Package: stringArt (via r-universe)

May 29, 2026

Title Tools for Generating String Art Figures

Version 0.1.0

Description Provides tools to generate, visualize, and audit geometric string art figures for mathematics teaching. The package includes functions for circular, cardioid-like, elliptical, triangular, polygonal, star, parabolic, net-based, radial, hexagonal, lotus-like, rose-like, spiral, Lissajous, grid-based, decimal, and contour-based string art patterns. Each function returns peg coordinates, connection tables, total string length, audit information, and metadata, supporting educational applications in geometry, analytic geometry, modular arithmetic, trigonometry, rational numbers, and visual mathematics.

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

URL https://github.com/fsbmat-ufv/stringArt, https://fsbmat-ufv.github.io/stringArt/

BugReports https://github.com/fsbmat-ufv/stringArt/issues

Depends R (>= 4.1)

Imports graphics

Suggests shiny, shinydashboard, colourpicker, DT, markdown, knitr, rmarkdown, testthat (>= 3.0.0), pkgdown

VignetteBuilder knitr

Config/testthat/edition 3

Repository https://fsbmat-ufv.r-universe.dev

Date/Publication 2026-05-21 02:24:38 UTC

RemoteUrl https://github.com/fsbmat-ufv/stringart

RemoteRef HEAD

RemoteSha 1c7985fb48aea99e09892bbb2528fbf5a5cc4d0f

Contents

`run_stringArt_app` *Launch the stringArt Shiny app*

Description

Opens the interactive Shiny application included with the package.

Usage

```
run_stringArt_app(launch.browser = TRUE, ...)
```

Arguments

`launch.browser`	Logical. If `TRUE`, the application is opened in the user's default browser.
`...`	Additional arguments passed to `shiny::runApp()`.

Value

Invisibly returns the result of `shiny::runApp()`.

Examples

```
if (interactive()) {
  run_stringArt_app()
}
```

`stcardioid` *Generate a circular string art pattern with a cardioid effect*

Description

`stcardioid()` generates a circular string art pattern by placing equally spaced pegs on a circle and connecting each peg to another peg according to a multiplicative modular rule.

Usage

```
stcardioid(
  n = 120,
  k = 2,
  col = "antiquewhite",
  lwd = 0.8,
  plot = TRUE,
  show_points = FALSE,
  show_labels = FALSE,
  verbose = FALSE,
  r = 1,
  rotate = 0,
  show_strings = TRUE,
  template = FALSE,
  point_col = "darkorange2",
  point_cex = 0.8,
  point_pch = 21,
  point_bg = "white",
  label_cex = 0.7,
  label_col = "black",
  border_col = "goldenrod3",
  border_lwd = 1.2,
  bg = "white",
  main = NULL
)
```

Arguments

`n`	Integer. Number of pegs placed on the circle. Must be at least 3.
`k`	Integer. Multiplication factor used in the modular connection rule. Must satisfy `1 <= k <= n - 1`.
`col`	String color passed to `graphics::segments()`.
`lwd`	Positive number. Line width used to draw the strings.
`plot`	Logical. If `TRUE`, draws the figure.
`show_points`	Logical. If `TRUE`, draws the pegs.
`show_labels`	Logical. If `TRUE`, draws peg labels.

verbose	Logical. If TRUE, prints a short audit to the console.
r	Positive number. Radius of the circle.
rotate	Numeric. Rotation angle in radians applied to the whole figure.
show_strings	Logical. If TRUE, draws the string connections.
template	Logical. If TRUE, draws only the peg template, without string connections. This is equivalent to setting show_strings = FALSE and show_points = TRUE.
point_col	Peg border color.
point_cex	Positive number. Peg size.
point_pch	Plotting symbol used for pegs.
point_bg	Peg background color when applicable.
label_cex	Positive number. Label size.
label_col	Label color.
border_col	Circle border color.
border_lwd	Positive number. Circle border line width.
bg	Plot background color.
main	Optional plot title. If NULL, no title is displayed.

Details

The pegs are placed on a circle of radius r, centered at the origin. Pegs are indexed from 1 to n in counterclockwise order, starting at (r, 0), after applying the optional rotation angle rotate.

The multiplicative modular connection rule is:

to = ((k * (from - 1)) %% n) + 1.

For k = 2, this rule produces the classical circular multiplication-table pattern commonly associated with a cardioid-like envelope.

Value

Invisibly returns a list of class stringart_result with:

pegs A data.frame with columns index, x, and y.

connections A data.frame with columns connection_index, from, to, x_from, y_from, x_to, y_to, and length.

total_length Total string length.

audit A character vector with audit information.

meta A list with construction metadata.

Examples

```
stcardioid()

res <- stcardioid(plot = FALSE)
head(res$pegs)
head(res$connections)
res$total_length

stcardioid(n = 80, k = 2, col = "steelblue", lwd = 0.8)
stcardioid(n = 24, k = 5, show_points = TRUE, show_labels = TRUE)
stcardioid(template = TRUE)
```

---

stcircle *Circular String Art*

Description

stcircle() creates a circular String Art figure by placing equally spaced pegs on a circle and connecting each peg to another peg using an additive modular step.

Usage

```
stcircle(
  n = 30,
  k = 5,
  col = "blue",
  lwd = 1,
  plot = TRUE,
  show_points = TRUE,
  show_labels = FALSE,
  verbose = FALSE,
  r = 1,
  show_strings = TRUE,
  template = FALSE,
  point_col = "black",
  point_cex = 0.8,
  point_pch = 19,
  point_bg = "white",
  label_cex = 0.7,
  label_col = "black",
  border_col = "grey50",
  border_lwd = 1,
  main = NULL
)
```

Arguments

n	Integer. Number of pegs on the circle. Defaults to 30.
k	Integer. Additive modular step used to define the connections. Defaults to 5.
col	String color used to draw the segments. Defaults to "blue".
lwd	Positive number. Line width of the string segments. Defaults to 1.
plot	Logical. If TRUE, the figure is drawn using base R graphics.
show_points	Logical. If TRUE, the pegs are shown.
show_labels	Logical. If TRUE, peg labels are shown.
verbose	Logical. If TRUE, an audit summary is printed to the console.
r	Positive number. Radius of the circle. Defaults to 1.
show_strings	Logical. If TRUE, string segments are drawn.
template	Logical. If TRUE, only the peg template is drawn. This sets show_strings = FALSE and show_points = TRUE internally.
point_col	Color of the pegs.
point_cex	Positive number. Size of the pegs.
point_pch	Plotting symbol used for the pegs.
point_bg	Background color of the pegs when the plotting symbol allows filling.
label_cex	Positive number. Size of the peg labels.
label_col	Color of the peg labels.
border_col	Color of the circular border.
border_lwd	Positive number. Line width of the circular border.
main	Character string. Plot title. If NULL, a default title is used.

