

# Package ‘thinkr’

May 8, 2026

**Title** Tools for Cleaning Up Messy Files

**Version** 0.16

**Description** Some tools for cleaning up messy 'Excel' files to be suitable for R. People who have been working with 'Excel' for years built more or less complicated sheets with names, characters, formats that are not homogeneous. To be able to use them in R nowadays, we built a set of functions that will avoid the majority of importation problems and keep all the data at best.

**License** GPL-3

**URL** <https://github.com/Thinkr-open/thinkr>

**BugReports** <https://github.com/Thinkr-open/thinkr/issues>

**Depends** R (>= 3.1)

**Imports** assertthat, cli, devtools, dplyr, ggplot2, lazyeval, lubridate, magrittr, methods, officer, rvg, stats, stringi, stringr, tidyr, utils, withr

**Suggests** knitr, rmarkdown, testthat

**VignetteBuilder** knitr

**Encoding** UTF-8

**RoxygenNote** 7.2.0

**NeedsCompilation** no

**Author** Vincent Guyader [aut, cre] (ORCID:  
<<https://orcid.org/0000-0003-0671-9270>>),  
Sébastien Rochette [aut] (ORCID:  
<<https://orcid.org/0000-0002-1565-9313>>),  
ThinkR [cph]

**Maintainer** Vincent Guyader <vincent@thinkr.fr>

**Repository** CRAN

**Date/Publication** 2022-08-22 13:30:02 UTC

## Contents

.efface_test . . . . .	2
all_ggplot_to_pptx . . . . .	3
as_mon_numeric . . . . .	3
clean_levels . . . . .	4
clean_names . . . . .	5
clean_vec . . . . .	5
dput_levels . . . . .	6
excel_names . . . . .	6
find_name . . . . .	7
from_excel_to_posixt . . . . .	8
gsub2 . . . . .	8
is.01 . . . . .	9
is.12 . . . . .	9
is_full_figures . . . . .	10
is_full_na . . . . .	10
is_likert . . . . .	11
look_like_a_number . . . . .	11
make_unique . . . . .	12
peep . . . . .	12
replace_pattern . . . . .	13
save_as_csv . . . . .	14
set_col_type . . . . .	14
%ni% . . . . .	15
<b>Index</b>	<b>16</b>

---

.efface_test	<i>delete .test file in testthat folder</i>
--------------	---

---

### Description

Only usefull during package developpement using testthat package

### Usage

```
.efface_test()
```

---

all\_ggplot\_to\_pptx      *Save all ggplot in a pptx*

---

### Description

Save all ggplot in a pptx

### Usage

```
all_ggplot_to_pptx(  
  out = "tous_les_graphs.pptx",  
  open = TRUE,  
  png = TRUE,  
  folder = "dessin",  
  global = TRUE  
)
```

### Arguments

out	output file name
open	booleen open file after creation
png	booleen also save as png
folder	png's folder
global	booleen use .GlobalEnv

### Examples

```
## Not run:  
all_ggplot_to_pptx()  
  
## End(Not run)
```

---

as\_mon\_numeric      *Transform a vector into numeric if meaningful, even with bad decimal, space or %*

---

### Description

Transform a vector into numeric if meaningful, even with bad decimal, space or %

### Usage

```
as_mon_numeric(vec)
```

**Arguments**

vec                    a vector

**Details**

Note that text and factors are not transformed as numeric (except FALSE, TRUE, F, T), contrary to R default behavior with 'as.numeric(factor()'

**Value**

a numeric vector

**Examples**

```
as_mon_numeric(c("1", "0", "1"))
as_mon_numeric(c("1.3", "1,5", "1;6", "16%", "17 87 "))
as_mon_numeric(c(TRUE, "A", "F"))
as_mon_numeric(c(TRUE, TRUE, FALSE))
as_mon_numeric(factor(c("toto", "tata", "toto")))
```

---

clean\_levels

*Clean levels label*

---

**Description**

Clean levels label

**Usage**

```
clean_levels(vec, verbose = FALSE, translit = FALSE, punct = FALSE)
```

**Arguments**

vec                    a factor

verbose                boolean is the function verbose

translit                boolean remove non ascii character

punct                    boolean do you remove punctuation

---

clean_names	<i>clean_names</i>
-------------	--------------------

---

**Description**

clean\_names

**Usage**

```
clean_names(dataset, verbose = FALSE, translit = TRUE)
```

**Arguments**

dataset	a dataframe
verbose	logical
translit	logical remove non ascii character

**Value**

a dataframe

**Examples**

```
data(iris)
clean_names(iris)
```

---

clean_vec	<i>Clean character vector</i>
-----------	-------------------------------

