

# Package ‘HFWutils’

February 14, 2012

**Version** 0.9.5.2011.03.21

**Date** 2011-03-21

**Title** csv import, csv export, garbage collection,string matching and passing by reference

**Author** Felix Wittmann <hfwittmann@gmail.com>

**Maintainer** Felix Wittmann <hfwittmann@gmail.com>

**LazyLoad** yes

**ZipData** no

**Depends** R (>= 2.0.0),debug,grDevices,methods,R.oo

**Description** package containing functions for convenient csv import and export with a view of using R as a calculation back end to spreadsheet-like programs such as scale in open office; string matching and passing by reference

**License** GPL (>= 2)

**Repository** CRAN

**Date/Publication** 2011-03-21 15:56:05

## R topics documented:

align . . . . .	2
as.ExcelDate . . . . .	3
ByRef . . . . .	5
Clean-methods . . . . .	6
cleanTexts . . . . .	7
Contacts . . . . .	8
df2matrix . . . . .	9
doReplace . . . . .	10
excelDate2Date . . . . .	11
export.R2X.Files . . . . .	12
extractFromList . . . . .	14

gcHFW . . . . .	16
getFunctionName . . . . .	17
import.X2R.Files . . . . .	18
initReplaceBy . . . . .	19
inst . . . . .	20
largeDataframe . . . . .	20
logDataframe . . . . .	22
logMessage . . . . .	23
Matches . . . . .	25
MatchesIndices . . . . .	26
matrix2df . . . . .	28
profiling . . . . .	29
sourceRfiles . . . . .	31

<b>Index</b>	<b>33</b>
--------------	-----------

---

align	<i>align ordering of two vectors</i>
-------	--------------------------------------

---

### Description

order one vector to match order of first vector

### Usage

```
align(names1, names2)
```

### Arguments

names1	~~ Describe names1 here ~~
names2	~~ Describe names2 here ~~

### Details

inputs names1,names2 must have same lengths

### Value

~Describe the value returned If it is a LIST, use

comp1	Description of 'comp1'
comp2	Description of 'comp2'

...

### Note

~~ further notes ~~

**Author(s)**

Felix Wittmann <hfittmann@gmail.com>

**References**

~put references to the literature/web site here ~

**See Also**

~~objects to See Also as [help](#), ~~~

**Examples**

```
x1 <- 1:4
x2 <- c(1,3,2,4)
align(names1=x1, names2=x2)
# [1] 1 3 2 4
```

```
x1 <- LETTERS[1:4]
x2 <- c(1,3,2,4)
align(names1=x1, names2=x2)
# [1] 1 3 2 4
```

```
x1 <- LETTERS[1:4]
x2 <- LETTERS[c(1,3,2,4)]
align(names1=x1, names2=x2)
# [1] 1 3 2 4
```

```
x1 <- LETTERS[1:5]
x2 <- LETTERS[c(1,3,2,4)]
align(names1=x1, names2=x2)
# lengths don't match :
# names1 : A B C D E
# names2 : A C B D
```

---

as.ExcelDate

*convert date to Excel date integer*

---

**Description**

~~ A concise description of what the function does. ~~

**Usage**

```
as.ExcelDate(DateIn)
```

**Arguments**

DateIn           ~~ Describe DateIn here ~~

**Details**

stimmt nicht vor "1900-03-01"

**Value**

~Describe the value returned If it is a LIST, use

comp1           Description of 'comp1'

comp2           Description of 'comp2'

...

**Note**

~~ further notes ~~

**Author(s)**

Felix Wittmann <hfwittmann@gmail.com>

**References**

~put references to the literature/web site here ~

**See Also**

~~objects to See Also as [excelDate2Date](#), ~~~

**Examples**

```
as.ExcelDate("2007/01/15")
# [1] 39097
```

```
as.ExcelDate("2007-01-15")
# [1] 39097
```

---

ByRef                      *object for passing by reference*

---

### Description

creates object for passing by reference

### Usage

ByRef(Name, vars)

### Arguments

Name	object name
vars	Data is put in here as a list

### Details

creates R.oo object for passing by reference. Data is held in \$.vars

### Value

~Describe the value returned If it is a LIST, use

comp1	Description of 'comp1'
comp2	Description of 'comp2'

...

