Package: centrifugeR (via r-universe)

September 15, 2024

Type Package

Title Non-Trivial Balance of Centrifuge Rotors

Version 0.1.8

Description Find the numbers of test tubes that can be balanced in centrifuge rotors and show various ways to load them. Refer to Pham (2020) <doi:10.31224/osf.io/4xs38> for more information on package functionality.

Depends R (>= 4.2.0)

Imports pracma (>= 2.4.2), shiny (>= 1.7.4), shinythemes (>= 1.2.0), grDevices, graphics, utils

URL https://phamdn.shinyapps.io/centrifugeR/

BugReports https://github.com/phamdn/centrifugeR/issues

License GPL-3

Encoding UTF-8

RoxygenNote 7.2.3

Repository https://phamdn.r-universe.dev

RemoteUrl https://github.com/phamdn/centrifuger

RemoteRef HEAD

RemoteSha db23c134641f2cf14e50ffdce00f1351865d0e99

Contents

`centrifugeR-package` *centrifugeR: Non-Trivial Balance of Centrifuge Rotors*

Description

Find the numbers of test tubes that can be balanced in centrifuge rotors and show various ways to load them. Refer to Pham (2020) doi:10.31224/osf.io/4xs38 for more information on package functionality.

Guidelines

centrifugeR helps obtain the perfect centrifuge balance. Call `rotor` to know how many tubes can be loaded into the rotor and see different ways to place these tubes. Alternatively, call `launch` to run the Shiny app. Also, call `rotorVerify` to verify the balance of pre-existing tube configurations. Call `rotorSpeed` for RPM/RCF conversion.

Copyright

Author(s)

Duy Nghia Pham <nghiapham@yandex.com>

`launch` *Run the Shiny App*

Description

Run the Shiny App

Usage

```
launch()
```

p2c *Convert Hole Positions to Coordinates*

Description

p2c returns the coordinates of rotor holes. This is not meant to be called directly.

Usage

```
p2c(x, y)
```

Arguments

x	an integer or a vector, the hole position(s).
y	an integer, the number of rotor holes.

Value

p2c returns a vector of coordinates.

pie2 *Draw a Rotor*

Description

pie2 was modified after pie. This is not meant to be called directly.

Usage

```
pie2(
  x,
  labels = names(x),
  edges = 200,
  radius = 0.8,
  clockwise = FALSE,
  init.angle = if (clockwise) 90 else 0,
  density = NULL,
  angle = 45,
  col = NULL,
  border = NULL,
  lty = NULL,
  main = NULL,
  panel = NULL,
  ...
)
```

Arguments

x	See pie for more details.
labels	See pie for more details.
edges	See pie for more details.
radius	See pie for more details.
clockwise	See pie for more details.
init.angle	See pie for more details.
density	See pie for more details.
angle	See pie for more details.
col	See pie for more details.
border	See pie for more details.
lty	See pie for more details.
main	See pie for more details.
panel	a letter for labeling panel.
...	See pie for more details.

Value

pie2 returns a figure.

---

rotor *Balance Tubes Using Higher-Order Symmetrical Configurations*

Description

rotor returns the numbers of tubes that can and cannot be loaded in a centrifuge rotor and optionally shows various ways to balance a certain number of tubes.

Usage

```
rotor(n, k = NULL, seed = 1, elapse = 1)
```

Arguments

n	an integer, the number of rotor holes.
k	an integer, the number of tubes (optional).
seed	an integer, the seed for random number generation. Setting a seed ensures the reproducibility of the result. See set.seed for more details.
elapse	an integer, the constrained time in seconds for random sampling.

Details

The number of rotor holes n ranges from 4 to 48.

Value

`rotor` returns a list with two components:

`check`	a list with three components:
	`n` the number of rotor holes.
	`valid` a vector containing the numbers of tubes that can be loaded.
	`invalid` a vector containing the numbers of tubes that cannot be loaded.
`load`	a list with three components:
	`k` the number of tubes.
	`decompose` a data frame showing different ways to decompose `k`.
	`hole` a data frame showing hole positions to load tubes.
	`visual` a list of rotor images showing hole positions to load tubes.

References

Sivek G. On vanishing sums of distinct roots of unity. Integers. 2010;10(3):365-8.

Peil O, Hauryliuk V. A new spin on spinning your samples: balancing rotors in a non-trivial manner. arXiv preprint arXiv:1004.3671. 2010 Apr 21.

See Also

`rotorVerify` for verifying the balance of pre-existing tube configurations.

Examples

```
rotor(30, 7)
```

---

rotorCheck *Check Centrifuge Rotors*

Description

`rotorCheck` returns the numbers of tubes that can and cannot be loaded in a single operation.

Usage

```
rotorCheck(n, k = NULL)
```

Arguments

`n`	an integer, the number of rotor buckets.
`k`	an integer, the number of tubes.

Details

The number of rotor buckets `n` ranges from `4` to `48`.

If `k` is specified, `rotorCheck` will check whether the input number of tubes can be loaded or not.

Value

`rotorCheck` returns a list with two components:

`valid`	a vector containing the numbers of tubes that can be loaded.
`invalid`	a vector containing the numbers of tubes that cannot be loaded.

References

Sivek G. On vanishing sums of distinct roots of unity. Integers. 2010;10(3):365-8.

