Package 'RSpincalc'

July 17, 2015

Type Package

Title Conversion Between Attitude Representations of DCM, Euler Angles, Quaternions, and Euler Vectors

Version 1.0.2

Encoding UTF-8

Description
Conversion between attitude representations: DCM, Euler angles, Quaternions, and Euler vectors.
Plus conversion between 2 Euler an-
gle set types (xyx, yzy, zxz, xzx, yxy, zyz, xyz, yzx, zxy, xzy, yxz, zyx).
Fully vectorized code, with warnings/errors for Euler angles (singularity, out of range, in-
valid angle order),
DCM (orthogonality, not proper, exceeded tolerance to unity determinant) and Euler vec-
tors(not unity).
Also quaternion and other useful functions.
Based on SpinCalc by John Fuller and SpinConv by Paolo de Leva.

License GPL (>= 3)

Author Jose Gama [aut, cre],
John Fuller [aut, cph],
Paolo Leva [aut, cph]

Maintainer Jose Gama <rxprtgama@gmail.com>

Repository CRAN

Repository/R-Forge/Project rspincalc

Repository/R-Forge/Revision 16

Repository/R-Forge/DateTimeStamp 2015-07-16 22:34:34

Date/Publication 2015-07-17 12:51:32

NeedsCompilation no

R topics documented:

DCM2EA *Convert from Direction Cosine Matrix to Euler Angles*

Description

DCM2EA converts from Direction Cosine Matrix (DCM) to Euler Angles (EA).

Usage

```
DCM2EA(DCM, EulerOrder='zyx', tol = 10 * .Machine$double.eps, ichk = FALSE,
ignoreAllChk = FALSE)
```

Arguments

`DCM`	Direction Cosine Matrix (DCM) is a rotation matrix 3x3 (N=1) or an array 3x3xN
`EulerOrder`	Euler Angles (EA) is a vector [psi, theta, phi]
`tol`	Tolerance from deviations from unity for the determinant of rotation matrices or the the vector length for unitary vectors.
`ichk`	Logical, FALSE=disables near-singularity warnings.
`ignoreAllChk`	Logical, TRUE=disables all warnings and error checks (use with caution!).

Details

Euler Angles (EA) xyz <=> x(roll) y(pitch) z(yaw) Type 1 Rotations (Tait-Bryan angles): xyz - xzy - yxz - yzx - zyx - zxy Singular if second rotation angle is -90 or 90 degrees. Type 2 Rotations (proper Euler angles): xyx - xzx - yxy - yzy - zxz - zyz Singular if second rotation angle is 0 or 180 degrees.

Euler angles [psi, theta, phi] range from -90 to 90 degrees. Tait-Bryan angles [psi, theta, phi] range from 0 to 180 degrees. Angles about Euler vectors range from 0 to 180 degrees.

Value

Euler Angles (EA) vector [psi, theta, phi]

Author(s)

Jose Gama

References

by John Fuller, 14 Jul 2008 SpinCalc, Function to Convert between DCM, Euler angles, Quaternions, and Euler vectors. `http://www.mathworks.com/matlabcentral/fileexchange/20696-function-to-convert-b`

Paolo de Leva, 01 May 2013 SpinConv, Conversion from a rotation representation type to another. `http://www.mathworks.com/matlabcentral/fileexchange/41562-spinconv`

See Also

`EA2DCM`

Examples

```
DCM <- matrix(c(-0.3573404, -0.1515663, 0.9215940, 0.6460385, 0.6724915,
0.3610947, -0.6744939, 0.7244189, -0.1423907),3,3,byrow=TRUE)
DCM2EA(DCM,'xyz')
```

DCM2EV *Convert from Direction Cosine Matrix to Euler Vectors*

Description

DCM2EV converts from Direction Cosine Matrix (DCM) to Euler Vectors (EV).

Usage

```
DCM2EV(DCM, tol = 10 * .Machine$double.eps, ichk = FALSE, ignoreAllChk = FALSE)
```

Arguments

DCM	Direction Cosine Matrix (DCM) is a rotation matrix 3x3 (N=1) or an array 3x3xN.
tol	Tolerance from deviations from unity for the determinant of rotation matrices or the the vector length for unitary vectors.
ichk	Logical, FALSE=disables near-singularity warnings.
ignoreAllChk	Logical, TRUE=disables all warnings and error checks (use with caution!).

