Package ‘RConics’

October 12, 2022

Type Package

Title Computations on Conics

Version 1.1.1

Date 2022-03-11

Maintainer Emanuel Huber <emanuel.huber@pm.me>

Description Solve some conic related problems (intersection of conics with lines and conics, arc length of an ellipse, polar lines, etc.).

License GPL (>= 2)

Encoding UTF-8

Depends base, R (>= 2.10)

URL https://github.com/emanuelhuber/RConics

BugReports https://github.com/emanuelhuber/RConics/issues

RoxygenNote 7.1.2

Suggests conics

NeedsCompilation no

Author Emanuel Huber [aut, cre] (<https://orcid.org/0000-0002-5528-0412>)

Repository CRAN

Date/Publication 2022-03-11 12:00:05 UTC

R topics documented:

`RConics-package` *RConics: Computations on conics*

Description

A package to solve some conic related problems (intersection of conics with lines and conics, arc length of an ellipse, polar lines, etc.).

Details

Some of the functions are based on the *projective* geometry. In projective geometry parallel lines meet at an infinite point and all infinite points are incident to a line at infinity. Points and lines of a projective plane are represented by *homogeneous* coordinates, that means by 3D vectors: (x, y, z) for the points and (a, b, c) such that $ax + by + c = 0$ for the lines. The Euclidian points correspond to $(x, y, 1)$, the infinite points to $(x, y, 0)$, the Euclidean lines to (a, b, c) with $a \neq 0$ or $b \neq 0$, the line at infinity to $(0, 0, 1)$.

Advice: to plot conics use the package `conics` from Bernard Desgraupes.

This work was funded by the Swiss National Science Foundation within the ENSEMBLE project (grant no. CRSI_132249).

Author(s)

Maintainer: Emanuel Huber `<emanuel.huber@pm.me>` (ORCID)

References

Richter-Gebert, Jürgen (2011). *Perspectives on Projective Geometry - A Guided Tour Through Real and Complex Geometry*, Springer, Berlin, ISBN: 978-3-642-17285-4

See Also

Useful links:

- `https://github.com/emanuelhuber/RConics`
- Report bugs at `https://github.com/emanuelhuber/RConics/issues`

addLine *Plot a "homogeneous" line to a plot.*

Description

Add a homogeneous line to a plot. The line parameters must be in homogeneous coordinates, e.g. (a, b, c).

Usage

```
addLine(l, ...)

plotHLine(l, ...)
```

Arguments

`l`	A 3×1 vector of the homogeneous representation of a line.
`...`	graphical parameters such as `col`, `lty` and `lwd`.

Examples

```
# two points in homogeneous coordinates
p1 <- c(3,1,1)
p2 <- c(0,2,1)

# homogeneous line joining p1 and p2
l_12 <- join(p1,p2)
l_12

# plot
plot(0,0,type="n", xlim=c(-2,5),ylim=c(-2,5),asp=1)
points(t(p1))
points(t(p2))
addLine(l_12,col="red",lwd=2)
```

adjoint *Adjoint matrix*

Description

Compute the classical adjoint (also called adjugate) of a square matrix. The adjoint is the transpose of the cofactor matrix.

Usage

```
adjoint(A)
```

Arguments

`A`	a square matrix.

Value

The adjoint matrix of A (square matrix with the same dimension as A).

See Also

`cofactor`, `minor`

Examples

```
A <- matrix(c(1,4,5,3,7,2,2,8,3),nrow=3,ncol=3)
A
B <- adjoint(A)
B
```

`arcLengthEllipse` *Arc length of an ellipse*

Description

This function computes the arc length of an ellipse centered in $(0, 0)$ with the semi-axes aligned with the x- and y-axes. The arc length is defined by the points 1 and 2. These two points do not need to lie exactly on the ellipse: the x-coordinate of the points and the quadrant where they lie define the positions on the ellipse used to compute the arc length.

Usage

```
arcLengthEllipse(p1, p2 = NULL, saxes, n = 5)
```

Arguments

`p1`	a (2×1) vector of the Cartesian coordinates of point 1.
`p2`	a (2×1) vector of the Cartesian coordinates of point 2 (optional).
`saxes`	a (2×1) vector of length of the semi-axes of the ellipse.
`n`	the number of iterations used in the numerical approximation of the incomplete elliptic integral of the second kind.