---

**Description**

Clean character vector

**Usage**

```
clean_vec(
  vec,
  verbose = FALSE,
  unique = TRUE,
  keep_number = FALSE,
  translit = TRUE,
  punct = TRUE
)
```

**Arguments**

vec	character vector to clean
verbose	logical is the function verbose
unique	logical do we have to apply make_unique
keep_number	logical keep number at begining
translit	logical remove non ascii character
punct	logical do you remove punctuation

---

dput_levels	<i>return R instruction to create levels</i>
-------------	--

---

**Description**

return R instruction to create levels

**Usage**

```
dput_levels(vec)
```

**Arguments**

vec	a factor or character vector
-----	------------------------------

**Value**

a R instruction

**Examples**

```
dput_levels(iris$Species)
```

---

excel_names	<i>Get position or excel name of column</i>
-------------	---

---

**Description**

ncol\_to\_excel returns excel column name from a position number. excel\_to\_ncol returns excel column position number from a column name. excel\_col returns all excel column name.

**Usage**

```
ncol_to_excel(n)
excel_to_ncol(col_name)
excel_col()
```

**Arguments**

n                    the column position  
col\_name            the column name

**Examples**

```
ncol_to_excel(35)
excel_to_ncol("BF")
excel_col()
ncol_to_excel(1:6)
excel_to_ncol(c('AF', 'AG', 'AH'))
```

---

find_name	<i>find pattern in name's dataset</i>
-----------	---------------------------------------

---

**Description**

find pattern in name's dataset

**Usage**

```
find_name(dataset, pattern)
```

**Arguments**

dataset            a data.frame (or list or anything with names parameter)  
pattern            pattern we are looking for

**Value**

a list with position and value

**Examples**

```
find_name(iris,"Sepal")
```

---

`from_excel_to_posixt` *transform the excel numeric date format into POSIXct*

---

**Description**

transform the excel numeric date format into POSIXct

**Usage**

```
from_excel_to_posixt(vec, origin = "1904-01-01")
```

**Arguments**

<code>vec</code>	a vector
<code>origin</code>	a date-time object, or something which can be coerced by <code>as.POSIXct(tz = "GMT")</code> to such an object.

---

`gsub2` *like gsub but keep a factor as factor*

---

**Description**

like gsub but keep a factor as factor

**Usage**

```
gsub2(x, ...)
```

**Arguments**

<code>x</code>	a vector
<code>...</code>	les parametres de la fonction gsub

**Value**

a vector

---

is.01                      *does this vector only contains 0 and 1*

---

**Description**

does this vector only contains 0 and 1

**Usage**

is.01(x)

**Arguments**

x                      a vector

**Value**

a boolean

**Examples**

```
is.01(c(0,1,0,0,1))
is.01(c(0,1,0,0,5))
```

---

is.12                      *does this vector only contains 1 and 2*

---

**Description**

does this vector only contains 1 and 2

**Usage**

is.12(x)

**Arguments**

x                      a vector

**Value**

a boolean

**Examples**

```
is.12(c(1,1,2,1,2))
is.12(c(1,1,2,1,5))
```

---

is_full_figures	<i>Predicate for character vector full of figures</i>
-----------------	---

---

**Description**

detects if a character vector is only made with figures. Useful when you

**Usage**

```
is_full_figures(.)
```

**Arguments**

. a vector of character (and eventually NA's)

**Value**

a boolean

**Examples**

```
is_full_figures(c(NA,"0","25.3"))  
is_full_figures((c(NA,"0","25_3")))
```

---

is_full_na	<i>Predicate for full NA vector</i>
------------	-------------------------------------

---

**Description**

is\_full\_na test if the vector is full of NA's

**Usage**

```
is_full_na(.)
```

**Arguments**

. a vector

**Value**

a vector of boolean

**Examples**

```
is_full_na(c(NA, NA, NA))
```

---

is_likert	<i>is a factor a likert scale</i>
-----------	-----------------------------------

---

**Description**

is a factor a likert scale

**Usage**

```
is_likert(vec, lev)
```

**Arguments**

vec	a factor
lev	le scale

**Value**

boolean

**Examples**

```
is_likert(iris$Species,c("setosa","versicolor","virginica"))  
is_likert(iris$Species,c("setosa","versicolor","virginica","banana"))  
is_likert(iris$Species,c("setosa","versicolor"))
```