See Also

`rotorEqual` for balancing tubes of equal mass and `rotorUnequal` for balancing tubes of unequal mass.

`rotorEqual` *Balance Tubes of Equal Mass*

Description

`rotorEqual` returns the positions of rotor buckets that must be loaded or empty to balance tubes of equal mass.

Usage

```
rotorEqual(n, k, seed = 2019)
```

Arguments

`n`	an integer, the number of rotor buckets.
`k`	an integer, the number of tubes.
`seed`	an integer, the seed for random number generation. Setting a seed ensures the reproducibility of the result. See `set.seed` for more details.

Details

The number of rotor buckets `n` ranges from `4` to `48`. The number of tubes `k` must be greater than `0` and smaller than the number of rotor buckets `n`.

Value

`rotorEqual` returns a list with two components:

`loaded`	a vector containing the positions of rotor buckets that must be loaded.
`empty`	a vector containing the positions of rotor buckets that must be empty.

`rotorEqual` also plots a schematic diagram of the centrifuge rotor.

References

Sivek G. On vanishing sums of distinct roots of unity. Integers. 2010;10(3):365-8.

Peil O, Hauryliuk V. A new spin on spinning your samples: balancing rotors in a non-trivial manner. arXiv preprint arXiv:1004.3671. 2010 Apr 21.

See Also

`rotorCheck` for checking centrifuge rotors and `rotorUnequal` for balancing tubes of unequal mass.

`rotorSpeed` *Calculate RPM/RCF*

Description

`rotorSpeed` converts rotational speed to relative centrifuge force and vice versa.

Usage

```
rotorSpeed(radius, value, type)
```

Arguments

`radius`	a numeric, the centrifugal radius in millimeters.
`value`	a numeric, the rotational speed in revolutions per minute or the relative centrifuge force in × g.
`type`	the type of the above-mentioned `value`, "rpm" for rotational speed or "rcf" for relative centrifuge force.

Value

`rotorSpeed` returns a numeric that is the rotational speed in revolutions per minute or the relative centrifuge force in × g.

References

Rickwood D, editor. Centrifugation: a practical approach. London: Information Retrieval Ltd; 1978. 224 p.

Examples

```
rotorSpeed(100, 12000, "rpm")
rotorSpeed(100, 6000, "rcf")
```

`rotorUnequal` *Balance Tubes of Unequal Mass*

Description

`rotorUnequal` returns the required masses and the positions of tubes of unequal initial mass.

Usage

```
rotorUnequal(n, mass = NULL, seed = 2019)
```

Arguments

`n`	an integer, the number of rotor buckets.
`mass`	a numeric vector with optional `names` attribute, the masses (and optional names) of tubes.
`seed`	an integer, the seed for random number generation. Setting a seed ensures the reproducibility of the result. See `set.seed` for more details.

Details

The number of rotor buckets `n` ranges from `4` to `48`. The number of tubes (i.e. `length(mass)`) should not be greater than the number of rotor buckets `n`.

If `mass` is not specified, the names and the masses of tubes must then be taken from the keyboard. In case `mass` has no `names` attribute, tubes will be named automatically (i.e. `S1, S2, S3,` etc.).

Value

`rotorUnequal` returns a data frame with three columns:

`initial`	a vector containing the initial masses of tubes.
`required`	a vector containing the required masses of tubes.
`position`	a vector containing the bucket positions of tubes.

`rotorUnequal` also plots a schematic diagram of the centrifuge rotor.

References

Sivek G. On vanishing sums of distinct roots of unity. Integers. 2010;10(3):365-8.

Peil O, Hauryliuk V. A new spin on spinning your samples: balancing rotors in a non-trivial manner. arXiv preprint arXiv:1004.3671. 2010 Apr 21.

See Also

rotorCheck for checking centrifuge rotors and rotorEqual for balancing tubes of equal mass.

Examples

```
# Call the function then input the names and the masses of tubes
rotorUnequal(30)
liver
10.05
gill
9.68
muscle
9.88

# Prepare the masses of tubes then call the function
samples <- round(rnorm(19, mean = 10, sd = 0.5), 2)
rotorUnequal(30, samples)

# Prepare the masses and the names of tubes then call the function
small.samples <- c(10.05, 9.68, 9.88)
names(small.samples) <- c("liver", "gill", "muscle")
rotorUnequal(30, small.samples)
```

---

rotorVerify *Verify Centrifuge Balance*

Description

rotorVerify checks whether the rotor is balanced given the positions of tubes of equal mass.

Usage

```
rotorVerify(n, pos)
```

Arguments

n	an integer, the number of rotor buckets.
pos	an integer vector, the positions of tubes.

Details

The number of rotor buckets n ranges from 4 to 48. The positions of tubes pos ranges from 1 to n.

Value

rotorVerify returns 1 if the rotor is balanced and 0 if the rotor is unbalanced.

References

Johnsson M. Balancing a centrifuge. R-bloggers. 2016. Available from: https://www.r-bloggers.com/2016/06/balancing-a-centrifuge/.

See Also

rotorCheck for checking centrifuge rotors.

Examples

```
rotorVerify(30, c(10, 20, 30))
rotorVerify(30, c(1, 11, 21, 4, 28))
```

Index