Details

If the coordinates `p2` of the point 2 are omitted the function `arcLengthEllipse` computes the arc length between the point 1 and the point defined by $(0, b)$, b beeing the minor semi-axis.

Value

The length of the shortest arc of the ellipse defined by the points 1 and 2.

Source

Van de Vel, H. (1969). *On the series expansion method for Computing incomplete elliptic integrals of the first and second kinds*, Math. Comp. 23, 61-69.

See Also

pEllipticInt

Examples

```
p1 <- c(3,1)
p2 <- c(0,2)

# Ellipse with semi-axes: a = 5, b= 2
saxes <- c(5,2)

# 1 iteration
arcLengthEllipse(p1,p2,saxes,n=1)

# 5 iterations
arcLengthEllipse(p1,p2,saxes,n=5)

# 10 iterations
arcLengthEllipse(p1,p2,saxes,n=10)
```

---

cofactor — *(i, j)-cofactor and (i, j)-minor of a matrix*

Description

Compute the (i, j)-cofactor, respectively the (i, j)-minor of the matrix A. The (i, j)-cofactor is obtained by multiplying the (i, j)-minor by $(-1)^{i+j}$. The (i, j)-minor of A, is the determinant of the $(n-1) \times (n-1)$ matrix that results by deleting the i-th row and the j-th column of A.

Usage

```
cofactor(A, i, j)

minor(A, i, j)
```

Arguments

A	a square matrix.
i	the i-th row.
j	the j-th column.

Value

The (i, j)-minor/cofactor of the matrix A (single value).

See Also

adjoint

Examples

```
A <- matrix(c(1,4,5,3,7,2,2,8,3),nrow=3,ncol=3)
A
minor(A,2,3)
cofactor(A,2,3)
```

---

colinear *Test for colinearity*

Description

Tests if three points are colinear. The coordinates of the points have to be in homogeneous coordinates.

Usage

```
colinear(p1, p2, p3)
```

Arguments

p1	(3×1) vector of the homogeneous coordinates of point 1.
p2	(3×1) vector of the homogeneous coordinates of point 2.
p3	(3×1) vector of the homogeneous coordinates of point 3.

Value

TRUE if the three points are colinear, else FALSE.

Source

Richter-Gebert, Jürgen (2011). *Perspectives on Projective Geometry - A Guided Tour Through Real and Complex Geometry*, Springer, Berlin, ISBN: 978-3-642-17285-4

Examples

```
# points: homogeneous coordinates
p1 <- c(3,1,1)
p2 <- c(0,2,1)
p3 <- c(1.5,-2,1)
p4 <- c(1,3,1)

# homogeneous line passing through p1 and p2
l1 <- join(p1,p2)

# homogeneous line passing through p3 and p3
l2 <- join(p3,p4)

# homogeneous points formed by the intersection of the lines
p5 <- meet(l1,l2)

# test for colinearity
colinear(p1, p2, p3)
colinear(p1, p2, p5)
colinear(p3, p4, p5)

# plot
plot(rbind(p1,p2,p3,p4),xlim=c(-5,5),ylim=c(-5,5),asp=1)
abline(h=0,v=0,col="grey",lty=3)
addLine(l1,col="red")
addLine(l2,col="blue")
points(t(p5),cex=1.5,pch=20,col="blue")
```

`conicMatrixToEllipse` *Transformation of the matrix representation of an ellipse into the ellipse parameters*

Description

Ellipses can be represented by a (3×3) matrix A, such that for each point x on the ellipse $x^T Ax = 0$. The function `conicMatrixToEllipse` transforms the matrix A into the ellipse parameters: center location, semi-axes length and angle of rotation.

Usage

```
conicMatrixToEllipse(A)
```

Arguments

`A`	a (3×3) matrix representation of an ellipse.

Value

`loc`	a (2×1) vector of the Cartesian coordinates of the ellipse center.
`saxes`	a (2×1) vector of the length of the ellipse semi-axes.
`theta`	the angle of rotation of the ellipse (in radians).

References

Wolfram, Mathworld (http://mathworld.wolfram.com/).