### Author(s)

Felix Wittmann <hfwittmann@gmail.com>

### References

~put references to the literature/web site here ~

### See Also

~~objects to See Also as [help](#), ~~~

**Examples**

```
G <- ByRef(Name="Global",vars = 1:10)

G
# "This is a R.oo object of type Global"

G$.vars
# [1] 1 2 3 4 5 6 7 8 9 10

f <- function (x) G$.vars <- c(G$.vars,11)

f(G)
G$.vars
# [1] 1 2 3 4 5 6 7 8 9 10 11
```

---

Clean-methods

*remove a pattern in an object*


---

**Description**

~~ Methods for function Clean ~~

**Methods**

**x = "list", pattern = "character"** method for list object

**x = "vector", pattern = "character"** method for vector object

**x = "matrix", pattern = "character"** method for matrix object

**Author(s)**

Felix Wittmann <hfwittmann@gmail.com>

**Examples**

```
pattern <- "aX"

LIST <- list(paste("aXX",c(1:10)))
VECTOR <- paste("aXX",c(1:10))
MATRIX <- matrix(paste("aXX",c(1:25)),nrow=5)

Clean(x=LIST,pattern=pattern)
# [[1]]
# [1] "X 1" "X 2" "X 3" "X 4" "X 5" "X 6" "X 7" "X 8" "X 9" "X 10"

Clean(x=VECTOR,pattern=pattern)
```

```
# [1] "X 1" "X 2" "X 3" "X 4" "X 5" "X 6" "X 7" "X 8" "X 9" "X 10"

Clean(x=MATRIX,pattern=pattern)
#      [,1] [,2] [,3] [,4] [,5]
# [1,] "X 1" "X 6" "X 11" "X 16" "X 21"
# [2,] "X 2" "X 7" "X 12" "X 17" "X 22"
# [3,] "X 3" "X 8" "X 13" "X 18" "X 23"
# [4,] "X 4" "X 9" "X 14" "X 19" "X 24"
# [5,] "X 5" "X 10" "X 15" "X 20" "X 25"
```

---

cleanTexts	<i>adds spaces, removes multiple spaces, and trims spaces</i>
------------	---

---

### Description

vectorised clean up does three things a) adds spaces around special characters b) gets rid of multiple spaces c) trims leading and trailing spaces

### Usage

```
cleanTexts(x)
```

### Arguments

x                   ~~ Describe x here ~~

### Details

~~ If necessary more details than the description above ~~

### Note

~~ further notes ~~

### Author(s)

Felix Wittmann <hfwittmann@gmail.com>

### References

~put references to the literature/web site here ~

### See Also

~~objects to See Also as [help](#), ~~~

**Examples**

```
s0 <- c('asd:-asd', 'asd?asd')
# [1] "asd:-asd" "asd?asd"
cleanTexts(s0)
# [1] "asd : - asd" "asd ? asd"
```

---

 Contacts
 

---



---

 ~~ data name/kind ... ~~
 

---

**Description**

~~ A concise (1-5 lines) description of the dataset. ~~

**Usage**

```
data(Contacts)
```

**Format**

A data frame with 4 observations on the following 6 variables.

Function a factor with levels Client acquisition Client service Contact Function

Name a factor with levels Martin Paul Name Paris Hilton Paul Gerhard

Telephone a factor with levels 44 122222222222 44 207 777777 44 26 66666666 Telephone

Fax a factor with levels 44 122222222222 44 207 777777 44 26 66666666 Fax

E.Mail a factor with levels e@company.com E.Mail

**Details**

example data

**Author(s)**

Felix Wittmann <hfwittmann@gmail.com>

**Source**

~~ reference to a publication or URL from which the data were obtained ~~

**References**

~~ possibly secondary sources and usages ~~

**See Also**

[df2matrix](#), [logDataframe](#)

**Examples**

```
# see links
```

---

df2matrix	<i>data.frame to matrix</i>
-----------	-----------------------------

---

**Description**

converts data.frame to matrix

**Usage**

```
df2matrix(x_df)
```

**Arguments**

x\_df           ~~ Describe x\_df here ~~

**Details**

~~ If necessary more details than the description above ~~

**Value**

~Describe the value returned If it is a LIST, use

comp1           Description of 'comp1'

comp2           Description of 'comp2'

...