Details

Pegs are numbered from 1 to n counterclockwise, starting at (r, 0). For each peg i, the connected peg is defined by

$$j = ((i + k - 1) \bmod n) + 1.$$

When gcd(n, k) = 1, the rule creates a single cycle passing through all pegs. When gcd(n, k) > 1, the figure is decomposed into independent cycles.

Value

Invisibly returns a list with the following elements:

pegs A data frame with columns index, x, and y.

connections A data frame with columns connection_index, from, to, x_from, y_from, x_to, y_to, and length.

total_length Total string length.

audit Character vector with an audit summary.

meta List with metadata about the construction.

Examples

```
stcircle()

res <- stcircle(plot = FALSE)
res$total_length
head(res$connections)

stcircle(n = 24, k = 7, col = "firebrick", lwd = 1.2,
         show_points = TRUE, show_labels = TRUE)

stcircle(n = 24, k = 7, template = TRUE)
```

`stdecimal` *Generate a string art path from a rational decimal expansion*

Description

`stdecimal()` places digit pegs on a circle and connects consecutive digits in the expansion of a rational number. By default, the function uses base 10, so the pegs are labeled from 0 to 9.

Usage

```
stdecimal(
  numerator = 1,
  denominator = 7,
  n = 10,
  k = 2,
  col = "blue",
  lwd = 1,
  plot = TRUE,
  show_points = TRUE,
  show_labels = TRUE,
  verbose = FALSE,
  radius = 1,
  rotate = pi/2,
  include_integer_part = TRUE,
  show_strings = TRUE,
  template = FALSE,
  point_col = "black",
  point_cex = 0.9,
  point_pch = 21,
  point_bg = "white",
  label_cex = 0.8,
  label_col = "black",
  border_col = "grey50",
  border_lwd = 1,
```

```
  bg = "white",
  main = NULL
)
```

Arguments

numerator	Integer. Numerator of the rational number.
denominator	Integer. Denominator of the rational number. Must be nonzero.
n	Integer. Numeral base and number of digit pegs. Default is 10.
k	Integer. Number of repetitions of the repetend shown in the plot when the expansion is repeating. Must be at least 1.
col	String color passed to graphics::segments().
lwd	Positive number. Line width used to draw the digit path.
plot	Logical. If TRUE, draws the figure.
show_points	Logical. If TRUE, draws the digit pegs.
show_labels	Logical. If TRUE, draws digit labels.
verbose	Logical. If TRUE, prints a short audit to the console.
radius	Positive number. Circle radius.
rotate	Numeric. Rotation angle in radians applied to the digit circle.
include_integer_part	Logical. If TRUE, includes the integer part of the rational number in the displayed digit sequence.
show_strings	Logical. If TRUE, draws the digit connections.
template	Logical. If TRUE, draws only the digit template, without connections. This is equivalent to setting show_strings = FALSE and show_points = TRUE.
point_col	Peg color.
point_cex	Positive number. Peg size.
point_pch	Plotting symbol used for pegs.
point_bg	Peg background color when applicable.
label_cex	Positive number. Label size.
label_col	Label color.
border_col	Border color of the digit circle.
border_lwd	Positive number. Border line width.
bg	Plot background color.
main	Optional plot title. If NULL, no title is displayed.

Details

The function computes the base-n expansion of numerator / denominator using exact long division. When the expansion is repeating, the repetend is displayed k times after the preperiod.

For the default setting n = 10, the function is especially useful for exploring decimal expansions, repeating decimals, periodicity, and patterns in rational numbers.

Value

Invisibly returns a list of class `stringart_result` with:

pegs A `data.frame` with columns `index`, `x`, `y`, and `digit`.

connections A `data.frame` with columns `connection_index`, `from`, `to`, `x_from`, `y_from`, `x_to`, `y_to`, `length`, `digit_from`, `digit_to`, and `position`.

total_length Total string length.

audit A character vector with audit information.

meta A list with construction metadata.

Examples

```
stdecimal()
stdecimal(1, 7)
stdecimal(1, 13)
stdecimal(22, 7)
stdecimal(template = TRUE)
```

---

stellipse *Generate an elliptical string art pattern*

Description

`stellipse()` generates a string art pattern by placing equally spaced pegs along an ellipse and connecting each peg to another peg according to an additive modular rule.

Usage

```
stellipse(
  n = 30,
  k = 5,
  col = "blue",
  lwd = 1,
  plot = TRUE,
  show_points = TRUE,
  show_labels = FALSE,
  verbose = FALSE,
  a = 2,
  b = 1,
  rotate = 0,
  show_strings = TRUE,
  template = FALSE,
  point_col = "black",
  point_cex = 0.8,
  point_pch = 19,
```

```
  point_bg = "white",
  label_cex = 0.7,
  label_col = "black",
  border_col = "grey50",
  border_lwd = 1,
  bg = "white",
  main = NULL
)
```

Arguments

n	Integer. Number of pegs placed on the ellipse. Must be at least 3.
k	Integer. Step used in the modular connection rule. Must satisfy 1 <= k <= n - 1.
col	String color passed to graphics::segments().
lwd	Positive number. Line width used to draw the strings.
plot	Logical. If TRUE, draws the figure.
show_points	Logical. If TRUE, draws the pegs.
show_labels	Logical. If TRUE, draws peg labels.
verbose	Logical. If TRUE, prints a short audit to the console.
a	Positive number. Semi-major horizontal axis of the ellipse.
b	Positive number. Semi-minor vertical axis of the ellipse.
rotate	Numeric. Rotation angle in radians applied to the whole figure.
show_strings	Logical. If TRUE, draws the string connections.
template	Logical. If TRUE, draws only the peg template, without string connections. This is equivalent to setting show_strings = FALSE and show_points = TRUE.
point_col	Peg color.
point_cex	Positive number. Peg size.
point_pch	Plotting symbol used for pegs.
point_bg	Peg background color when applicable.
label_cex	Positive number. Label size.
label_col	Label color.
border_col	Ellipse border color.
border_lwd	Positive number. Ellipse border line width.
bg	Plot background color.
main	Optional plot title. If NULL, no title is displayed.

Details

The pegs are placed along the parametric ellipse x = a * cos(theta) and y = b * sin(theta), centered at the origin. Pegs are indexed from 1 to n in counterclockwise order, starting at (a, 0), after applying the optional rotation angle rotate.

The additive modular connection rule is:

to = ((from + k - 1) %% n) + 1.

When gcd(n, k) = 1, this rule generates a single cycle passing through all pegs. When gcd(n, k) > 1, the figure decomposes into independent cycles.

Value

Invisibly returns a list of class `stringart_result` with:

pegs A `data.frame` with columns `index`, `x`, and `y`.

connections A `data.frame` with columns `connection_index`, `from`, `to`, `x_from`, `y_from`, `x_to`, `y_to`, and `length`.

total_length Total string length.

audit A character vector with audit information.

meta A list with construction metadata.