See Also

ellipseToConicMatrix

Examples

```
# ellipse parameter
saxes <- c(5,2)
loc <- c(0,0)
theta <- pi/4
# matrix representation of the ellipse
C <- ellipseToConicMatrix(saxes,loc,theta)
C
# back to the ellipse parameters
conicMatrixToEllipse(C)
```

---

conicThrough5Points *Compute the conic that passes through 5 points*

Description

Return the matrix representation of the conic that passes through exactly 5 points.

Usage

```
conicThrough5Points(p1, p2, p3, p4, p5)
```

Arguments

`p1`	(3×1) vectors of the homogeneous coordinates of one of the five points.
`p2`	(3×1) vectors of the homogeneous coordinates of one of the five points.
`p3`	(3×1) vectors of the homogeneous coordinates of one of the five points.
`p4`	(3×1) vectors of the homogeneous coordinates of one of the five points.
`p5`	(3×1) vectors of the homogeneous coordinates of one of the five points.

Value

A (3×3) matrix representation of the conic passing through the 5 points.

Source

Richter-Gebert, Jürgen (2011). *Perspectives on Projective Geometry - A Guided Tour Through Real and Complex Geometry*, Springer, Berlin, ISBN: 978-3-642-17285-4

Examples

```
# five points
p1 <- c(-4.13, 6.24, 1)
p2 <- c(-8.36, 1.17, 1)
p3 <- c(-2.03, -4.61, 1)
p4 <- c(9.70, -3.49, 1)
p5 <- c(8.02, 3.34, 1)

# matrix representation of the conic passing
# through the five points
C5 <- conicThrough5Points(p1,p2,p3,p4,p5)

# plot
plot(rbind(p1,p2,p3,p4,p5),xlim=c(-10,10), ylim=c(-10,10), asp=1)
# from matrix to ellipse parameters
E5 <- conicMatrixToEllipse(C5)
lines(ellipse(E5$saxes, E5$loc, E5$theta, n=500))
```

---

cubic *Roots of the cubic equation.*

Description

Return the roots of a cubic equation of the form $ax^3 + bx^2 + cx + d = 0$.

Usage

```
cubic(p)
```

Arguments

p	a (4×1) vector of the four parameters (a, b, c, d) of the cubic equation.

Value

A vector corresponding to the roots of the cubic equation.

Source

W. H. Press, S.A. Teukolsky, W.T. Vetterling, B.P. Flannery (2007). *NUMERICAL RECIPES - the art of scientific computing*. Cambridge, University Press, chap 5.6, p. 227-229.

Examples

```
# cubic equation x^3 - 6x^2 + 11x - 6 = 0
# parameter
b <- c(1,-6, 11, -6)

# roots
x0 <- cubic(b)

# plot
x <- seq(0,4, by=0.001)
y <- b[1]*x^3 + b[2]*x^2 + b[3]*x + b[4]

# plot
plot(x,y,type="l")
abline(h=0,v=0)
points(cbind(x0,c(0,0,0)), pch=20,col="red",cex=1.8)
```

`ellipse` *Return ellipse points*

Description

Return ellipse points. Usefull for ploting ellipses.

Usage

```
ellipse(
  saxes = c(1, 1),
  loc = c(0, 0),
  theta = 0,
  n = 201,
  method = c("default", "angle", "distance")
)
```

Arguments

`saxes`	a (2×1) vector of the length of the ellipse semi-axes.
`loc`	a (2×1) vector of the Cartesian coordinates of the ellipse center.
`theta`	the angle of rotation of the elllipse (in radians).
`n`	the number of points returned by the function.
`method`	The method used to return the points: either `"default"`, `"angle"`, or `"distance"` (see Details).

Details

`"default"` returns points according to the polar equation;

`"angle"` returns points radially equidistant;

`"distance"` returns points that are equidistant on the ellipse arc.

Value

A $(n \times 2)$ matrix whose columns correspond to the Cartesian coordinates of the points lying on the ellipse.

Examples

```
# Ellipse parameters
saxes <- c(5,2)
loc <- c(0,0)
theta <- pi/4

# Plot
plot(ellipse(saxes, loc, theta, n=500),type="l")
points(ellipse(saxes, loc, theta, n=30),pch=20,col="red")
points(ellipse(saxes, loc, theta, n=30, method="angle"),pch=20,col="blue")
points(ellipse(saxes, loc, theta, n=30, method="distance"),pch=20,col="green")
```

`ellipseToConicMatrix` *Transformation of the ellipse parameters into the matrix representation*

Description

Transformation of the ellipse parameters (Cartesian coordinates of the ellipse center, length of the semi-axes and angle of rotation) into the (3×3) into the matrix representation of conics.

