Package 'codecountR'

May 8, 2026

Title Counting Codes in a Text and Preparing Data for Analysis

Version 0.0.4.8

Description Data analysis often requires coding, especially when data are collected through interviews, observations, or questionnaires. As a result, code counting and data preparation are essential steps in the analysis process. Analysts may need to count the codes in a text (Tokenization, counting of pre-established codes, computing the co-occurrence matrix by line) and prepare the data (e.g., min-max normalization, Z-score, robust scaling, Box-Cox transformation, and non-parametric bootstrap). For the Box-Cox transformation (Box & Cox, 1964, <https://www.jstor.org/stable/2984418>), the optimal Lambda is determined using the log-likelihood method. Non-parametric bootstrap involves randomly sampling data with replacement. Two random number generators are also integrated: a Lehmer congruential generator for uniform distribution and a Box-Muller generator for normal distribution. Package for educational purposes.

License GPL-3

Encoding UTF-8

RoxygenNote 7.2.3

Imports stats

Suggests knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation no

Author Philippe Cohard [aut, cre]

Maintainer Philippe Cohard <p.cohard@laposte.net>

Repository CRAN

Date/Publication 2025-03-01 16:10:02 UTC

Contents

`analysCodesList` *analysCodesList*

Description

analysCodesList

Usage

```
analysCodesList(dataS, codesLis)
```

Arguments

`dataS`	a character
`codesLis`	a character

Value

a list

Examples

```
codes=list("@essai@","@test@")
data = "this is an example @essai@, a bit long @essai@ text"
Result=analysCodesList(data,codes)
Result
```

bootStrap *bootStrap*

Description

bootStrap

Usage

```
bootStrap(nameDframe, grpSize)
```

Arguments

nameDframe	a data.frame
grpSize	a number

Value

a matrix

Examples

```
j=c(10,14,56,30,58,78,99,1)
k=c(10,12,14,16,18,20,22,24)
x=data.frame(j,k)
res=bootStrap(x,5)
res
```

BoxAndCox *BoxAndCox*

Description

BoxAndCox

Usage

```
BoxAndCox(rawVect, minLambda)
```

Arguments

rawVect	a vector
minLambda	a number

Value

a list

Examples

```
vec=rlnorm(100, log(3), log(3))
BandC=BoxAndCox(vec, -3)
BandC
BAC=unlist(BandC$par)
BAC
rawVectBCFinal=unlist(subCalcBoxAndCox(vec, BandC$par))
```

BoxMullerGen *BoxMullerGen*

Description

BoxMullerGen

Usage

```
BoxMullerGen(r, s)
```

Arguments

r	a number
s	a number

Value

a vector

Examples

```
#with runif
v=BoxMullerGen(runif(1), runif(1))
print(v)

#with congruGen
seed = 123456789
X=c()
for(i in 1: 2) {
Z=congruGen(seed)
seed=Z$seedUpdate
X=append(X, Z$aleaNum)
}
#print(X)

N=BoxMullerGen(X[1], X[2])
print(N[1])
print(N[2])
```

codeCount *codeCount*

Description

codeCount

Usage

```
codeCount(dataSet, code)
```

Arguments

dataSet	a character
code	a character

Value

a number

Examples

```
data = "this is an example @essai@"
codeCount(data, "@essai@") #number of lines containing the chain
```

congruGen *congruGen*

Description

congruGen

Usage

```
congruGen(seed, a)
```

Arguments

seed	a number
a	a number

Value

a list

Examples

```
seed = 123456789
for(i in 1: 10) {
Z=congruGen(seed)
seed=Z$seedUpdate
num=Z$aleaNum
print(num)
}
```

---

cooc *cooc*

Description

cooc

Usage

```
cooc(lines, code1, code2)
```

Arguments

lines	character
code1	character
code2	character

Value

an integer

Examples

```
lines ="Companies can boost responsiveness @performance@ by digital @digital@."
code1 = "@performance@"
code2 = "@digital@"
res=cooc(lines, code1, code2)
print(res)
```

loadCodes *loadCodes*

Description

loadCodes

Usage

```
loadCodes(txtFile)
```

Arguments

txtFile	a character

Value

a list

Examples

```
theFile =system.file("codesList.txt", package = "codecountR")
data=loadCodes(theFile)
```

normMinMax *normMinMax*

Description

normMinMax

Usage

```
normMinMax(nameDframe)
```

Arguments

nameDframe	a data.frame

Value

a data.frame

Examples

```
j=c(10,14,56,30,58,78,99,1)
k=c(10,12,14,16,18,20,22,24)
x=data.frame(j,k)
xMinMax=normMinMax(x)
xMinMax
```

`robustScal` *robustScal*

Description

robustScal

Usage

```
robustScal(nameDframe)
```

Arguments

`nameDframe`	a data.frame

Value

a data.frame

Examples

```
j=c(10,14,56,30,58,78,99,1)
k=c(10,12,14,16,18,20,22,24)
x=data.frame(j,k)
xRsc=robustScal(x)
xRsc
```

`subCalcBoxAndCox` *subCalcBoxAndCox*

Description

subCalcBoxAndCox

Usage

```
subCalcBoxAndCox(sortedVect, actualLambda)
```

Arguments

sortedVect	a vector
actualLambda	a number

Value

a vector

Examples

```
vec=rlnorm(100, log(3), log(3))
BandC=subCalcBoxAndCox(vec, -3)
```

---

testPairs *testPairs*

Description

testPairs

Usage

```
testPairs(listCodes, lines)
```

Arguments

listCodes	character
lines	character

Value

a list

Examples

```
#Co-occurrences computed line by line in the file. Structure the file accordingly.
#Multiple identical pairs on one line count as one unit.
lines =c("Companies can boost responsiveness @performance@ by digital @digital@.",
  "softwares @digital@ may reduce response time @performance@ improving @satisfaction@.")
listCodes=c("@satisfaction@", "@digital@", "@performance@")
coocurences = testPairs(listCodes, lines)
print(coocurences$matrix)
#save to file
#nameFile = paste("CooccurrenceMatrix_",format(Sys.time(),"%d_%m_%Y-%Hh%Mm%Ss"),".csv",sep = "")
#write.csv(coocurences$matrix, nameFile, row.names = TRUE)
```

tokenization *tokenization*

Description

tokenization

Usage

```
tokenization(txtFile)
```

Arguments

txtFile	a character

Value

a list

Examples

```
theFile =system.file("ExText.txt", package = "codecountR")
data=tokenization(theFile)
```

verify *verify*

Description

verify

Usage

```
verify(lines, code1, code2)
```

Arguments

lines	character
code1	character
code2	character

Examples

```
lines ="Companies can boost responsiveness @performance@ by digital @digital@."
code1 = "@performance@"
code2 = "@digital@"
verify(lines,code1,code2)
```

zScore *zScore*

Description

zScore

Usage

```
zScore(nameDframe)
```

Arguments

nameDframe	a data.frame

Value

a data.frame

Examples

```
j=c(10,14,56,30,58,78,99,1)
k=c(10,12,14,16,18,20,22,24)
x=data.frame(j,k)
xZsc=zScore(x)
xZsc
```

Index