Examples

```
stellipse()

res <- stellipse(plot = FALSE)
head(res$pegs)
head(res$connections)
res$total_length

stellipse(n = 40, k = 7, a = 2.5, b = 1.2, col = "purple", lwd = 1)
stellipse(n = 24, k = 5, show_points = TRUE, show_labels = TRUE)
stellipse(template = TRUE)
```

---

stgrid *Generate string art on a rectangular grid boundary*

Description

`stgrid()` generates string art by placing pegs uniformly along the boundary of a rectangle and connecting them using an additive modular rule.

Usage

```
stgrid(
  n = 60,
  k = 7,
  col = "blue",
  lwd = 1,
  plot = TRUE,
  show_points = TRUE,
  show_labels = FALSE,
  verbose = FALSE,
  width = 2,
  height = 1,
  rotate = 0,
  show_strings = TRUE,
```

```
  template = FALSE,
  border_col = "grey50",
  border_lwd = 1,
  point_col = "black",
  point_cex = 0.8,
  point_pch = 19,
  point_bg = "white",
  label_cex = 0.7,
  label_col = "black",
  bg = "white",
  main = NULL
)
```

Arguments

n	Integer. Number of pegs placed on the rectangle boundary. Must be at least 4.
k	Integer. Additive modular step used in the connection rule. Must satisfy 1 <= k <= n - 1.
col	String color passed to graphics::segments().
lwd	Positive number. Line width used to draw the strings.
plot	Logical. If TRUE, draws the figure.
show_points	Logical. If TRUE, draws the pegs.
show_labels	Logical. If TRUE, draws peg labels.
verbose	Logical. If TRUE, prints a short audit to the console.
width	Positive number. Rectangle width.
height	Positive number. Rectangle height.
rotate	Numeric. Rotation angle in radians.
show_strings	Logical. If TRUE, draws the string connections.
template	Logical. If TRUE, draws only the peg template.
border_col	Rectangle border color.
border_lwd	Positive number. Border line width.
point_col	Peg color.
point_cex	Positive number. Peg size.
point_pch	Plotting symbol used for pegs.
point_bg	Peg background color when applicable.
label_cex	Positive number. Label size.
label_col	Label color.
bg	Plot background color.
main	Optional plot title.

Details

Pegs are distributed uniformly along the rectangle boundary and connected using the additive modular rule to = ((from + k - 1) %% n) + 1.

Value

Invisibly returns a list of class `stringart_result`.

Examples

```
stgrid()
stgrid(width = 2, height = 1)
stgrid(template = TRUE)
```

`sthexaflower` *Generate a hexagonal flower string art pattern*

Description

`sthexaflower()` generates a string art pattern based on three concentric hexagonal peg circuits and one central peg. The construction is fully reproducible and returns both the peg coordinates and the connection table.

Usage

```
sthexaflower(
  n = 24,
  k = 5,
 col = c("black", "forestgreen", "darkorange", "deepskyblue4", "firebrick", "purple"),
  lwd = 1,
  plot = TRUE,
  show_points = TRUE,
  show_labels = FALSE,
  verbose = FALSE,
  r = 1,
  scale_mid = 0.72,
  scale_inner = 0.42,
  offset_mid = 0,
  offset_inner = 0,
  rotate = 0,
  show_strings = TRUE,
  template = FALSE,
  point_col = "black",
  point_cex = 0.8,
  point_pch = 19,
  point_bg = "white",
  label_cex = 0.7,
  label_col = "black",
  border_col = "grey50",
  border_lwd = 1,
  bg = "white",
```

```
  main = NULL
)
```

Arguments

n	Integer. Number of pegs in each hexagonal circuit. Must be a multiple of 6 and at least 6.
k	Integer. Step used in the local modular connection rule. Must satisfy 1 <= k <= n - 1.
col	String color passed to graphics::segments(). It may have length 1 or 6. If length 6, colors are used by sector.
lwd	Positive number. Line width used to draw the strings.
plot	Logical. If TRUE, draws the figure.
show_points	Logical. If TRUE, draws the pegs.
show_labels	Logical. If TRUE, draws peg labels.
verbose	Logical. If TRUE, prints a short audit to the console.
r	Positive number. Radius of the outer hexagonal circuit.
scale_mid	Positive number in (0, 1). Scale of the middle hexagonal circuit relative to the outer circuit.
scale_inner	Positive number in (0, scale_mid). Scale of the inner hexagonal circuit relative to the outer circuit.
offset_mid	Numeric value in [0, 1). Discrete relative offset applied to the middle circuit along the peg sequence.
offset_inner	Numeric value in [0, 1). Discrete relative offset applied to the inner circuit along the peg sequence.
rotate	Numeric. Rotation angle in radians applied to the whole figure.
show_strings	Logical. If TRUE, draws the string connections.
template	Logical. If TRUE, draws only the peg template, without string connections. This is equivalent to setting show_strings = FALSE and show_points = TRUE.
point_col	Peg color.
point_cex	Positive number. Peg size.
point_pch	Plotting symbol used for pegs.
point_bg	Peg background color when applicable.
label_cex	Positive number. Label size.
label_col	Label color.
border_col	Hexagonal border color.
border_lwd	Positive number. Hexagonal border line width.
bg	Plot background color.
main	Optional plot title. If NULL, no title is displayed.

Details

The function builds three concentric hexagonal circuits with `n` pegs each and one central peg.

The peg table contains the columns `index`, `x`, `y`, `group`, `layer`, and `local_index`.

The construction uses four connection blocks:

- `outer_border`: consecutive connections on the outer hexagon.
- `outer_to_middle`: connections from the outer circuit to the middle circuit.
- `middle_to_inner`: connections from the middle circuit to the inner circuit.
- `vertices_to_center`: connections from the outer vertices to the central peg.

The local additive modular rule used in the two radial blocks is

`to_local = ((from_local + k - 1) %% n) + 1`.

Value

Invisibly returns a list of class `stringart_result` with:

pegs A `data.frame` with peg coordinates and metadata.

connections A `data.frame` with columns `connection_index`, `from`, `to`, `x_from`, `y_from`, `x_to`, `y_to`, `length`, `block`, and `sector`.

total_length Total string length.

audit A character vector with audit information.

meta A list with construction metadata.

Examples

```
sthexaflower()

res <- sthexaflower(plot = FALSE)
head(res$pegs)
head(res$connections)
res$total_length

sthexaflower(n = 30, k = 7, col = "steelblue", lwd = 0.8)
sthexaflower(n = 24, k = 5, show_points = TRUE, show_labels = TRUE)
sthexaflower(template = TRUE)
```

`stlissajous` *Generate a Lissajous string art pattern*

Description

`stlissajous()` generates a string art figure from pegs sampled on a Lissajous curve.