Value

Euler Vectors (EV) vector [m1, m2, m3, MU]

Author(s)

Jose Gama

References

by John Fuller, 14 Jul 2008 SpinCalc, Function to Convert between DCM, Euler angles, Quaternions, and Euler vectors. http://www.mathworks.com/matlabcentral/fileexchange/20696-function-to-convert-b

Paolo de Leva, 01 May 2013 SpinConv, Conversion from a rotation representation type to another. http://www.mathworks.com/matlabcentral/fileexchange/41562-spinconv

See Also

EV2DCM

Examples

```
DCM <- matrix(c(-0.3573404, -0.1515663, 0.9215940, 0.6460385, 0.6724915,
0.3610947, -0.6744939, 0.7244189, -0.1423907),3,3,byrow=TRUE)
DCM2EV(DCM)
```

DCM2Q *Convert from Direction Cosine Matrix to rotation Quaternions*

Description

DCM2Q converts from Direction Cosine Matrix (DCM) to Quaternions (Q).

Usage

```
DCM2Q(DCM, tol = 10 * .Machine$double.eps, ichk = FALSE, ignoreAllChk = FALSE)
```

Arguments

DCM	Direction Cosine Matrix (DCM) is a rotation matrix 3x3 (N=1) or an array 3x3xN
tol	Tolerance from deviations from unity for the determinant of rotation matrices or the the vector length for unitary vectors.
ichk	Logical, FALSE=disables near-singularity warnings.
ignoreAllChk	Logical, TRUE=disables all warnings and error checks (use with caution!).

Value

Quaternion (Q) vector [q1, q2, q3, q4].

Author(s)

Jose Gama

References

by John Fuller, 14 Jul 2008 SpinCalc, Function to Convert between DCM, Euler angles, Quaternions, and Euler vectors. http://www.mathworks.com/matlabcentral/fileexchange/20696-function-to-convert-b

Paolo de Leva, 01 May 2013 SpinConv, Conversion from a rotation representation type to another. http://www.mathworks.com/matlabcentral/fileexchange/41562-spinconv

See Also

Q2DCM

Examples

```
DCM <- matrix(c(-0.3573404, -0.1515663, 0.9215940, 0.6460385, 0.6724915,
0.3610947, -0.6744939, 0.7244189, -0.1423907),3,3,byrow=TRUE)
DCM2Q(DCM)
```

DCMrandom *Generate uniform random direction cosine matrices*

Description

DCMrandom generates uniform random direction cosine matrices.

Usage

```
DCMrandom(n=NA, tol = 10 * .Machine$double.eps, ignoreAllChk=FALSE)
```

Arguments

n	Optional integer for the number of generated direction cosine matrices, default = 1.
tol	Tolerance from deviations from unity for the determinant of rotation matrices or the the vector length for unitary vectors.
ignoreAllChk	Logical, TRUE=disables all warnings and error checks (use with caution!).

Value

DCM	Direction cosine matrix or array (DCM).

Author(s)

Jose Gama

Examples

```
DCMrandom()
DCMrandom(5)
```

EA2DCM *Convert from Euler Angles to Direction Cosine Matrix*

Description

EA2DCM converts from Euler Angles (EA) to Direction Cosine Matrix (DCM).

Usage

```
EA2DCM(EA, EulerOrder='zyx', tol = 10 * .Machine$double.eps, ichk = FALSE,
ignoreAllChk = FALSE)
```

Arguments

`EA`	Euler Angles (EA) vector [psi, theta, phi].
`EulerOrder`	Euler Order (xyx, yzy, zxz, xzx, yxy, zyz, xyz, yzx, zxy, xzy, yxz, zyx)
`tol`	Tolerance from deviations from unity for the determinant of rotation matrices or the the vector length for unitary vectors.
`ichk`	Logical, FALSE=disables near-singularity warnings.
`ignoreAllChk`	Logical, TRUE=disables all warnings and error checks (use with caution!).

Details

Euler Angles (EA) xyz <=> x(roll) y(pitch) z(yaw) Type 1 Rotations (Tait-Bryan angles): xyz - xzy - yxz - yzx - zyx - zxy Singular if second rotation angle is -90 or 90 degrees. Type 2 Rotations (proper Euler angles): xyx - xzx - yxy - yzy - zxz - zyz Singular if second rotation angle is 0 or 180 degrees.

Euler angles [psi, theta, phi] range from -90 to 90 degrees. Tait-Bryan angles [psi, theta, phi] range from 0 to 180 degrees. Angles about Euler vectors range from 0 to 180 degrees.