**Note**

~~ further notes ~~

**Author(s)**

Felix Wittmann <hf Wittmann@gmail.com>

**References**

~put references to the literature/web site here ~

**See Also**

[logDataFrame](#), [Contacts](#)

**Examples**

```

data(Contacts)
Contacts
#   X      Function      Name      Telephone      Fax
# 1 NA      Function      Name      Telephone      Fax
# 2 1      Contact      Martin Paul    44 207 777777    44 207 777777
# 3 2 Client acquisition Paul Gerhard    44 26 66666666    44 26 66666666
# 4 3      Client service Paris Hilton    44 12222222222222 44 12222222222222
#      E.Mail
# 1      E.Mail
# 2 e@company.com
# 3 e@company.com
# 4 e@company.com

```

---

doReplace                      *performs replacements in a string*

---

**Description**

performs replacements in a string using a dataframe set up with `initReplaceBy`

**Usage**

```
doReplace(sqlCommand, ReplaceBy, fixed = TRUE)
```

**Arguments**

sqlCommand        the target string  
ReplaceBy         a dataframe set up with `initReplaceBy`  
fixed              *~~ Describe fixed here ~~*

**Details**

*~~ If necessary more details than the description above ~~*

**Value**

*~Describe the value returned If it is a LIST, use*

comp1             Description of 'comp1'  
comp2             Description of 'comp2'

...

**Note**

~~ further notes ~~

**Author(s)**

Felix Wittmann <hfittmann@gmail.com>

**References**

~put references to the literature/web site here ~

**See Also**

~~objects to See Also as [initReplaceBy](#), ~~~

**Examples**

```
ReplaceBy <- initReplaceBy(2)
ReplaceBy[1,] <- c(Replace = "TABLE"           , By = 'USArrests')
ReplaceBy[2,] <- c(Replace = "CONDITION"      , By = 'Assault>250')

sqlCommand <- "SELECT * FROM TABLE WHERE CONDITION"
doReplace(sqlCommand, ReplaceBy)
# "SELECT * FROM USArrests WHERE Assault>250"
```

---

excelDate2Date            *convert Excel date integer to date*

---

**Description**

~~ A concise 1-5 lines description of what the function does. ~~

**Usage**

```
excelDate2Date(excelDate)
```

**Arguments**

excelDate            ~~ Describe excelDate here ~~

**Details**

stimmt nicht vor 1900-03-01

**Value**

~Describe the value returned If it is a LIST, use

```
comp1      Description of 'comp1'
comp2      Description of 'comp2'
...
```

**Note**

~~ further notes ~~

**Author(s)**

Felix Wittmann <hfwittmann@gmail.com>

**References**

~put references to the literature/web site here ~

**See Also**

~~objects to See Also as [as.ExcelDate](#), ~~~

**Examples**

```
excelDate2Date(39097)
# [1] "2007-01-15"
excelDate2Date(0)
# "1899-12-30"
```

---

```
export.R2X.Files      ~~ function to do ... ~~
```

---

**Description**

export all variables held in R2X into csv files

**Usage**

```
export.R2X.Files(R2X, pfaad = "data/R2X/")
```

**Arguments**

```
R2X          is a list
pfaad        can be any folder name
```

**Details**

R2X is a list.

**Value**

~Describe the value returned If it is a LIST, use

comp1            Description of 'comp1'

comp2            Description of 'comp2'

...