Usage

```
stlissajous(
  n = 300,
  k = 19,
  col = "purple",
  lwd = 0.7,
  plot = TRUE,
  show_points = FALSE,
  show_labels = FALSE,
  verbose = FALSE,
  a = 3,
  b = 2,
  phase = pi/2,
  amplitude_x = 1,
  amplitude_y = 1,
  rotate = 0,
  show_strings = TRUE,
  template = FALSE,
  draw_curve = TRUE,
  border_col = "grey50",
  border_lwd = 1,
  point_col = "black",
  point_cex = 0.5,
  point_pch = 19,
  point_bg = "white",
  label_cex = 0.6,
  label_col = "black",
  bg = "white",
  main = NULL
)
```

Arguments

`n`	Integer. Number of pegs sampled on the curve. Must be at least 3.
`k`	Integer. Additive modular step used in the connection rule. Must satisfy `1 <= k <= n - 1`.
`col`	String color passed to `graphics::segments()`.
`lwd`	Positive number. Line width used to draw the strings.
`plot`	Logical. If `TRUE`, draws the figure.

show_points	Logical. If TRUE, draws the pegs.
show_labels	Logical. If TRUE, draws peg labels.
verbose	Logical. If TRUE, prints a short audit to the console.
a	Positive integer. Frequency in the x-coordinate.
b	Positive integer. Frequency in the y-coordinate.
phase	Numeric. Phase shift in radians.
amplitude_x	Positive number. Horizontal amplitude.
amplitude_y	Positive number. Vertical amplitude.
rotate	Numeric. Rotation angle in radians.
show_strings	Logical. If TRUE, draws the string connections.
template	Logical. If TRUE, draws only the peg template.
draw_curve	Logical. If TRUE, draws the underlying Lissajous curve.
border_col	Curve color.
border_lwd	Positive number. Curve line width.
point_col	Peg color.
point_cex	Positive number. Peg size.
point_pch	Plotting symbol used for pegs.
point_bg	Peg background color when applicable.
label_cex	Positive number. Label size.
label_col	Label color.
bg	Plot background color.
main	Optional plot title.

Details

The curve is given by

`x(t) = amplitude_x * sin(a * t + phase)` and `y(t) = amplitude_y * sin(b * t)`.

Value

Invisibly returns a list of class `stringart_result`.

Examples

```
stlissajous()
stlissajous(a = 3, b = 2)
stlissajous(a = 5, b = 4, phase = pi / 3)
stlissajous(template = TRUE)
```

stlotus *Generate a lotus-like string art pattern*

Description

`stlotus()` generates a stylized lotus-like string art figure by combining one outer circular module, one central circular module, and several petal modules arranged around the center.

Usage

```
stlotus(
  n = 40,
  k = 11,
  col = "deeppink4",
  lwd = 0.8,
  plot = TRUE,
  show_points = FALSE,
  show_labels = FALSE,
  verbose = FALSE,
  petals = 5,
  outer_radius = 1,
  petal_radius = 0.34,
  petal_center_radius = 0.34,
  inner_radius = 0.18,
  rotate = 0,
  show_strings = TRUE,
  template = FALSE,
  point_col = "black",
  point_cex = 0.6,
  point_pch = 19,
  point_bg = "white",
  label_cex = 0.6,
  label_col = "black",
  border_col = "grey50",
  border_lwd = 1,
  bg = "white",
  main = NULL
)
```

Arguments

`n`	Integer. Number of pegs in each circular module. Must be at least 3.
`k`	Integer. Additive modular step used in each module. Must satisfy `1 <= k <= n - 1`.
`col`	String color passed to `graphics::segments()`. It may have length 1 or `petals + 2`. If length is 1, the same color is used for all modules.
`lwd`	Positive number. Line width used to draw the strings.

plot	Logical. If TRUE, draws the figure.
show_points	Logical. If TRUE, draws the pegs.
show_labels	Logical. If TRUE, draws peg labels.
verbose	Logical. If TRUE, prints a short audit to the console.
petals	Integer. Number of petal modules around the center.
outer_radius	Positive number. Radius of the outer circular module.
petal_radius	Positive number. Radius of each petal module.
petal_center_radius	Positive number. Distance from the origin to each petal center.
inner_radius	Positive number. Radius of the central circular module.
rotate	Numeric. Rotation angle in radians applied to the whole figure.
show_strings	Logical. If TRUE, draws the string connections.
template	Logical. If TRUE, draws only the peg template, without string connections. This is equivalent to setting show_strings = FALSE and show_points = TRUE.
point_col	Peg color.
point_cex	Positive number. Peg size.
point_pch	Plotting symbol used for pegs.
point_bg	Peg background color when applicable.
label_cex	Positive number. Label size.
label_col	Label color.
border_col	Border color used for the module outlines.
border_lwd	Positive number. Border line width.
bg	Plot background color.
main	Optional plot title. If NULL, no title is displayed.

Details

The figure is built from petals + 2 circular modules:

- one outer circular module;
- petals petal modules;
- one central circular module.

Each module contains n equally spaced pegs and uses the same additive modular rule:

to_local = ((from_local + k - 1) %% n) + 1.

The final figure is obtained by superimposing all module connections.

Value

Invisibly returns a list of class `stringart_result` with:

pegs A `data.frame` with peg coordinates and metadata.

connections A `data.frame` with columns `connection_index`, `from`, `to`, `x_from`, `y_from`, `x_to`, `y_to`, `length`, `module`, `local_from`, and `local_to`.

total_length Total string length.

audit A character vector with audit information.

meta A list with construction metadata.

Examples

```
stlotus()

res <- stlotus(plot = FALSE)
head(res$pegs)
head(res$connections)
res$total_length

stlotus(n = 50, k = 13, col = "deeppink4", lwd = 0.8)
stlotus(show_points = TRUE, show_labels = TRUE)
stlotus(template = TRUE)
```

---

stnet *Generate a string art net from two rays*

Description

`stnet()` generates a string art net by placing pegs on two rays that share a common vertex and connecting corresponding peg positions. The construction generalizes the classical parabolic string art envelope by allowing the angle between the two supporting rays to vary.

