Package 'tfer'

October 14, 2022

Type Package

Title Forensic Glass Transfer Probabilities

Version 1.3

Date 2020-07-14

Author James Curran and TingYu Huang

Maintainer James Curran <j.curran@auckland.ac.nz>

Description Statistical interpretation of forensic glass transfer (Simulation of the probability distribution of recovered glass fragments).

License GPL-2

LazyLoad yes

Encoding UTF-8

RoxygenNote 7.1.1

NeedsCompilation no

Repository CRAN

Date/Publication 2020-07-15 09:40:02 UTC

R topics documented:

getParams *Extract Transfer and Persistence Parameters I*

Description

Displays input parameters and arguments passed to transfer.

Usage

```
getParams(tferObj)
```

Arguments

tferObj	An object of class transfer

Details

getParams is one of the two accessor functions for a transfer object.

Value

getParams returns a list of input parameters and their corresponding values.

Author(s)

TingYu Huang

See Also

transfer

Examples

```
library(tfer)

y = transfer()

getParams(y)
```

getValues	*Extract Transfer Values n* getValues *is a accessor function which returns the number of recovered glass fragments generated by* transfer.

Description

Extract Transfer Values n getValues is a accessor function which returns the number of recovered glass fragments generated by transfer.

Usage

```
getValues(tferObj)
```

Arguments

tferObj	An object of class tfer

Value

values returns a numeric vector of random variates.

Author(s)

TingYu Huang and James Curran

See Also

transfer

Examples

```
library(tfer)
y = transfer()
getValues(y)
```

plot.tfer *plot method for objects of transfer class*

Description

plot method for objects of transfer class

Usage

```
## S3 method for class 'tfer'
plot(
  x,
  ptype = c("density", "freq", "hist"),
  xlab = "n",
  main = "",
  col = "red",
  ...
)
```

Arguments

x	an object of class transfer
ptype,	one of "density", "freq", or "hist". "density" will give a barplot with probability on the y-axis, "frequency" will give a barplot with frequencies (counts) on the y-axis, and "hist" will produce a historgram with frequency (counts) on the y-axis. One-letter versions will also work, i.e. "d", "f" and "h". The original 0, 1, 2 will also work, but this usage is deprecated and will produce a warning.
xlab	the x-axis label, by default "n"
main	the plot title, empty by default
col	the colour of the bars in the plot, by default "red"
...	any other arguments to be passed to barplot or histogram

print.transfer print *method for transfer objects*

Description

Prints a summary of the simulation input parameters

Usage

```
## S3 method for class 'transfer'
print(x, ...)
```

Arguments

x	an object of class transfer
...	included for consistency but not used

`summary.transfer` `summary` *method for transfer objects*

Description

Prints a summary of the simulation input parameters

Usage

```
## S3 method for class 'transfer'
summary(object, ...)
```

Arguments

object	an object of class transfer
...	extra arguments passed to `summary.default`

Value

A `list` with three elements is returned invisibly:

parameters list containing all the simulation parameters

values a numeric vector of the simulated values

probability a named numeric vector giving the probability of recovering 0, 1, 2, ... fragments

---

`tprob` *Return a table of T probabilities for all observed values*

Description

Return a table of T probabilities for all observed values

Usage

```
tprob(tferObj, x)
```

Arguments

`tferObj`	an object of class `transfer`
`x`	an optional set of values which specify the desired T-terms. E.g. x = c(0,1,2) would return T0, T1, and T2 and so on. Negative values of x will cause the function to stop. Values of x which exceed those observed will be assigned a value of zero. The return values will be returned in ascending order regardless of the order of x (although I suppose I could preserve the order if someone really cares).

Value

A table of T probabilities, giving the probability that x fragments were recovered given they were transferred and persisted according to the other inputs of the model.

Examples

```
set.seed(123)
y = transfer()

tprob(y)
tprob(y, 55:120) ## max observed value is 113
```

`transfer` *Glass Transfer, Persistence and Recovery Probabilities*

Description

Construct a transfer object to simulate the number of glass fragments recovered given the conditions set by the user.