**Note**

~~ further notes ~~

**Author(s)**

Felix Wittmann <hfwittmann@gmail.com>

**References**

~put references to the literature/web site here ~

**See Also**

~~objects to See Also as [import.X2R.Files](#), ~~~

**Examples**

```
x<- 1:10
y <- matrix(LETTERS[1:25],nrow=5,ncol=5)

R2X <- list(x=x,y=y)

inst()
export.R2X.Files(R2X)
# now the variables x and y are saved in corresponding csv files in the folder /data/R2X

X2R <- list()
X2R <- import.X2R.Files('data/R2X/')

# now the variables x and y are loaded from corresponding csv files in the folder /data/R2X into X2R

#> X2R$x
#   x
#1  1
#2  2
#3  3
#4  4
```

```

#5 5
#6 6
#7 7
#8 8
#9 9
#10 10#

#> X2R$y
# V1 V2 V3 V4 V5
#1 A F K P U
#2 B G L Q V
#3 C H M R W
#4 D I N S X
#5 E J O T Y

```

---

extractFromList	<i>extract variables From List</i>
-----------------	------------------------------------

---

### Description

extract data from lists of lists ; these are typically the output of a lapply or sapply

### Usage

```
extractFromList(mIn, names)
```

### Arguments

mIn	~~ Describe mIn here ~~
names	~~ Describe names here ~~

### Details

~~ If necessary more details than the description above ~~

### Value

~Describe the value returned If it is a LIST, use

comp1	Description of 'comp1'
comp2	Description of 'comp2'
...	

**Note**

~~ further notes ~~

**Author(s)**

Felix Wittmann <hfwittmann@gmail.com>

**References**

~put references to the literature/web site here ~

**See Also**

~~objects to See Also as [help](#), ~~~

**Examples**

```
c1 <- list(a=1,b=2)
c2 <- list(a=3,b=4)

d<- list(c1,c2)
d
#      [[1]]
#      [[1]]$a
#      [1] 1
#
#      [[1]]$b
#      [1] 2
#
#
#      [[2]]
#      [[2]]$a
#      [1] 3
#
#      [[2]]$b
#      [1] 4

extractFromList(d,names=c('a','b'))

# $a
#      [,1]
#[1,] 1
#[2,] 3
#
# $b
#      [,1]
#[1,] 2
#[2,] 4
```

---

gcHFW

*garbage collection in global environment*

---

**Description**

performs garbage collection in the global environment

**Usage**

gcHFW()

**Details**

takes a sizable amount of time (ie is slow) but may be used for processes which take a lot of memory

**Value**

~Describe the value returned If it is a LIST, use

comp1            Description of 'comp1'

comp2            Description of 'comp2'

...

**Author(s)**

Felix Wittmann <hfwittmann@gmail.com>

**References**

~put references to the literature/web site here ~

**See Also**

~~objects to See Also as [help](#), ~~~

**Examples**

gcHFW()

---

`getFunctionName`      *knows name of current function*

---

**Description**

~~ A concise 1-5 lines description of what the function does. ~~

**Usage**

`getFunctionName(generationBack = 1)`

**Arguments**

`generationBack`    ~~ Describe `generationBack` here ~~

**Details**

~~ If necessary more details than the description above ~~

**Value**

~Describe the value returned If it is a LIST, use

`comp1`              Description of 'comp1'

`comp2`              Description of 'comp2'

...

**Note**

~~ further notes ~~

**Author(s)**

Felix Wittmann <hf Wittmann@gmail.com>

**References**

~put references to the literature/web site here ~

**See Also**

~~ objects to See Also as [help](#), ~~

**Examples**

```
x <- function() getFunctionName()
x()

test <- function() getFunctionName()
test()
```

---

```
import.X2R.Files      ~~ function to do ... ~~
```

---

**Description**

all csv from a given folder are loaded into a variable, G, where G is a list.

**Usage**

```
import.X2R.Files(pfad = "data/X2R/")
```

**Arguments**

```
pfad          ~~ Describe pfad here ~~
```

**Details**

*~~ If necessary more details than the description above ~~*

**Value**

~Describe the value returned If it is a LIST, use

```
comp1          Description of 'comp1'
comp2          Description of 'comp2'
```

...