Usage

```
stnet(
  n = 30,
  k = 1,
  col = "blue",
  lwd = 1,
  plot = TRUE,
  show_points = TRUE,
  show_labels = FALSE,
  verbose = FALSE,
  length1 = 1,
  length2 = 1,
  angle = pi/2,
```

```
  rotate = 0,
  show_strings = TRUE,
  template = FALSE,
  show_envelope = FALSE,
  envelope_col = "red",
  envelope_lwd = 1,
  envelope_lty = 2,
  point_col = "black",
  point_cex = 0.8,
  point_pch = 19,
  point_bg = "white",
  label_cex = 0.7,
  label_col = "black",
  border_col = "grey50",
  border_lwd = 1,
  bg = "white",
  main = NULL
)
```

Arguments

n	Integer. Number of pegs placed on each ray. Must be at least 3.
k	Integer. Number of shifted sweeps used in the construction. Must satisfy `1 <= k <= n - 1`. The classical net is obtained with `k = 1`.
col	String color passed to `graphics::segments()`.
lwd	Positive number. Line width used to draw the strings.
plot	Logical. If `TRUE`, draws the figure.
show_points	Logical. If `TRUE`, draws the pegs.
show_labels	Logical. If `TRUE`, draws peg labels.
verbose	Logical. If `TRUE`, prints a short audit to the console.
length1	Positive number. Length of the first ray.
length2	Positive number. Length of the second ray.
angle	Numeric. Angle in radians from the first ray to the second ray. Must not be a multiple of `pi`.
rotate	Numeric. Rotation angle in radians applied to the whole net.
show_strings	Logical. If `TRUE`, draws the string connections.
template	Logical. If `TRUE`, draws only the peg template, without string connections. This is equivalent to setting `show_strings = FALSE` and `show_points = TRUE`.
show_envelope	Logical. If `TRUE`, draws the theoretical envelope of the basic construction.
envelope_col	Color used for the theoretical envelope.
envelope_lwd	Positive number. Line width used for the envelope.
envelope_lty	Line type used for the envelope.
point_col	Peg color.

point_cex	Positive number. Peg size.
point_pch	Plotting symbol used for pegs.
point_bg	Peg background color when applicable.
label_cex	Positive number. Label size.
label_col	Label color.
border_col	Ray color.
border_lwd	Positive number. Ray line width.
bg	Plot background color.
main	Optional plot title. If NULL, no title is displayed.

Details

The construction uses two rays with a common vertex. Pegs are placed uniformly along the first ray from the common vertex to the first endpoint and along the second ray from the second endpoint back to the common vertex.

For k = 1, the peg at local position i on the first ray is connected to the peg at local position i on the second ray. In oblique coordinates determined by the two rays, the theoretical envelope is given by C(t) = t^2 A + (1 - t)^2 B,

where A and B are the endpoints of the two rays and 0 <= t <= 1.

For k > 1, the function adds shifted sweeps of the same construction, producing denser string art nets.

Value

Invisibly returns a list of class stringart_result with:

pegs A data.frame with columns index, x, y, ray, and local_index.

connections A data.frame with columns connection_index, from, to, x_from, y_from, x_to, y_to, length, sweep, offset, local_from, and local_to.

total_length Total string length.

audit A character vector with audit information.

meta A list with construction metadata.

Examples

```
stnet()

res <- stnet(plot = FALSE)
head(res$pegs)
head(res$connections)
res$total_length

stnet(n = 40, k = 1, angle = pi / 2, col = "steelblue")
stnet(n = 40, k = 3, angle = 2 * pi / 3, col = "firebrick", lwd = 0.7)
stnet(show_envelope = TRUE)
stnet(template = TRUE)
```

stparabola *Generate a parabolic string art envelope*

Description

stparabola() generates a classical string art construction on two perpendicular axes. Pegs are placed on a horizontal axis and on a vertical axis, and straight strings are drawn between corresponding peg positions. The resulting family of line segments visually suggests a parabolic envelope.

Usage

```
stparabola(
  n = 30,
  k = 1,
  col = "blue",
  lwd = 1,
  plot = TRUE,
  show_points = TRUE,
  show_labels = FALSE,
  verbose = FALSE,
  width = 1,
  height = 1,
  show_strings = TRUE,
  template = FALSE,
  show_envelope = FALSE,
  envelope_col = "red",
  envelope_lwd = 1,
  envelope_lty = 2,
  point_col = "black",
  point_cex = 0.8,
  point_pch = 19,
  point_bg = "white",
  label_cex = 0.7,
  label_col = "black",
  border_col = "grey50",
  border_lwd = 1,
  bg = "white",
  main = NULL
)
```

Arguments

n	Integer. Number of pegs placed on each axis. Must be at least 3.
k	Integer. Number of shifted sweeps used in the construction. Must satisfy 1 <= k <= n - 1. The classical construction is obtained with k = 1.

col	String color passed to graphics::segments().
lwd	Positive number. Line width used to draw the strings.
plot	Logical. If TRUE, draws the figure.
show_points	Logical. If TRUE, draws the pegs.
show_labels	Logical. If TRUE, draws peg labels.
verbose	Logical. If TRUE, prints a short audit to the console.
width	Positive number. Length of the horizontal axis.
height	Positive number. Length of the vertical axis.
show_strings	Logical. If TRUE, draws the string connections.
template	Logical. If TRUE, draws only the peg template, without string connections. This is equivalent to setting show_strings = FALSE and show_points = TRUE.
show_envelope	Logical. If TRUE, draws the theoretical envelope of the classical construction.
envelope_col	Color used for the theoretical envelope.
envelope_lwd	Positive number. Line width used for the envelope.
envelope_lty	Line type used for the envelope.
point_col	Peg color.
point_cex	Positive number. Peg size.
point_pch	Plotting symbol used for pegs.
point_bg	Peg background color when applicable.
label_cex	Positive number. Label size.
label_col	Label color.
border_col	Axis color.
border_lwd	Positive number. Axis line width.
bg	Plot background color.
main	Optional plot title. If NULL, no title is displayed.

Details

This is one of the most classical string art constructions. Pegs are placed on two perpendicular axes. In the basic case, the peg at position i on the horizontal axis is connected to the peg at position i on the vertical axis, where the vertical axis is indexed from top to bottom.

For k = 1, the construction corresponds to the standard family of segments joining points (t, 0) and (0, 1 - t), after scaling by width and height. Its ideal envelope satisfies sqrt(x / width) + sqrt(y / height) = 1.

For k > 1, the function adds shifted sweeps of the same construction, producing denser string art patterns while preserving the same pedagogical idea of a family of straight lines generating a curved envelope.

Value

Invisibly returns a list of class `stringart_result` with:

pegs A `data.frame` with columns `index`, `x`, `y`, `axis`, and `local_index`.

connections A `data.frame` with columns `connection_index`, `from`, `to`, `x_from`, `y_from`, `x_to`, `y_to`, `length`, `sweep`, `offset`, `local_from`, and `local_to`.

total_length Total string length.

audit A character vector with audit information.

meta A list with construction metadata.

Examples

```
stparabola()

res <- stparabola(plot = FALSE)
head(res$pegs)
head(res$connections)
res$total_length

stparabola(n = 40, k = 1, col = "steelblue", lwd = 1)
stparabola(n = 40, k = 4, col = "firebrick", lwd = 0.6)
stparabola(show_points = TRUE, show_labels = TRUE)
stparabola(template = TRUE)
```

---

stpolygon *Generate string art on a regular polygon*

Description

`stpolygon()` generates a string art figure on the boundary of a regular polygon. Pegs are distributed uniformly along the polygon perimeter and connected using an additive modular rule.