Value

Direction Cosine Matrix (DCM) 3x3xN.

Author(s)

Jose Gama

References

by John Fuller, 14 Jul 2008 SpinCalc, Function to Convert between DCM, Euler angles, Quaternions, and Euler vectors. `http://www.mathworks.com/matlabcentral/fileexchange/20696-function-to-convert-b`

Paolo de Leva, 01 May 2013 SpinConv, Conversion from a rotation representation type to another. `http://www.mathworks.com/matlabcentral/fileexchange/41562-spinconv`

See Also

`DCM2EA`

Examples

```
EAxyx <- c(-170.6607, 110.937, 136.2344) * (pi/180)
EA2DCM(EAxyx,'xyx')
```

EA2EA *Convert from Euler Angles to Euler Angles*

Description

EA2EA converts from Euler Angles (EA) to Euler Angles (EA).

Usage

```
EA2EA(EA, EulerOrder1='zyx', EulerOrder2='zyx', tol = 10 * .Machine$double.eps,
ichk = FALSE, ignoreAllChk = FALSE)
```

Arguments

EA	Euler Angles (EA) vector [psi, theta, phi].
EulerOrder1	Euler Order 1 (xyx, yzy, zxz, xzx, yxy, zyz, xyz, yzx, zxy, xzy, yxz, zyx)
EulerOrder2	Euler Order 2 (xyx, yzy, zxz, xzx, yxy, zyz, xyz, yzx, zxy, xzy, yxz, zyx)
tol	Tolerance from deviations from unity for the determinant of rotation matrices or the the vector length for unitary vectors.
ichk	Logical, FALSE=disables near-singularity warnings.
ignoreAllChk	Logical, TRUE=disables all warnings and error checks (use with caution!).

Details

Euler Angles (EA) xyz <=> x(roll) y(pitch) z(yaw) Type 1 Rotations (Tait-Bryan angles): xyz - xzy - yxz - yzx - zyx - zxy Singular if second rotation angle is -90 or 90 degrees. Type 2 Rotations (proper Euler angles): xyx - xzx - yxy - yzy - zxz - zyz Singular if second rotation angle is 0 or 180 degrees.

Euler angles [psi, theta, phi] range from -90 to 90 degrees. Tait-Bryan angles [psi, theta, phi] range from 0 to 180 degrees. Angles about Euler vectors range from 0 to 180 degrees.

Value

Euler Angles (EA) vector [psi, theta, phi].

Author(s)

Jose Gama

References

by John Fuller, 14 Jul 2008 SpinCalc, Function to Convert between DCM, Euler angles, Quaternions, and Euler vectors. http://www.mathworks.com/matlabcentral/fileexchange/20696-function-to-convert-b

Paolo de Leva, 01 May 2013 SpinConv, Conversion from a rotation representation type to another. http://www.mathworks.com/matlabcentral/fileexchange/41562-spinconv

See Also

EA2DCM, EA2Q, EA2EV

Examples

```
## Not run:
EAxyx <- c(-170.6607, 110.937, 136.2344)
EA2EA(EAxyx,'xyx','xyz')
EA2EA(EAxyx,'xyx','xzy')
EA2EA(EAxyx,'xyx','yzx')
EA2EA(EAxyx,'xyx','yxz')
EA2EA(EAxyx,'xyx','zxy')
EA2EA(EAxyx,'xyx','zyx')
EA2EA(EAxyx,'xyx','xzx')
EA2EA(EAxyx,'xyx','yxy')
EA2EA(EAxyx,'xyx','yzy')
EA2EA(EAxyx,'xyx','zxz')
EA2EA(EAxyx,'xyx','zyz')

## End(Not run)
```

EA2EV *Convert from Euler Angles to Euler Vectors*

Description

EA2EV converts from Euler Angles (EA) to Euler Vectors (EV).

Usage

```
EA2EV(EA, EulerOrder='zyx', tol = 10 * .Machine$double.eps, ichk = FALSE,
ignoreAllChk = FALSE)
```

Arguments

EA	Euler Angles (EA) vector [psi, theta, phi].
EulerOrder	Euler Order (xyx, yzy, zxz, xzx, yxy, zyz, xyz, yzx, zxy, xzy, yxz, zyx)
tol	Tolerance from deviations from unity for the determinant of rotation matrices or the the vector length for unitary vectors.
ichk	Logical, FALSE=disables near-singularity warnings.
ignoreAllChk	Logical, TRUE=disables all warnings and error checks (use with caution!).

