsum01.txt")
write(x = loremIpsum02, file = "RESULTS/loremIpsum02.txt")
write(x = loremIpsum03, file = "RESULTS/loremIpsum03.txt")
write(x = loremIpsum04, file = "RESULTS/loremIpsum04.txt")
write(x = loremIpsum05, file = "RESULTS/loremIpsum05.txt")
wordOccuDF <- getwordOccuDF(mywd = paste0(getwd(), "/RESULTS"),
  excludeSW = FALSE)
file.remove(list.files(full.names = TRUE,
  path = paste0(getwd(), "/RESULTS"), pattern = "loremIpsum"))

## End(Not run)
```

getXFreqWord — *Returns most frequent words*

Description

Returns most frequent words

Usage

```
getXFreqWord(wordF, occuWords)
```

Arguments

wordF	The data.frame containing word occurrences.
occuWords	The minimum number of occurrences for words to be returned.

Value

A vector with most frequent words.

Examples

```
data("wordOccuDF")
getXFreqWord(wordF = wordOccuDF, occuWords = 5)
```

IdentifyStructure — *Copy of the identifyStructure function from Tad Dallas metacom package.*

Description

Identifies structure (or quasi-structure) and outputs a classification.

Usage

```
IdentifyStructure(metacom.obj)
```

Arguments

metacom.obj	The result of the ‘Metacommunity‘ function, containing a list of 4 elements; the empirical matrix being tested, and results for coherence, turnover, and boundary clumping.

Details

Tad Dallas <tdallas@uga.edu> identifyStructure function no longer maintained in metacom package. see https://github.com/taddallas/metacom. This function was copy-pasted from version 1.4.4 of package metacom with minor modification (fix warning: the condition has length > 1 and only the first element will be used).

Value

Ouputs a classification of the metacommunity.

Note

Quasi structures, as well as 'random' and 'Gleasonian' structures, may not strictly be discernable through the EMS approach, as they rely on inferring a result from a non-significant test ('accepting the null'), which is typically a bad idea.

`inpdfr` *inpdfr: A package to analyse PDF Files Using Ecological Tools.*

Description

The inpdfr package allows analysing and comparing PDF/TXT documents using both classical text mining tools and those from theoretical ecolgy. In the later, words are considered as species and documents as communities, therefore allowing analysis at the community and metacommunity levels. The inpdfr package provides three cathegories of functions: functions to extract and process text into a word-occurrence data.frame, functions to analyse the word-occurrence data.frame with standard and ecological tools, and functions to use inpdfr through a Gtk2 Graphical User Interface (GitHub version only).

`loremIpsum` *Lorem Ipsum text.*

Description

A vector containing a Lorem Ipsum text for testing purposes.

Usage

```
loremIpsum
```

Format

A vector with 556 elements, each element corresponds to a line in the original text (character).

Source

`https://lipsum.com/`.

makeWordcloud *Word cloud based on the word-occurrence data.frame.*

Description

Plot a word cloud from the word-occurrence data.frame using `wordcloud` function.

Usage

```
makeWordcloud(
  wordF,
  wcFormat = "png",
  wcminFreq = 3,
  wcmaxWords = Inf,
  wcRandOrder = FALSE,
  wcCol = RColorBrewer::brewer.pal(8, "Dark2"),
  getPlot = c(TRUE, TRUE),
  mwidth = 1000,
  mheight = 1000,
  formatType = "png"
)
```

Arguments

`wordF`	The data.frame containing word occurrences.
`wcFormat`	Output format for the word cloud (deprecated, only "png").
`wcminFreq`	Minimum word frequency for words to be ploted (see `wordcloud`).
`wcmaxWords`	Maximum number of words to be ploted (see `wordcloud`).
`wcRandOrder`	Plot words in random order (see `wordcloud`).
`wcCol`	Color words (see `wordcloud`).
`getPlot`	A vector with two logical values. If `plots[1]==TRUE`, a word cloud is made for each document. If `plots[2]==TRUE`, a word cloud is made for the combinaison of all documents.
`mwidth`	The width of the plot in pixels.
`mheight`	The height of the plot in pixels.
`formatType`	The format for the output file ("eps", "pdf", "png", "svg", "tiff", "jpeg", "bmp").

Examples

```
## Not run:
makeWordcloud(wordF = myDF)