Usage

```
stpolygon(
  n = 60,
  k = 7,
  col = "blue",
  lwd = 1,
  plot = TRUE,
  show_points = TRUE,
  show_labels = FALSE,
  verbose = FALSE,
  sides = 5,
  radius = 1,
  rotate = pi/2,
```

```
  show_strings = TRUE,
  template = FALSE,
  point_col = "black",
  point_cex = 0.8,
  point_pch = 19,
  point_bg = "white",
  label_cex = 0.7,
  label_col = "black",
  border_col = "grey50",
  border_lwd = 1,
  bg = "white",
  main = NULL
)
```

Arguments

n	Integer. Number of pegs placed along the polygon boundary. Must be at least 3 and at least sides.
k	Integer. Additive modular step used in the connection rule. Must satisfy 1 <= k <= n - 1.
col	String color passed to graphics::segments().
lwd	Positive number. Line width used to draw the strings.
plot	Logical. If TRUE, draws the figure.
show_points	Logical. If TRUE, draws the pegs.
show_labels	Logical. If TRUE, draws peg labels.
verbose	Logical. If TRUE, prints a short audit to the console.
sides	Integer. Number of polygon sides. Must be at least 3.
radius	Positive number. Circumradius of the polygon.
rotate	Numeric. Rotation angle in radians applied to the polygon.
show_strings	Logical. If TRUE, draws the string connections.
template	Logical. If TRUE, draws only the peg template, without string connections. This is equivalent to setting show_strings = FALSE and show_points = TRUE.
point_col	Peg color.
point_cex	Positive number. Peg size.
point_pch	Plotting symbol used for pegs.
point_bg	Peg background color when applicable.
label_cex	Positive number. Label size.
label_col	Label color.
border_col	Polygon border color.
border_lwd	Positive number. Polygon border line width.
bg	Plot background color.
main	Optional plot title. If NULL, no title is displayed.

Details

The polygon is inscribed in a circle of radius `radius`. Pegs are distributed uniformly along the full boundary, not only at the vertices. The additive modular connection rule is:

`to = ((from + k - 1) %% n) + 1`.

This construction supports the exploration of regular polygons, central and interior angles, symmetry, congruence, divisibility, and modular arithmetic.

Value

Invisibly returns a list of class `stringart_result` with:

pegs A `data.frame` with columns `index`, `x`, `y`, `side`, and `local_index`.

connections A `data.frame` with columns `connection_index`, `from`, `to`, `x_from`, `y_from`, `x_to`, `y_to`, and `length`.

total_length Total string length.

audit A character vector with audit information.

meta A list with construction metadata.

Examples

```
stpolygon()
stpolygon(sides = 5)
stpolygon(sides = 6)
stpolygon(sides = 8)
stpolygon(n = 60, k = 7, sides = 5)
stpolygon(template = TRUE)
```

---

stradial *Generate a radial string art pattern with triangular modules*

Description

`stradial()` generates a radial string art pattern composed of triangular modules rotated around the origin. In each module, pegs are placed along the triangular boundary and connected according to an additive modular rule.

Usage

```
stradial(
  n = 18,
  k = 5,
  col = "blue",
  lwd = 1,
  plot = TRUE,
  show_points = TRUE,
```

```
  show_labels = FALSE,
  verbose = FALSE,
  m = 6,
  r = 1.2,
  spread = pi/5,
  rotate = 0,
  show_strings = TRUE,
  template = FALSE,
  point_col = "black",
  point_cex = 0.8,
  point_pch = 21,
  point_bg = "white",
  label_cex = 0.7,
  label_col = "black",
  border_col = "grey50",
  border_lwd = 1,
  bg = "white",
  main = NULL,
  show_center = TRUE,
  center_col = "black",
  center_cex = 0.9
)
```

Arguments

n	Integer. Number of pegs in each triangular module. Must be at least 3.
k	Integer. Additive modular step used inside each module. Must satisfy 1 <= k <= n - 1.
col	String color, or a vector of colors with length 1 or m, used to draw the string connections.
lwd	Positive number. Line width used to draw the strings.
plot	Logical. If TRUE, draws the figure.
show_points	Logical. If TRUE, draws the pegs.
show_labels	Logical. If TRUE, draws peg labels.
verbose	Logical. If TRUE, prints a short audit to the console.
m	Integer. Number of triangular modules.
r	Positive number. Distance from the origin to the two outer vertices of each triangular module.
spread	Positive number. Angular opening, in radians, of each module.
rotate	Numeric. Rotation angle in radians applied to the whole figure.
show_strings	Logical. If TRUE, draws the string connections.
template	Logical. If TRUE, draws only the peg template, without string connections. This is equivalent to setting show_strings = FALSE and show_points = TRUE.
point_col	Peg border color.
point_cex	Positive number. Peg size.

point_pch	Plotting symbol used for pegs.
point_bg	Peg background color when applicable.
label_cex	Positive number. Label size.
label_col	Label color.
border_col	Module border color.
border_lwd	Positive number. Module border line width.
bg	Plot background color.
main	Optional plot title. If NULL, no title is displayed.
show_center	Logical. If TRUE, highlights the origin.
center_col	Color used to highlight the origin.
center_cex	Positive number. Size of the highlighted origin.

Details

Each module is a triangle with vertices at the origin and at two outer points determined by r and spread. The base module is rotated m times around the origin.

Within each module, the local connection rule is:

to = ((from + k - 1) %% n) + 1.

This means that each local peg is connected to the peg k positions ahead, using modular indexing. The same local rule is applied independently to all modules.

Value

Invisibly returns a list of class stringart_result with:

pegs A data.frame with columns index, x, y, module, and local_index.

connections A data.frame with columns connection_index, from, to, x_from, y_from, x_to, y_to, length, module, local_from, local_to, and color.

total_length Total string length.

audit A character vector with audit information.

meta A list with construction metadata.

Examples

```
stradial()

res <- stradial(plot = FALSE)
head(res$pegs)
head(res$connections)
res$total_length

stradial(n = 18, k = 5, m = 6, col = "steelblue", lwd = 0.8)
stradial(n = 12, k = 4, m = 5, show_points = TRUE, show_labels = TRUE)
stradial(template = TRUE)
```

stregion *Generate a string art figure from a closed region contour*

Description

`stregion()` generates a filled string art pattern from a closed contour. Pegs are distributed along the contour and connected to approximately opposite pegs, producing strings that cross the interior of the region.