Details

Euler Angles (EA) xyz <=> x(roll) y(pitch) z(yaw) Type 1 Rotations (Tait-Bryan angles): xyz - xzy - yxz - yzx - zyx - zxy Singular if second rotation angle is -90 or 90 degrees. Type 2 Rotations (proper Euler angles): xyx - xzx - yxy - yzy - zxz - zyz Singular if second rotation angle is 0 or 180 degrees.

Euler angles [psi, theta, phi] range from -90 to 90 degrees. Tait-Bryan angles [psi, theta, phi] range from 0 to 180 degrees. Angles about Euler vectors range from 0 to 180 degrees.

Value

Euler Vectors (EV) vector [m1, m2, m3, MU].

Author(s)

Jose Gama

References

by John Fuller, 14 Jul 2008 SpinCalc, Function to Convert between DCM, Euler angles, Quaternions, and Euler vectors. http://www.mathworks.com/matlabcentral/fileexchange/20696-function-to-convert-b

Paolo de Leva, 01 May 2013 SpinConv, Conversion from a rotation representation type to another. http://www.mathworks.com/matlabcentral/fileexchange/41562-spinconv

See Also

EV2EA

Examples

```
EAxyx <- c(-170.6607, 110.937, 136.2344) * (pi/180)
EA2EV(EAxyx,'xyx')
```

EA2Q *Convert from Euler Angles to rotation Quaternions*

Description

EA2Q converts from Euler Angles (EA) to Quaternions (Q).

Usage

```
EA2Q(EA, EulerOrder='zyx', ichk = FALSE, ignoreAllChk = FALSE)
```

Arguments

`EA`	Euler Angles (EA) vector [psi, theta, phi].
`EulerOrder`	Euler Order (xyx, yzy, zxz, xzx, yxy, zyz, xyz, yzx, zxy, xzy, yxz, zyx)
`ichk`	Logical, FALSE=disables near-singularity warnings.
`ignoreAllChk`	Logical, TRUE=disables all warnings and error checks (use with caution!).

Details

Euler Angles (EA) xyz <=> x(roll) y(pitch) z(yaw) Type 1 Rotations (Tait-Bryan angles): xyz - xzy - yxz - yzx - zyx - zxy Singular if second rotation angle is -90 or 90 degrees. Type 2 Rotations (proper Euler angles): xyx - xzx - yxy - yzy - zxz - zyz Singular if second rotation angle is 0 or 180 degrees.

Euler angles [psi, theta, phi] range from -90 to 90 degrees. Tait-Bryan angles [psi, theta, phi] range from 0 to 180 degrees. Angles about Euler vectors range from 0 to 180 degrees.

Value

Quaternions (Q) vector [q1, q2, q3, q4].

Author(s)

Jose Gama

References

by John Fuller, 14 Jul 2008 SpinCalc, Function to Convert between DCM, Euler angles, Quaternions, and Euler vectors. http://www.mathworks.com/matlabcentral/fileexchange/20696-function-to-convert-b

Paolo de Leva, 01 May 2013 SpinConv, Conversion from a rotation representation type to another. http://www.mathworks.com/matlabcentral/fileexchange/41562-spinconv

See Also

Q2EA

Examples

```
EAxyx <- c(-170.6607, 110.937, 136.2344) * (pi/180)
EA2Q(EAxyx,'xyx')
```

EArandom *Generate uniform random Euler Angles*

Description

EArandom generates uniform random Euler Angles.

Usage

```
EArandom(n=NA, EulerOrder='zyx', tol = 10 * .Machine$double.eps, ignoreAllChk=FALSE)
```

Arguments

n	Optional integer for the number of generated Euler Angles, default = 1.
EulerOrder	Euler Order (xyx, yzy, zxz, xzx, yxy, zyz, xyz, yzx, zxy, xzy, yxz, zyx).
tol	Tolerance from deviations from unity for the determinant of rotation matrices or the the vector length for unitary vectors.
ignoreAllChk	Logical, TRUE=disables all warnings and error checks (use with caution!).

Value

EA	Euler Angles (EA).

Author(s)

Jose Gama

Examples

```
EArandom()
EArandom(5)
```

EV2DCM *Convert from Euler Vectors to Direction Cosine Matrix*

Description

EV2DCM converts from Euler Vectors (EV) to Direction Cosine Matrix (DCM).