Usage

```
transfer(
  N = 10000,
  d = 0.5,
  deffect = TRUE,
  lambda = 120,
  Q = 0.05,
  l0 = 0.8,
  u0 = 0.9,
  lstar0 = 0.1,
  ustar0 = 0.15,
  lj = 0.45,
  uj = 0.7,
  lstarj = 0.05,
  ustarj = 0.1,
```

```
  lR = 0.5,
  uR = 0.7,
  lt = 1,
  ut = 2,
  r = 0.5,
  timeDist = c("negbin", "cnegbin", "uniform"),
  loop = FALSE
)
```

Arguments

N	Simulation size
d	The breaker's distance from the window
deffect	Distance effect. deffect = TRUE when distance effect exists. Otherwise deffect = FALSE.
lambda	The average number of glass fragments transferred to the breaker's clothing.
Q	Proportion of high persistence fragments.
l0	Lower bound on the percentage of fragments lost in the first hour
u0	Upper bound on the percentage of fragments lost in the first hour
lstar0	Lower bound on the percentage of high persistence fragments lost in the first hour
ustar0	Upper bound on the percentage of high persistence fragments lost in the first hour
lj	Lower bound on the percentage of fragments lost in the j'th hour
uj	Upper bound on the percentage of fragments lost in the j'th hour
lstarj	Lower bound on the percentage of high persistence fragments lost in the j'th hour
ustarj	Upper bound on the percentage of high persistence fragments lost in the j'th hour
lR	Lower bound on the percentage of fragments expected to be detected in the lab
uR	Upper bound on the percentage of fragments expected to be detected in the lab
lt	Lower bound on time between commission of crime and apprehension of suspect
ut	Upper bound on time between commission of crime and apprehension of suspect
r	Probability r in ti ~ NegBinom(t, r)
timeDist	the distribution for the random amount of time between the commission of the crime and the apprehension of the suspect. There are three choices "negbin", "cnegbin", and "uniform". Before talking about these it should be noted that if lt is equal to ut - then there is no randomness in this calculation. If lt does not equal ut, then the average of these two values is used in the two negative binomial options: "negbin" and "cnegbin". The difference betweeen them is that "cnegbin" is a constrained negative binomial where the allowable times are constrained to be between lt and ut. If "uniform" is selected, then a uniformly distributed random time between lt and ut is used in each iteration.

`loop` if `TRUE` an element by element version of the simulation is used, if `FALSE` then a (mostly) vectorised element version of the simulation is used. The results from the two methods appear to be almost identical - they won't be the same even with the same seed because of the way the random variates are generated. I (James) believe the vectorised version is faster and better. There was also a small mistake which has been corrected in that the initial set of persistent fragments was not being

Value

a `list` containing:

results The simulated values of recovered glass fragments

paramList Input parameters

The returned object has S3 class types tfer and transfer for backwards compatibility

Author(s)

James Curran and TingYu Huang

References

Curran, J. M., Hicks, T. N. & Buckleton, J. S. (2000). *Forensic interpretation of glass evidence*. Boca Raton, FL: CRC Press.

Curran, J. M., Triggs, C. M., Buckleton, J. S., Walsh, K. A. J. & Hicks T. N. (January, 1998). Assessing transfer probabilities in a Bayesian interpretation of forensic glass evidence. *Science & Justice*, *38*(1), 15-21.

Examples

```
library(tfer)

## create a transfer object using default arguments
y = transfer()

## probability table
probs = tprob(y)

## extract the probabilities of recovering 8 to 15
## glass fragments given the user-specified arguments
tprob(y, 8:15)

## produce a summary table for a transfer object
summary(y)

## barplot of probabilities (default)
plot(y)
plot(y)
```

```
## barplot of transfer frequencies
plot(y, ptype = "f")

## histogram
plot(y, ptype = "h")
```

Index