Usage

```
stregion(
  n = 100,
  k = 4,
  col = "red",
  lwd = 0.6,
  plot = TRUE,
  show_points = TRUE,
  show_labels = FALSE,
  verbose = FALSE,
  contour = NULL,
  show_strings = TRUE,
  template = FALSE,
  draw_border = TRUE,
  border_col = "grey50",
  border_lwd = 1,
  point_col = "black",
  point_cex = 0.5,
  point_pch = 19,
  point_bg = "white",
  label_cex = 0.6,
  label_col = "black",
  bg = "white",
  main = NULL,
  add = FALSE,
  xlim = NULL,
  ylim = NULL
)
```

Arguments

`n`	Integer. Number of pegs placed along the contour. Must be at least 4.
`k`	Integer. Number of sweep offsets used to fill the region. Must be at least 1.
`col`	String color passed to `graphics::segments()`.
`lwd`	Positive number. Line width used to draw the strings.
`plot`	Logical. If `TRUE`, draws the figure.

show_points	Logical. If TRUE, draws the pegs.
show_labels	Logical. If TRUE, draws peg labels.
verbose	Logical. If TRUE, prints a short audit to the console.
contour	Optional data.frame with columns x and y defining a closed or open polygonal contour. If NULL, a default ellipse-like contour is used.
show_strings	Logical. If TRUE, draws the string connections.
template	Logical. If TRUE, draws only the peg template, without string connections. This is equivalent to setting show_strings = FALSE and show_points = TRUE.
draw_border	Logical. If TRUE, draws the region border.
border_col	Border color.
border_lwd	Positive number. Border line width.
point_col	Peg color.
point_cex	Positive number. Peg size.
point_pch	Plotting symbol used for pegs.
point_bg	Peg background color when applicable.
label_cex	Positive number. Label size.
label_col	Label color.
bg	Plot background color.
main	Optional plot title. If NULL, no title is displayed.
add	Logical. If TRUE, adds the string art region to the current graphics device instead of creating a new plot.
xlim, ylim	Optional axis limits used when plot = TRUE and add = FALSE.

Details

Unlike circular, elliptical, or triangular modular string art patterns that usually connect nearby pegs using a fixed additive step, stregion() is designed to fill a region. It connects each peg to a peg located approximately on the opposite side of the contour.

The main connection rule is:

to = ((from - 1 + floor(n / 2) + offset) %% n) + 1.

The argument k controls the number of offsets. Each offset produces one sweep of strings across the interior. Larger values of k create denser fillings.

Value

Invisibly returns a list of class stringart_result with:

pegs A data.frame with columns index, x, and y.

connections A data.frame with columns connection_index, from, to, x_from, y_from, x_to, y_to, length, sweep, and offset.

total_length Total string length.

audit A character vector with audit information.

meta A list with construction metadata.

Examples

```
stregion()

res <- stregion(plot = FALSE)
head(res$pegs)
head(res$connections)
res$total_length

custom_contour <- data.frame(
  x = c(0, 1, 0.6, -0.6, -1),
  y = c(1, 0.2, -0.9, -0.9, 0.2)
)
stregion(contour = custom_contour, n = 80, k = 3, col = "steelblue")
stregion(template = TRUE)
```

`strose` *Generate a rose-like string art pattern*

Description

`strose()` generates a rose-like string art figure based on a polar curve of the form `r(theta) = amplitude * (1 + cos(petals * theta)) / 2`.

Usage

```
strose(
  n = 240,
  k = 17,
  col = "deeppink4",
  lwd = 0.7,
  plot = TRUE,
  show_points = FALSE,
  show_labels = FALSE,
  verbose = FALSE,
  petals = 6,
  amplitude = 1,
  rotate = 0,
  show_strings = TRUE,
  template = FALSE,
  draw_curve = TRUE,
  border_col = "grey50",
  border_lwd = 1,
  point_col = "black",
  point_cex = 0.5,
  point_pch = 19,
  point_bg = "white",
  label_cex = 0.6,
```

```
  label_col = "black",
  bg = "white",
  main = NULL
)
```

Arguments

n	Integer. Number of pegs sampled along the curve. Must be at least 3.
k	Integer. Additive modular step used in the connection rule. Must satisfy 1 <= k <= n - 1.
col	String color passed to graphics::segments().
lwd	Positive number. Line width used to draw the strings.
plot	Logical. If TRUE, draws the figure.
show_points	Logical. If TRUE, draws the pegs.
show_labels	Logical. If TRUE, draws peg labels.
verbose	Logical. If TRUE, prints a short audit to the console.
petals	Integer. Number of petals in the rose-like curve.
amplitude	Positive number. Maximum radial amplitude.
rotate	Numeric. Rotation angle in radians.
show_strings	Logical. If TRUE, draws the string connections.
template	Logical. If TRUE, draws only the peg template, without string connections. This is equivalent to setting show_strings = FALSE and show_points = TRUE.
draw_curve	Logical. If TRUE, draws the underlying rose-like curve.
border_col	Curve color.
border_lwd	Positive number. Curve line width.
point_col	Peg color.
point_cex	Positive number. Peg size.
point_pch	Plotting symbol used for pegs.
point_bg	Peg background color when applicable.
label_cex	Positive number. Label size.
label_col	Label color.
bg	Plot background color.
main	Optional plot title. If NULL, no title is displayed.

Details

The pegs are sampled from the polar curve r(theta) = amplitude * (1 + cos(petals * theta)) / 2, which produces a rose-like pattern with petals visible petals.

The connections follow the additive modular rule to = ((from + k - 1) %% n) + 1.

Value

Invisibly returns a list of class stringart_result.

Examples

```
strose()
strose(petals = 6)
strose(petals = 8)
strose(template = TRUE)
```

`stspiral` *Generate a spiral string art pattern*

Description

`stspiral()` generates a string art pattern from pegs placed on an Archimedean spiral.

Usage

```
stspiral(
  n = 180,
  k = 13,
  col = "steelblue",
  lwd = 0.7,
  plot = TRUE,
  show_points = FALSE,
  show_labels = FALSE,
  verbose = FALSE,
  turns = 3,
  spacing = 0.6,
  inner_radius = 0,
  rotate = 0,
  show_strings = TRUE,
  template = FALSE,
  draw_curve = TRUE,
  border_col = "grey50",
  border_lwd = 1,
  point_col = "black",
  point_cex = 0.5,
  point_pch = 19,
  point_bg = "white",
  label_cex = 0.6,
  label_col = "black",
  bg = "white",
  main = NULL
)
```

Arguments

n	Integer. Number of pegs placed on the spiral. Must be at least 3.
k	Integer. Additive modular step used in the connection rule. Must satisfy 1 <= k <= n - 1.
col	String color passed to graphics::segments().
lwd	Positive number. Line width used to draw the strings.
plot	Logical. If TRUE, draws the figure.
show_points	Logical. If TRUE, draws the pegs.
show_labels	Logical. If TRUE, draws peg labels.
verbose	Logical. If TRUE, prints a short audit to the console.
turns	Positive number. Number of spiral turns.
spacing	Positive number. Radial growth per turn.
inner_radius	Nonnegative number. Initial spiral radius.
rotate	Numeric. Rotation angle in radians.
show_strings	Logical. If TRUE, draws the string connections.
template	Logical. If TRUE, draws only the peg template.
draw_curve	Logical. If TRUE, draws the underlying spiral.
border_col	Curve color.
border_lwd	Positive number. Curve line width.
point_col	Peg color.
point_cex	Positive number. Peg size.
point_pch	Plotting symbol used for pegs.
point_bg	Peg background color when applicable.
label_cex	Positive number. Label size.
label_col	Label color.
bg	Plot background color.
main	Optional plot title.