**Note**

*~~ further notes ~~*

**Author(s)**

```
Felix Wittmann <hfittmann@gmail.com>
```

**References**

~put references to the literature/web site here ~

**See Also**

~~objects to See Also as [export.R2X.Files](#), ~~~

---

initReplaceBy	<i>sets up dataframe used by doReplace</i>
---------------	--

---

**Description**

~~ A concise 1-5 lines description of what the function does. ~~

**Usage**

initReplaceBy(n)

**Arguments**

n                   ~~ Describe n here ~~

**Details**

~~ If necessary more details than the description above ~~

**Value**

~Describe the value returned If it is a LIST, use

comp1               Description of 'comp1'

comp2               Description of 'comp2'

...

**Author(s)**

Felix Wittmann <hfwittmann@gmail.com>

**References**

~put references to the literature/web site here ~

**See Also**

~~objects to See Also as [doReplace](#), ~~~

---

inst	<i>provides path to working directory</i>
------	---

---

**Description**

if directory name ends with ~.Rcheck switch to sub-directory ~ contained in ~.Rcheck. Example directory name is TAutils.Rcheck then switch to sub-directory TAutils.

**Usage**

```
inst()
```

**Details**

useful for check process. In the check process these directories have already been created.

**Author(s)**

Felix Wittmann <hfwittmann@gmail.com>

**References**

~put references to the literature/web site here ~

---

largeDataframe	<i>produce dataframe with constant columns</i>
----------------	--

---

**Description**

produce dataframe with constant columns. as specified by the input row vector

**Usage**

```
largeDataframe(smallDf, Times)
```

**Arguments**

smallDf	input row vector
Times	number of rows

**Details**

~~ If necessary more details than the description above ~~

**Value**

~Describe the value returned If it is a LIST, use

comp1            Description of 'comp1'

comp2            Description of 'comp2'

...

**Note**

~~ further notes ~~

**Author(s)**

Felix Wittmann <hfwittmann@gmail.com>

**References**

~put references to the literature/web site here ~

**See Also**

~~objects to See Also as [help](#), ~~~

**Examples**

```
strategic <- data.frame(REX = 0.50, DJSTOXX=0.17,SP500=0.17,SPI=0.08,NIKKEI=0.08)

# strategic
#   REX DJSTOXX SP500  SPI NIKKEI
# 1 0.5   0.17  0.17 0.08  0.08

largeDataframe(smallDf=strategic, Times=10)
#   REX DJSTOXX SP500  SPI NIKKEI
# 1 0.5   0.17  0.17 0.08  0.08
# 2 0.5   0.17  0.17 0.08  0.08
# 3 0.5   0.17  0.17 0.08  0.08
# 4 0.5   0.17  0.17 0.08  0.08
# 5 0.5   0.17  0.17 0.08  0.08
# 6 0.5   0.17  0.17 0.08  0.08
# 7 0.5   0.17  0.17 0.08  0.08
# 8 0.5   0.17  0.17 0.08  0.08
# 9 0.5   0.17  0.17 0.08  0.08
# 10 0.5  0.17  0.17 0.08  0.08
```

---

logDataframe	<i>save a dataframe in a dumps directory</i>
--------------	--

---

**Description**

save data.frame as csv file in a dumps directory for logging purposes

**Usage**

```
logDataframe(x, fileName = as.character(substitute(x)))
```

**Arguments**

x	~~ Describe x here ~~
fileName	~~ Describe fileName here ~~

**Details**

append=TRUE col.names = FALSE sep=','

**Value**

~Describe the value returned If it is a LIST, use

comp1	Description of 'comp1'
comp2	Description of 'comp2'

...