Usage

```
ellipseToConicMatrix(saxes = c(1, 1), loc = c(0, 0), theta = 0)
```

Arguments

`saxes`	a (2×1) vector of the length of the ellipse semi-axes.
`loc`	a (2×1) vector of the Cartesian coordinates of the ellipse center.
`theta`	the angle of rotation of the ellipse (in radians).

Value

A (3×3) matrix that represents the ellipse.

See Also

`conicMatrixToEllipse`

Examples

```
# Ellipse parameters
saxes <- c(5,2)
loc <- c(0,0)
theta <- pi/4
# Matrix representation of the ellipse
C <- ellipseToConicMatrix(saxes,loc,theta)
```

`intersectConicConic` *Intersection between two conics*

Description

Returns the point(s) of intersection between two conics in homogeneous coordinates.

Usage

```
intersectConicConic(C1, C2)
```

Arguments

`C1`	(3×3) matrix representation of conics.
`C2`	(3×3) matrix representation of conics.

Value

The homogeneous coordinates of the intersection points. If there are k points of intersection, it returns a $(3 \times k)$ matrix whose columns correspond to the homogeneous coordinates of the intersection points. If there is only one point, a (3×1) vector of the homogeneous coordinates of the intersection point is returned. If there is no intersection, `NULL` is returned.

Source

Richter-Gebert, Jürgen (2011). *Perspectives on Projective Geometry - A Guided Tour Through Real and Complex Geometry*, Springer, Berlin, ISBN: 978-3-642-17285-4

Examples

```
# Ellipse with semi-axes a=8, b=2, centered in (0,0), with orientation angle = -pi/3
C1 <- ellipseToConicMatrix(c(8,2), c(0,0), -pi/3)

# Ellipse with semi-axes a=5, b=2, centered in (1,-2), with orientation angle = pi/5
C2 <- ellipseToConicMatrix(c(5,2), c(1,-2), pi/5)

# intersection conic C with conic C2
p_CC2 <- intersectConicConic(C1,C2)

# plot
```

```
plot(ellipse(c(8,2), c(0,0), -pi/3),type="l",asp=1)
lines(ellipse(c(5,2), c(1,-2), pi/5), col="blue")
points(t(p_CC2), pch=20,col="blue")
```

`intersectConicLine` *Intersections between a conic and a line*

Description

Returns the point(s) of intersection between a conic and a line in homogeneous coordinates.

Usage

```
intersectConicLine(C, l)
```

Arguments

`C`	(3×3) matrix representation of conics.
`l`	a (3×3) vector of the homogeneous representation of a line.

Value

The homogeneous coordinates of the intersection points. If there are two points of intersection, it returns a (3×2) matrix whose columns correspond to the homogeneous coordinates of the intersection points. If there is only one point, a (3×1) vector of the homogeneous coordinates of the intersection point is returned. If there is no intersection, `NULL` is returned.

Source

Richter-Gebert, Jürgen (2011). *Perspectives on Projective Geometry - A Guided Tour Through Real and Complex Geometry*, Springer, Berlin, ISBN: 978-3-642-17285-4

Examples

```
#' # Ellipse with semi-axes a=8, b=2, centered in (0,0), with orientation angle = -pi/3
C <- ellipseToConicMatrix(c(8,2),c(0,0),-pi/3)

# line
l <- c(0.25,0.85,-3)

# intersection conic C with line l:
p_Cl <- intersectConicLine(C,l)

# plot
plot(ellipse(c(8,2),c(0,0),-pi/3),type="l",asp=1)
addLine(l,col="red")
points(t(p_Cl), pch=20,col="red")
```

join *The join and meet of two points and the parallel*

Description

The join operation of two points is the cross-product of these two points and represents the line passing through them. The meet operation of two lines is the cross-product of these two lines and represents their intersection. The line parallel to a line l and passing through the point p corresponds to the join of p with the meet of l and the line at infinity.