Details

The spiral uses the polar form `r(theta) = inner_radius + spacing * theta / (2 * pi)`.

Value

Invisibly returns a list of class `stringart_result`.

Examples

```
stspiral()
stspiral(turns = 4)
stspiral(template = TRUE)
```

`ststar` *Generate a star polygon string art pattern*

Description

`ststar()` generates a classical star polygon `{n/k}` by placing `n` pegs on a circle and connecting each peg to the peg `k` positions ahead.

Usage

```
ststar(
  n = 5,
  k = 2,
  col = "blue",
  lwd = 1,
  plot = TRUE,
  show_points = TRUE,
  show_labels = FALSE,
  verbose = FALSE,
  radius = 1,
  rotate = pi/2,
  show_strings = TRUE,
  template = FALSE,
  draw_polygon = TRUE,
  border_col = "grey50",
  border_lwd = 1,
  point_col = "black",
  point_cex = 0.8,
  point_pch = 19,
  point_bg = "white",
  label_cex = 0.7,
  label_col = "black",
  bg = "white",
  main = NULL
)
```

Arguments

`n`	Integer. Number of pegs on the circle. Must be at least 3.
`k`	Integer. Star step. Must satisfy `1 <= k <= n - 1`.
`col`	String color passed to `graphics::segments()`.
`lwd`	Positive number. Line width used to draw the star.
`plot`	Logical. If `TRUE`, draws the figure.
`show_points`	Logical. If `TRUE`, draws the pegs.
`show_labels`	Logical. If `TRUE`, draws peg labels.

verbose	Logical. If TRUE, prints a short audit to the console.
radius	Positive number. Circle radius.
rotate	Numeric. Rotation angle in radians applied to the star.
show_strings	Logical. If TRUE, draws the star connections.
template	Logical. If TRUE, draws only the peg template, without star connections. This is equivalent to setting show_strings = FALSE and show_points = TRUE.
draw_polygon	Logical. If TRUE, draws the underlying regular polygon.
border_col	Polygon border color.
border_lwd	Positive number. Border line width.
point_col	Peg color.
point_cex	Positive number. Peg size.
point_pch	Plotting symbol used for pegs.
point_bg	Peg background color when applicable.
label_cex	Positive number. Label size.
label_col	Label color.
bg	Plot background color.
main	Optional plot title. If NULL, no title is displayed.

Details

The star polygon uses the additive modular rule

`to = ((from + k - 1) %% n) + 1`.

The greatest common divisor `gcd(n, k)` determines the number of cycles. If `gcd(n, k) = 1`, the star is a single cycle. Otherwise, the construction decomposes into several independent cycles.

Value

Invisibly returns a list of class `stringart_result` with:

pegs A `data.frame` with columns `index`, `x`, and `y`.

connections A `data.frame` with columns `connection_index`, `from`, `to`, `x_from`, `y_from`, `x_to`, `y_to`, and `length`.

total_length Total string length.

audit A character vector with audit information.

meta A list with construction metadata.

Examples

```
ststar()
ststar(n = 5, k = 2)
ststar(n = 7, k = 2)
ststar(n = 8, k = 3)
ststar(template = TRUE)
```

`sttriangle` *Generate a triangular string art pattern*

Description

`sttriangle()` generates a string art pattern on the boundary of an equilateral triangle. Pegs are distributed uniformly along the triangle perimeter and connected using an additive modular rule.

Usage

```
sttriangle(
  n = 30,
  k = 7,
  col = "blue",
  lwd = 1,
  plot = TRUE,
  show_points = TRUE,
  show_labels = FALSE,
  verbose = FALSE,
  side = 2,
  rotate = 0,
  show_strings = TRUE,
  template = FALSE,
  point_col = "black",
  point_cex = 0.8,
  point_pch = 19,
  point_bg = "white",
  label_cex = 0.7,
  label_col = "black",
  border_col = "grey50",
  border_lwd = 1,
  bg = "white",
  main = NULL
)
```

Arguments

`n`	Integer. Number of pegs placed along the triangle boundary. Must be at least 3.
`k`	Integer. Additive modular step used in the connection rule. Must satisfy `1 <= k <= n - 1`.
`col`	String color passed to `graphics::segments()`.
`lwd`	Positive number. Line width used to draw the strings.
`plot`	Logical. If `TRUE`, draws the figure.
`show_points`	Logical. If `TRUE`, draws the pegs.
`show_labels`	Logical. If `TRUE`, draws peg labels.
`verbose`	Logical. If `TRUE`, prints a short audit to the console.

side	Positive number. Side length of the equilateral triangle.
rotate	Numeric. Rotation angle in radians applied to the whole figure.
show_strings	Logical. If TRUE, draws the string connections.
template	Logical. If TRUE, draws only the peg template, without string connections. This is equivalent to setting show_strings = FALSE and show_points = TRUE.
point_col	Peg color.
point_cex	Positive number. Peg size.
point_pch	Plotting symbol used for pegs.
point_bg	Peg background color when applicable.
label_cex	Positive number. Label size.
label_col	Label color.
border_col	Triangle border color.
border_lwd	Positive number. Triangle border line width.
bg	Plot background color.
main	Optional plot title. If NULL, no title is displayed.

Details

The triangle is centered at the origin using its centroid. The pegs are placed uniformly along the perimeter in counterclockwise order.

The additive modular connection rule is:

to = ((from + k - 1) %% n) + 1.

When gcd(n, k) = 1, the modular rule forms a single cycle through all pegs. When gcd(n, k) > 1, the construction decomposes into independent cycles.

Value

Invisibly returns a list of class stringart_result with:

pegs A data.frame with columns index, x, and y.

connections A data.frame with columns connection_index, from, to, x_from, y_from, x_to, y_to, and length.

total_length Total string length.

audit A character vector with audit information.

meta A list with construction metadata, including the triangle vertices.

Examples

```
sttriangle()

res <- sttriangle(plot = FALSE)
head(res$pegs)
head(res$connections)
res$total_length
```

```
sttriangle(n = 30, k = 7, side = 2, col = "blue", lwd = 1)
sttriangle(n = 18, k = 5, show_points = TRUE, show_labels = TRUE)
sttriangle(template = TRUE)
```

Index