**Note**

~~ further notes ~~

**Author(s)**

Felix Wittmann <hfwittmann@gmail.com>

**References**

~put references to the literature/web site here ~

**See Also**

[df2matrix](#), [Contacts](#)

**Examples**

```

data(Contacts)
ContactsTest <- Contacts

inst()
unlink("data/dumps/ContactsTest.csv")

```

```

ContactsTest
# X Function Name Telephone Fax E.Mail
# 1 NA Function Name Telephone Fax E.Mail
# 2 1 Contact Martin Paul 44 207 777777 44 207 777777 e@company.com
# 3 2 Client acquisition Paul Gerhard 44 26 66666666 44 26 66666666 e@company.com
# 4 3 Client service Paris Hilton 44 122222222222 44 122222222222 e@company.com

```

```

logDataframe(ContactsTest)
read.csv(file="data/dumps/ContactsTest.csv",sep=";")
# X1 NA. Function Name Telephone Fax
# 1 2 1 Contact Martin Paul 44 207 777777 44 207 777777
# 2 3 2 Client acquisition Paul Gerhard 44 26 66666666 44 26 66666666
# 3 4 3 Client service Paris Hilton 44 122222222222 44 122222222222
# E.Mail X2007.08.28.19.06.15
#1 e@company.com 2007-08-28 19:06:15
#2 e@company.com 2007-08-28 19:06:15
#3 e@company.com 2007-08-28 19:06:15

```

```

logDataframe(ContactsTest)
read.csv(file="data/dumps/ContactsTest.csv",sep=";")
# X1 NA. Function Name Telephone Fax
#1 2 1 Contact Martin Paul 44 207 777777 44 207 777777
#2 3 2 Client acquisition Paul Gerhard 44 26 66666666 44 26 66666666
#3 4 3 Client service Paris Hilton 44 122222222222 44 122222222222
#4 1 NA Function Name Telephone Fax
#5 2 1 Contact Martin Paul 44 207 777777 44 207 777777
#6 3 2 Client acquisition Paul Gerhard 44 26 66666666 44 26 66666666
#7 4 3 Client service Paris Hilton 44 122222222222 44 122222222222
# E.Mail X2007.08.28.19.06.15
#1 e@company.com 2007-08-28 19:06:15
#2 e@company.com 2007-08-28 19:06:15
#3 e@company.com 2007-08-28 19:06:15
#4 E.Mail 2007-08-28 19:07:10
#5 e@company.com 2007-08-28 19:07:10
#6 e@company.com 2007-08-28 19:07:10
#7 e@company.com 2007-08-28 19:07:10

```

**Description**

message file contains additional information when and in which function the message was produced

**Usage**

```
logMessage(messageText)
```

**Arguments**

messageText     ~~ Describe messageText here ~~

**Details**

~~ If necessary more details than the description above ~~

**Value**

~Describe the value returned If it is a LIST, use

comp1            Description of 'comp1'

comp2            Description of 'comp2'

...

**Note**

~~ further notes ~~

**Author(s)**

Felix Wittmann <hfwittmann@gmail.com>

**References**

~put references to the literature/web site here ~

**See Also**

~~objects to See Also as [help](#), ~~~

**Examples**

```
bla <- function() logMessage('testBla')
bla()
```

---

Matches *string and patterns matching in character Vector*

---

**Description**

perform string and patterns matching in character Vector

**Usage**

```
Matches(Vector, patt, fixed = FALSE)
```

**Arguments**

Vector	~~ Describe Vector here ~~
patt	~~ Describe patt here ~~
fixed	~~ Describe fixed here ~~

**Details**

result is returned as a matrix

**Value**

~Describe the value returned If it is a LIST, use

comp1           Description of 'comp1'

comp2           Description of 'comp2'

...