Usage

```
join(p, q)

meet(l, m)

parallel(p, l)
```

Arguments

p	(3×1) vectors of the homogeneous coordinates of a point.
q	(3×1) vectors of the homogeneous coordinates of a point.
l	(3×1) vectors of the homogeneous representation of a line.
m	(3×1) vectors of the homogeneous representation of a line.

Value

A (3×1) vector of either the homogeneous coordinates of the meet of two lines (a point), the homogeneous representation of the join of two points (line), or the homogeneous representation of the parallel line. The vector has the form $(x, y, 1)$.

Source

Richter-Gebert, Jürgen (2011). *Perspectives on Projective Geometry - A Guided Tour Through Real and Complex Geometry*, Springer, Berlin, ISBN: 978-3-642-17285-4

Examples

```
p <- c(3,1,1)
q <- c(0,2,1)
l <- c(0.75,0.25,1)

# m is the line passin through p and q
m <- join(p,q)

# intersection point of m and l
ml <- meet(l,m)
```

```
# line parallel to l and through p
lp <- parallel(p,l)

# plot
plot(rbind(p,q),xlim=c(-5,5),ylim=c(-5,5))
abline(h=0,v=0,col="grey",lty=3)
addLine(l,col="red")
addLine(m,col="blue")
points(t(ml),cex=1.5,pch=20,col="blue")
addLine(lp,col="green")
```

---

pEllipticInt *Partial elliptic integral*

Description

Partial elliptic integral

Usage

```
pEllipticInt(x, saxes, n = 5)
```

Arguments

x	the x-coordinate.
saxes	a (2×1) vector of the length of the ellipse semi-axes.
n	the number of iterations.

Value

Return the partial elliptic integral.

Source

Van de Vel, H. (1969). *On the series expansion method for Computing incomplete elliptic integrals of the first and second kinds*, Math. Comp. 23, 61-69.

See Also

arcLengthEllipse

Examples

```
# Ellipse with semi-axes: a = 5, b= 2
saxes <- c(5,2)

# 1 iteration
pEllipticInt(3,saxes,n=1)
# 5 iterations
pEllipticInt(3,saxes,n=5)
# 10 iterations
pEllipticInt(3,saxes,n=10)
```

---

polar *Polar line of point with respect to a conic*

Description

Return the polar line l of a point p with respect to a conic with matrix representation C. The polar line l is defined by $l = Cp$.

Usage

```
polar(p, C)
```

Arguments

p	a (3×1) vector of the homogeneous coordinates of a point.
C	a (3×3) matrix representation of the conic.

Details

The polar line of a point p on a conic is tangent to the conic on p.

Value

A (3×1) vector of the homogeneous representation of the polar line.

Source

Richter-Gebert, Jürgen (2011). *Perspectives on Projective Geometry - A Guided Tour Through Real and Complex Geometry*, Springer, Berlin, ISBN: 978-3-642-17285-4

Examples

```
# Ellipse with semi-axes a=5, b=2, centered in (1,-2), with orientation angle = pi/5
C <- ellipseToConicMatrix(c(5,2),c(1,-2),pi/5)

# line
l <- c(0.25,0.85,-1)

# intersection conic C with line l:
p_Cl <- intersectConicLine(C,l)

# if p is on the conic, the polar line is tangent to the conic
l_p <- polar(p_Cl[,1],C)

# point outside the conic
p1 <- c(5,-3,1)
l_p1 <- polar(p1,C)

# point inside the conic
p2 <- c(-1,-4,1)
l_p2 <- polar(p2,C)

# plot
plot(ellipse(c(5,2),c(1,-2),pi/5),type="l",asp=1, ylim=c(-10,2))
# addLine(l,col="red")
points(t(p_Cl[,1]), pch=20,col="red")
addLine(l_p,col="red")
points(t(p1), pch=20,col="blue")
addLine(l_p1,col="blue")
points(t(p2), pch=20,col="green")
addLine(l_p2,col="green")

# DUAL CONICS
saxes <- c(5,2)
theta <- pi/7
E <- ellipse(saxes,theta=theta, n=50)
C <-  ellipseToConicMatrix(saxes,c(0,0),theta)
plot(E,type="n",xlab="x", ylab="y", asp=1)
points(E,pch=20)
E <- rbind(t(E),rep(1,nrow(E)))

All_tangant <- polar(E,C)
apply(All_tangant, 2, addLine, col="blue")
```

`quadraticFormToMatrix` *Transformation of the quadratic conic representation into the matrix representation.*

Description

Transformation of the quadratic conic representation into the matrix representation.