## End(Not run)
```

mergeWordFreq — *Merge word-occurrence data.frames into a single data.frame.*

Description

Merge word-occurrence data.frames into a single data.frame.

Usage

```
mergeWordFreq(wordF)
```

Arguments

wordF	The data.frame containing word occurrences.

Value

A single word-occurrrence data.frame with each column corresponding to a text file.

Examples

```
## Not run:
data("loremIpsum")
loremIpsum01 <- loremIpsum[1:100]
loremIpsum02 <- loremIpsum[101:200]
loremIpsum03 <- loremIpsum[201:300]
loremIpsum04 <- loremIpsum[301:400]
loremIpsum05 <- loremIpsum[401:500]
subDir <- "RESULTS"
dir.create(file.path(getwd(), subDir), showWarnings = FALSE)
write(x = loremIpsum01, file = "RESULTS/loremIpsum01.txt")
write(x = loremIpsum02, file = "RESULTS/loremIpsum02.txt")
write(x = loremIpsum03, file = "RESULTS/loremIpsum03.txt")
write(x = loremIpsum04, file = "RESULTS/loremIpsum04.txt")
write(x = loremIpsum05, file = "RESULTS/loremIpsum05.txt")
wordOccuFreq <- getTXT(myTXTs = list.files(path = paste0(getwd(),
  "/RESULTS/"), pattern = "loremIpsum", full.names = TRUE))
wordOccuDF <- mergeWordFreq(wordF = wordOccuFreq)
file.remove(list.files(full.names = TRUE,
  path = paste0(getwd(), "/RESULTS"), pattern = "loremIpsum"))

## End(Not run)
```

postProcTxt *Prossess vectors containing words into a data.frame of word occurrences.*

Description

Prossess vectors containing words into a data.frame of word occurrences.

Usage

```
postProcTxt(txt, minword = 1, maxword = 20, minFreqWord = 1)
```

Arguments

`txt`	A vector containing text.
`minword`	An integer specifying the minimum number of letters per word into the returned data.frame.
`maxword`	An integer to specifying the maximum number of letters per word into the returned data.frame.
`minFreqWord`	An integer specifying the minimum word frequency into the returned data.frame.

Value

A data.frame (freq = occurrences, stem = stem words, word = words), sorted by word occurrences.

preProcTxt *Extract text from txt files and pre-process content.*

Description

Extract text from txt files and pre-process content.

Usage

```
preProcTxt(filetxt, encodingIn = "UTF-8", encodingOut = "UTF-8")
```

Arguments

`filetxt`	A character containing the name of a txt file.
`encodingIn`	Encoding of the text file (default = "UTF-8").
`encodingOut`	Encoding of the text extracted (default = "UTF-8").

Value

A character vector with the content of the pre-process txt file (one element per line).

Examples

```
## Not run:
data("loremIpsum")
subDir <- "RESULTS"
dir.create(file.path(getwd(), subDir), showWarnings = FALSE)
write(x = loremIpsum, file = "RESULTS/loremIpsum.txt")
preProcTxt(filetxt = paste0(getwd(), "/RESULTS/loremIpsum.txt"))
file.remove(list.files(full.names = TRUE,
  path = paste0(getwd(), "/RESULTS"), pattern = "loremIpsum"))

## End(Not run)
```

quitSpaceFromChars *Delete spaces in file names.*

Description

Delete spaces in file names located in the current working directory.

Usage

```
quitSpaceFromChars(vectxt)
```

Arguments

vectxt	A vector containing character entries corresponding to the names of files in the current working directory.

Value

The function returns a logical for each file, with TRUE if the file has been found, and FALSE otherwise.

Examples

```
## Not run:
quitSpaceFromChars(c("my pdf.pdf","my other pdf.pdf"))

## End(Not run)
```

truncNumWords *Truncate the word-occurrence data.frame.*

Description

Truncate the word-occurrence data.frame.

Usage

```
truncNumWords(wordF, maxWords)
```

Arguments

`wordF`	The data.frame containing word occurrences.
`maxWords`	The maximum number of words in the data.frame.

Value

The data.frame containing word occurrences.

Examples

```
## Not run:
truncNumWords(wordF = myWordOccurrenceDF, maxWords = 50)

## End(Not run)
```

wordOccuDF *Lorem Ipsum word occurrences.*

Description

Lorem Ipsum word occurrences.

Usage

```
wordOccuDF
```

Format

A data.frame containing word name and occurences for testing purposes.

Index