**Note**

~~ further notes ~~

**Author(s)**

Felix Wittmann <hfwittmann@gmail.com>

**References**

~put references to the literature/web site here ~

**See Also**

~~objects to See Also as [help](#), ~~~

**Examples**

```

# Example 1
Vector <- "sum(F17:G18)+G190*G2+ sum(A1:B2)"
patt <- '[A-F]+[0-9]+'
Matches (Vector,patt)

#                               [A-F]+[0-9]+
#   sum(F17:G18)+G190*G2+ sum(A1:B2) Character,3

# Example 2
Matches (c(Vector, "sum(A1:B2)"),patt)

#                               [A-F]+[0-9]+
#   sum(F17:G18)+G190*G2+ sum(A1:B2) Character,3
#   sum(A1:B2)                      Character,2

# Example 3
Matches (Vector =c("Patrick Pasot, Patrick Schmidt","Patrick Denker"),
        patt   =c("Patrick","Schmidt")
        ) # Matches
#           Patrick   Schmidt
#Patrick Pasot, Patrick Schmidt Character,2 "Schmidt"
#Patrick Denker                "Patrick"   Character,0

# Example 4
out <- Matches (Vector =c("Patrof Pasot, Patrum Schmidt","Pater Denker"),
               patt   =c("Pat[a-z]+")
               ) # Matches

out
#                               Pat[a-z]+
#Patrick Pasot, Patrum Schmidt Character,2
#Pater Denker                "Pater"   v

out[1,1][[1]]
#[1] "Patrof" "Patrum"

```

---

MatchesIndices      *provides boolean vectors where matches were found*

---

**Description**

returns indices whether patterns were matched succesfully in a character vector



```

# TRUE

# Example 2
MatchesIndices (c(Vector, "sum(A1:B2)"),patt)

#      $index_x
#      sum(F17:G18)+G190*G2+ sum(A1:B2)          sum(A1:B2)
#              TRUE                               TRUE
#
#      $index_patt
#      [A-F]+[0-9]+
#              TRUE

# Example 3
MatchesIndices (x =c("Patrick Pasot, Patrick Schmidt","Patrick Denker"),
               patt =c("Patrick","Schmidt")
               ) # Matches

# $index_x
# Patrick Pasot, Patrick Schmidt          Patrick Denker
#              TRUE                       TRUE
#
# $index_patt
# Patrick Schmidt
# TRUE TRUE

# Example 4
out <- MatchesIndices (x =c("Patrick Pasot, Patrick Schmidt","Patrick Denker"),
                      patt =c("Pat[a-z]+")
                      ) # Matches

out
# $index_x
# Patrick Pasot, Patrick Schmidt          Patrick Denker
#              TRUE                       TRUE
#
# $index_patt
# Pat[a-z]+
# TRUE

```

---

matrix2df

*matrix to data.frame*


---

### Description

converts matrix to data.frame

**Usage**

matrix2df(mIn)

**Arguments**

mIn                   ~~ Describe mIn here ~~

**Details**

~~ If necessary more details than the description above ~~

**Value**

~Describe the value returned If it is a LIST, use

comp1                Description of 'comp1'

comp2                Description of 'comp2'

...

**Note**

~~ further notes ~~

**Author(s)**

Felix Wittmann <hfwittmann@gmail.com>

**References**

~put references to the literature/web site here ~

**See Also**

~~objects to See Also as [df2matrix](#), ~~~

---

profiling

*plots tree of execution times*

---

**Description**

determines how much time a function its and sub-functions (and sub-functions thereof etc) take to run ('profiling'). Also draws picture of this using the interrelations of functions.

**Usage**

```
profiling(      s = NULL,
               myFunction,
               myFunctionsBase,
plotType = c("profilingAll", "profilingTopQuantile" , "foodwebAll", "foodwebTopQuantile"),
cex = 1,
               ... ,
               topQuantile = 0.25)
```

**Arguments**

s	is optional; would be the result returned by a previous profiling run
myFunction	myFunction is the function to be profiled
myFunctionsBase	is a character vector of the functions that are used in the analysis
plotType	a character string either "profilingAll", "profilingTopQuantile" for a foodwebplot including the profiled times or a "foodwebAll", "foodwebTopQuantile" for the foodwebplot without the profiled times. TopQuantile shows only the functions that were profiled to take the most time
cex	is a standard plot parameter relating to the font size
...	are additional arguments that will in turn be passed to the 'foodweb' function
topQuantile	specify the limit for the display of the top time consumers, by quantile