Usage

```
quadraticFormToMatrix(v)
```

Arguments

v	a (6×1) vector of the parameters (a, b, c, d, e, f) of the quadratic form $ax^2 + bxy + cy^2 + dxz + eyz + fz^2 = 0$.

Value

A (3×3) matrix representation of the conic (symmetric matrix).

Examples

```
v <- c(2,2,-2,-20,20,10)
quadraticFormToMatrix(v)
```

---

rotation *Affine planar transformations matrix*

Description

(3×3) affine planar transformation matrix corresponding to reflection, rotation, scaling and translation in projective geometry. To transform a point p multiply the transformation matrix A with the homogeneous coordinates (x, y, z) of p (e.g. $p_{transformed} = Ap$).

Usage

```
rotation(theta, pt = NULL)

translation(v)

scaling(s)

reflection(alpha)
```

Arguments

theta	the angle of the rotation (in radian).
pt	the homogeneous coordinates of the rotation center (optional).
v	the (2×1) translation vector in direction x and y.
s	the (2×1) scaling vector in direction x and y.
alpha	the angle made by the line of reflection (in radian).

Value

A (3×3) affine transformation matrix.

Source

Richter-Gebert, Jürgen (2011). *Perspectives on Projective Geometry - A Guided Tour Through Real and Complex Geometry*, Springer, Berlin, ISBN: 978-3-642-17285-4

Examples

```
p1 <- c(2,5,1)  # homogeneous coordinate

# rotation
r_p1 <- rotation(4.5) %*% p1

# rotation centered in (3,1)
rt_p1 <- rotation(4.5, pt=c(3,1,1)) %*% p1

# translation
t_p1 <- translation(c(2,-4)) %*% p1

# scaling
s_p1 <- scaling(c(-3,1)) %*% p1

# plot
plot(t(p1),xlab="x",ylab="y", xlim=c(-5,5),ylim=c(-5,5),asp=1)
abline(v=0,h=0, col="grey",lty=1)
abline(v=3,h=1, col="grey",lty=3)
points(3,1,pch=4)
points(t(r_p1),col="red",pch=20)
points(t(rt_p1),col="blue",pch=20)
points(t(t_p1),col="green",pch=20)
points(t(s_p1),col="black",pch=20)
```

---

skewSymmetricMatrix (3×3) *skew symmetric matrix*

Description

Return a (3×3) skew symmetric matrix from three parameters (λ, μ, τ).

Usage

```
skewSymmetricMatrix(p)
```

Arguments

p	a (3×1) vector (λ, μ, τ)

Value

A (3×3) skew symmetric matrix, with :

- $A_{1,2} = -A_{2,1} = \tau$
- $-A_{1,3} = A_{3,1} = \mu$
- $A_{3,2} = -A_{2,3} = \lambda$

Source

Richter-Gebert, Jürgen (2011). *Perspectives on Projective Geometry - A Guided Tour Through Real and Complex Geometry*, Springer, Berlin, ISBN: 978-3-642-17285-4

Examples

```
p <- c(3,7,11)
skewSymmetricMatrix(p)
```

---

splitDegenerateConic *Split degenerate conic*

Description

Split a degenerate conic into two lines.

Usage

```
splitDegenerateConic(C)
```

Arguments

`C`	a (3×3) matrix representation of a degenerate conic.

Value

A (3×2) matrix whose columns correspond to the homongeneous representation of two lines (real or complex).

Source

Richter-Gebert, Jürgen (2011). *Perspectives on Projective Geometry - A Guided Tour Through Real and Complex Geometry*, Springer, Berlin, ISBN: 978-3-642-17285-4

Examples

```
# tw0 lines
g <- c(0.75,0.25,3)
h <- c(0.5,-0.25,2)

# a degenerate conic
D <- g %*% t(h) + h %*% t(g)

# split the degenerate conic into 2 lines
L <- splitDegenerateConic(D)

# plot
plot(0,0,xlim=c(-10,5),ylim=c(-10,10),type="n")
addLine(L[,1],col="red")
addLine(L[,2],col="green")
```

Index