---

look_like_a_number	<i>return TRUE if this look like a number</i>
--------------------	---

---

**Description**

return TRUE if this look like a number

**Usage**

```
look_like_a_number(vec)
```

**Arguments**

vec	a vector
-----	----------

**Value**

un boolean

---

make_unique	<i>make.unique improvement</i>
-------------	--------------------------------

---

**Description**

make.unique improvement

**Usage**

```
make_unique(vec, sep = "_")
```

**Arguments**

vec	a vector
sep	char separator to use

**Value**

a vector

**Examples**

```
make_unique(c("a", "a", "a", "b", "a", "b", "c"))
```

---

peep	<i>peep the pipeline</i>
------	--------------------------

---

**Description**

peep some data at one step of a pipeline.

**Usage**

```
peep(data, ..., printer = print, verbose = FALSE)
```

**Arguments**

data	some data
...	function names or expressions that use . as a placeholder for the data
printer	which function use to print
verbose	TRUE to include what is printed

**Value**

the input data

**Examples**

```

if( require(magrittr) ){
  # just symbols
  iris %>% peep(head,tail) %>% summary
  # expressions with .
  iris %>% peep(head(., n=2),tail(., n=3) ) %>% summary
  # or both
  iris %>% peep(head,tail(., n=3) ) %>% summary
  # use verbose to see what happens
  iris %>% peep(head,tail(., n=3), verbose = TRUE) %>% summary
}

```

---

replace_pattern	<i>Replace pattern everywhere in a data.frame</i>
-----------------	---

---

**Description**

Replace pattern everywhere in a data.frame

**Usage**

```
replace_pattern(dataset, pattern, replacement, exact = FALSE)
```

**Arguments**

dataset	a data.frame
pattern	Pattern to look for.
replacement	A character of replacements.
exact	a boolean if TRUE the whole value need ton match

**Value**

a data.frame

**Examples**

```

dataset <- data.frame(
  col_a = as.factor(letters)[1:7],
  col_b = letters[1:7],
  col_c = 1:7,
  col_d = paste0(letters[1:7], letters[1:7]),
  stringsAsFactors = FALSE
)

# replace pattern
replace_pattern(dataset, "a", "XXX-")

# With exact matching
replace_pattern(dataset, "a", "XXX-", exact = TRUE)

```

---

save_as_csv	<i>export a data.frame to csv</i>
-------------	-----------------------------------

---

**Description**

export a data.frame to csv

**Usage**

```
save_as_csv(dataset, path, row.names = FALSE, ...)
```

**Arguments**

dataset	a data.frame
path	the path
row.names	booleen do we have to save the row names
...	other write.csv parameters

**Value**

file name as character

**Examples**

```
## Not run:
iris %>% save_as_csv(file.path(tempdir(), 'coucou.csv')) %>% browseURL()

## End(Not run)
```

---

set_col_type	<i>set a given coltype to each column in a data.frame</i>
--------------	---

---

**Description**

set a given coltype to each column in a data.frame

**Usage**

```
set_col_type(dataset, col_type)
```

**Arguments**

dataset            a data.frame  
col\_type           a character vector containing the class to apply

**Value**

a data.frame

---

%ni%                            *not in*

---

**Description**

not in

**Usage**

x %ni% table

**Arguments**

x                            vector or NULL: the values to be matched  
table                        the values to be matched against

**Examples**

```
"a" %ni% letters  
"coucou" %ni% letters
```

# Index

`.efface_test`, 2  
`%ni%`, 15

`all_ggplot_to_pptx`, 3  
`as_mon_numeric`, 3

`clean_levels`, 4  
`clean_names`, 5  
`clean_vec`, 5

`dput_levels`, 6

`excel_col (excel_names)`, 6  
`excel_names`, 6  
`excel_to_ncol (excel_names)`, 6

`find_name`, 7  
`from_excel_to_posixt`, 8

`gsub2`, 8

`is.01`, 9  
`is.12`, 9  
`is_full_figures`, 10  
`is_full_na`, 10  
`is_likert`, 11

`look_like_a_number`, 11

`make_unique`, 12

`ncol_to_excel (excel_names)`, 6

`peep`, 12

`replace_pattern`, 13

`save_as_csv`, 14  
`set_col_type`, 14