**Author(s)**

Felix Wittmann <hfwittmann@gmail.com>

**Examples**

```
#####

y <- 0

testFunction1 <- function(X)
  {
    for (x in 1:X) {y <- y + x}
    return(y)
  } # function
testFunction2 <- function(X) testFunction1(X)
testFunction3 <- function(X) testFunction1(X)
testFunction4 <- function() {
  testFunction2(1000000)
  testFunction3(1000000)
} # testFunction4

#####
# namespacePackage <- find.funs("package:base")
# myFunctionsBase <- find.funs(pos=1)
##ByRef is R.oo-based; R.oo-based constructions don't seem to work with the foodweb from the mvbutils package
```

```
s1<-profiling(myFunction=testFunction4,myFunctionsBase=myFunctionsBase,plotType = "profilingAll")
s2<-profiling(myFunction=testFunction4,myFunctionsBase=myFunctionsBase,plotType = "profilingTopQuantile")
s3<-profiling(myFunction=testFunction4,myFunctionsBase=myFunctionsBase,plotType = "foodwebAll")
s4<-profiling(myFunction=testFunction4,myFunctionsBase=myFunctionsBase,plotType = "foodwebTopQuantile")
```

---

sourceRfiles	<i>source all files in a given folder</i>
--------------	---

---

**Description**

sources all r files in a given folder

**Usage**

```
sourceRfiles(pfade = "R/")
```

**Arguments**

pfade                   ~~ Describe pfade here ~~

**Details**

~~ If necessary more details than the description above ~~

**Value**

~Describe the value returned If it is a LIST, use

comp1                   Description of 'comp1'

comp2                   Description of 'comp2'

...

**Note**

~~ further notes ~~

**Author(s)**

Felix Wittmann <hfittmann@gmail.com>

**References**

~put references to the literature/web site here ~

**See Also**

~~objects to See Also as [help](#), ~~~

**Examples**

```
sourceRfiles()
```

# Index

- \*Topic **character**
  - [cleanTexts](#), 7
- \*Topic **database**
  - [inst](#), 20
- \*Topic **datasets**
  - [Contacts](#), 8
- \*Topic **manip**
  - [align](#), 2
  - [as.ExcelDate](#), 3
  - [ByRef](#), 5
  - [df2matrix](#), 9
  - [doReplace](#), 10
  - [excelDate2Date](#), 11
  - [export.R2X.Files](#), 12
  - [extractFromList](#), 14
  - [gcHFW](#), 16
  - [getFunctionName](#), 17
  - [import.X2R.Files](#), 18
  - [initReplaceBy](#), 19
  - [largeDataframe](#), 20
  - [logDataframe](#), 22
  - [logMessage](#), 23
  - [Matches](#), 25
  - [MatchesIndices](#), 26
  - [matrix2df](#), 28
  - [profiling](#), 29
  - [sourceRfiles](#), 31
- \*Topic **methods**
  - [Clean-methods](#), 6
- [align](#), 2
- [as.ExcelDate](#), 3, 12
- [ByRef](#), 5
- [Clean \(Clean-methods\)](#), 6
- [Clean, list, character-method \(Clean-methods\)](#), 6
- [Clean, matrix, character-method \(Clean-methods\)](#), 6
- [Clean, vector, character-method \(Clean-methods\)](#), 6
- [Clean-methods](#), 6
- [cleanTexts](#), 7
- [Contacts](#), 8, 9, 22
- [df2matrix](#), 8, 9, 22, 29
- [doReplace](#), 10, 19
- [excelDate2Date](#), 4, 11
- [export.R2X.Files](#), 12, 19
- [extractFromList](#), 14
- [gcHFW](#), 16
- [getFunctionName](#), 17
- [help](#), 3, 5, 7, 15–17, 21, 24, 25, 27, 32
- [import.X2R.Files](#), 13, 18
- [initReplaceBy](#), 11, 19
- [inst](#), 20
- [largeDataframe](#), 20
- [logDataframe](#), 8, 9, 22
- [logMessage](#), 23
- [Matches](#), 25
- [MatchesIndices](#), 26
- [matrix2df](#), 28
- [profiling](#), 29
- [sourceRfiles](#), 31