Package: frenchCurve (via r-universe)

September 5, 2024

Type Package

Title Generate Open or Closed Interpolating Curves

Version 0.2.0

Author Bill Venables

Maintainer Bill Venables <`Bill.Venables@gmail.com`>

Description Functions for finding smooth interpolating curves connecting a series of points in the plane. Curves may be open or closed, that is, with the first and last point of the curve at the initial point.

License GPL-2

Imports stats, sp

Depends graphics, grDevices

Encoding UTF-8

RoxygenNote 7.2.0

Suggests ggplot2, knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation no

Date/Publication 2022-06-18 06:30:06 UTC

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

RemoteUrl https://github.com/cran/frenchCurve

RemoteRef HEAD

RemoteSha e8327d1627a332ffdd3a5e3d323b9607410971eb

Contents

adjust_curve 2
as.data.frame.curve 3
as_complex 4
as_polygon 4
complexReplacement 5
open_curve 6
%inside% 7

Index **9**

---

adjust_curve *Interactive curve adjustment*

Description

A simple interactive device for adjusting a curve. Given a set of points, the curve is plotted and may then be adjusted interactively by clicking on any of the points, one at a time, and clicking again at its intended new position.

Usage

```
adjust_curve(
  x,
  y = NULL,
  ...,
  plotit = TRUE,
  curve = open_curve,
  ccolour = "#DF536B",
  pcolour = "#2297E6"
)
```

Arguments

`x, y` Any means of specifying points in the plane, as accepted by xy.coords()

`...` additional arguments past on to curve()

`plotit` logical: should the curve be plotted (TRUE) or can it be assumed the points are already on the display (FALSE)?

`curve` One of the curve type functions of this package

`ccolour` character string: colour for the curve in the plot

`pcolour` character string: colour for the points in the plot

Value

The adjusted points which define the adjusted curve

as.data.frame.curve *Conversion to data frame*

Description

Method function to convert an object inheriting from class "curve" to a data.frame

Usage

```
## S3 method for class 'curve'
as.data.frame(x, row.names = NULL, optional = FALSE, ...)
```

Arguments

x	An object inheriting from class "curve"
row.names, optional, ...	as for as.data.frame.

Value

A data frame object

Examples

```
library(ggplot2)
set.seed(1234)
z <- complex(real = runif(5), imaginary = runif(5))
z <- z[order(Arg(z - mean(z)))]
cz <- closed_curve(z)
oz <- open_curve(z)
ggplot() + geom_path(data = as.data.frame(cz), aes(x,y), colour = "#DF536B") +
    geom_path(data = as.data.frame(oz), aes(x,y), colour = "#2297E6") +
    geom_point(data = as.data.frame(z), aes(x = Re(z), y = Im(z))) +
    geom_segment(data = as.data.frame(z), aes(x = Re(mean(z)),
                                              y = Im(mean(z)),
                                              xend = Re(z),
                                              yend = Im(z)),
                 arrow = arrow(angle=15, length=unit(0.125, "inches")),
                 colour = alpha("grey", 1/2)) +
    theme_bw()
```

as_complex *Coerce two dimensional points to complex*

Description

Convenience function for allowing any of the usual ways two dimensional points can be specified in traditional graphics to define a complex variable

Usage

```
as_complex(x, y = NULL)
```

Arguments

x, y	A two dimensional specification, as allowed by grDevices::xy.coords

Value

A complex vector

Examples

```
loc <- cbind(runif(20), runif(20))
z <- as_complex(loc)
z <- z-mean(z)
Mod(z) <- 1
z <- z[order(Arg(z))]
plot(closed_curve(z), asp = 1, col = 2)
lines(z, col = 4)
points(z, pch=16)
```

as_polygon *Make a Simple Polygon or Points*

Description

A simple polygon is here defined as a data frame with numeric components x and y without any duplicate rows. The order of rows is significant in defining the associated figure.

Usage

```
as_polygon(x, y = NULL, ...)

## Default S3 method:
as_polygon(x, y = NULL, ...)

## S3 method for class 'curve'
as_polygon(x, y = NULL, ...)

as_points(x, y = NULL)
```

Arguments

`x, y`	any specification of 2-d points, or a "curve" object
`...`	additional arguments not currently used

Details

A ’points’ object is defined as a data frame with numeric columns x and y.

Value

a data frame with components x and y

complexReplacement *Complex vector property replacement functions*

Description

Complex vector property replacement functions

Usage

```
Re(x) <- value

Im(x) <- value

Mod(x) <- value

Arg(x) <- value
```

Arguments

`x`	a complex vector to be altered
`value`	the numerical value vector to be used in the alteration

Value

An appropriately modified complex vector

---

open_curve *Curved Interpolation*

Description

Interpolate between ordered 2-d points with a smooth curve. open_curve() produces an open curve; closed_curve() produces a closed curve. Bezier curves are also provided.

Usage

```
open_curve(x, y = NULL, n = 100 * length(z), asp = 1, ...)

## S3 method for class 'curve'
plot(
  x,
  y = NULL,
  type = "l",
  lty = "solid",
  xpd = NA,
  pch = 20,
  ...,
  include_points = TRUE
)

## S3 method for class 'curve'
points(x, pch = 20, xpd = NA, ...)

## S3 method for class 'curve'
lines(x, xpd = NA, ...)

closed_curve(x, y = NULL, n0 = 500 * length(z0), asp = 1, ...)

bezier_curve(x, y = NULL, n = 500, t = seq(0, 1, length.out = n), ...)
```

Arguments

x, y	Any of the forms used to specify a 2-d set of points or an object of class "curve"
n, n0	number of points in the interpolating curve
asp	the relative scale for x versus that of y
...	additional arguments past on to other methods
pch, type, lty, xpd	plot arguments or traditional graphics parameters
include_points	logical:should points be included in the plot?
t	for Bezier curves, parameter value sequence ranging from 0 to 1

Value

a list with components x, y, and points, of S3 class "curve"

Examples

```
oldPar <- par(pty = "s", mfrow = c(2, 2), mar = c(1,1,2,1), xpd = NA)
z <- (complex(argument = seq(-0.9*base::pi, 0.9*base::pi, length = 20)) +
     complex(modulus = 0.125, argument = runif(20, -base::pi, base::pi))) *
     complex(argument = runif(1, -base::pi, base::pi))

plot(z, asp=1, axes = FALSE, ann = FALSE, panel.first = grid())
title(main = "Open")
segments(Re(z[1]), Im(z[1]), Re(z[20]), Im(z[20]), col = "grey", lty = "dashed")
lines(open_curve(z), col = "red")

plot(z, asp=1, axes = FALSE, ann = FALSE, panel.first = grid())
title(main = "Closed")
lines(closed_curve(z), col = "royal blue")

plot(z, asp=1, axes = FALSE, ann = FALSE, panel.first = grid())
title(main = "Bezier")
lines(bezier_curve(z), col = "dark green")

plot(z, asp=1, axes = FALSE, ann = FALSE, panel.first = grid())
title(main = "Circle")
lines(complex(argument = seq(-base::pi, base::pi, len = 500)),
      col = "purple")

par(oldPar)
```

%inside% *Check if points lie inside a simple polygon*

Description

Check if points lie inside a simple polygon

Usage

```
points %inside% polygon
```

Arguments

points	a data.frame with components x,y specifying the points
polygon	a data.frame with components x,y specifying the polygon

Value

a logical value matching the number of points, TRUE = "inside"

Examples

```
oldPar <- par(pty = "s", las = 1, xpd = NA)
pts <- expand.grid(x = seq(0, 1, len=25), y = seq(0, 1, len=25))
pol <- (1 + 1i)/2 + complex(argument = seq(-base::pi, base::pi, len=100))/3
show_red <- as_points(pts) %inside% as_polygon(pol)
plot(pts, col = ifelse(show_red, "red", "royal blue"), ann = FALSE, bty = "n",
     pch = ".", cex = ifelse(show_red, 4, 2.5), asp = 1)
polygon(pol, lwd = 0.5)
par(oldPar)
```

Index

%inside%, 7

adjust_curve, 2
Arg<- (complexReplacement), 5
as.data.frame, *3*
as.data.frame.curve, 3
as_complex, 4
as_points (as_polygon), 4
as_polygon, 4

bezier_curve (open_curve), 6

closed_curve (open_curve), 6
complexReplacement, 5

Im<- (complexReplacement), 5

lines.curve (open_curve), 6

Mod<- (complexReplacement), 5

open_curve, 6

plot.curve (open_curve), 6
points.curve (open_curve), 6

Re<- (complexReplacement), 5