Usage

```
EV2DCM(EV, tol = 10 * .Machine$double.eps, ichk = FALSE, ignoreAllChk = FALSE)
```

Arguments

`EV`	Euler Vectors (EV) vector [m1, m2, m3, MU].
`tol`	Tolerance from deviations from unity for the determinant of rotation matrices or the the vector length for unitary vectors.
`ichk`	Logical, FALSE=disables near-singularity warnings.
`ignoreAllChk`	Logical, TRUE=disables all warnings and error checks (use with caution!).

Value

Direction Cosine Matrix (DCM) 3x3xN.

Author(s)

Jose Gama

References

by John Fuller, 14 Jul 2008 SpinCalc, Function to Convert between DCM, Euler angles, Quaternions, and Euler vectors. http://www.mathworks.com/matlabcentral/fileexchange/20696-function-to-convert-b

Paolo de Leva, 01 May 2013 SpinConv, Conversion from a rotation representation type to another. http://www.mathworks.com/matlabcentral/fileexchange/41562-spinconv

See Also

`DCM2EV`

Examples

```
EV <- c(-0.1995301, -0.8765382, -0.4380279, 114.4324 * (pi/180))
EV2DCM(EV,1e-7)
#EV2DCM(EV)
```

`EV2EA` *Convert from Euler Vectors to Euler Angles*

Description

`EV2EA` converts from Euler Vectors (EV) to Euler Angles (EA).

Usage

```
EV2EA(EV, EulerOrder='zyx', tol = 10 * .Machine$double.eps, ichk = FALSE,
ignoreAllChk = FALSE)
```

Arguments

`EV`	Euler Vectors (EV) vector [m1, m2, m3, MU].
`EulerOrder`	Euler Order (xyx, yzy, zxz, xzx, yxy, zyz, xyz, yzx, zxy, xzy, yxz, zyx)
`tol`	Tolerance from deviations from unity for the determinant of rotation matrices or the the vector length for unitary vectors.
`ichk`	Logical, FALSE=disables near-singularity warnings.
`ignoreAllChk`	Logical, TRUE=disables all warnings and error checks (use with caution!).

Details

Euler Angles (EA) xyz <=> x(roll) y(pitch) z(yaw) Type 1 Rotations (Tait-Bryan angles): xyz - xzy - yxz - yzx - zyx - zxy Singular if second rotation angle is -90 or 90 degrees. Type 2 Rotations (proper Euler angles): xyx - xzx - yxy - yzy - zxz - zyz Singular if second rotation angle is 0 or 180 degrees.

Euler angles [psi, theta, phi] range from -90 to 90 degrees. Tait-Bryan angles [psi, theta, phi] range from 0 to 180 degrees. Angles about Euler vectors range from 0 to 180 degrees.

Value

Euler Angles (EA) vector [psi, theta, phi].

Author(s)

Jose Gama

References

by John Fuller, 14 Jul 2008 SpinCalc, Function to Convert between DCM, Euler angles, Quaternions, and Euler vectors. http://www.mathworks.com/matlabcentral/fileexchange/20696-function-to-convert-b

Paolo de Leva, 01 May 2013 SpinConv, Conversion from a rotation representation type to another. http://www.mathworks.com/matlabcentral/fileexchange/41562-spinconv

See Also

EA2EV

Examples

```
## Not run:
EV <- c(-0.1995301, -0.8765382, -0.4380279, 114.4324 * (pi/180))
EV2EA(EV,'xyx')

## End(Not run)
```

EV2Q *Convert from Euler Vectors to rotation Quaternions*

Description

EV2Q converts from Euler Vectors (EV) to Quaternions (Q).

Usage

```
EV2Q(EV, tol = 10 * .Machine$double.eps, ichk = FALSE, ignoreAllChk = FALSE)
```

Arguments

EV	Euler Vectors (EV) vector [m1, m2, m3, MU].
tol	Tolerance from deviations from unity for the determinant of rotation matrices or the the vector length for unitary vectors.
ichk	Logical, FALSE=disables near-singularity warnings.
ignoreAllChk	Logical, TRUE=disables all warnings and error checks (use with caution!).

Value

Quaternions (Q) vector [q1, q2, q3, q4].

Author(s)

Jose Gama

References

by John Fuller, 14 Jul 2008 SpinCalc, Function to Convert between DCM, Euler angles, Quaternions, and Euler vectors. http://www.mathworks.com/matlabcentral/fileexchange/20696-function-to-convert-b

Paolo de Leva, 01 May 2013 SpinConv, Conversion from a rotation representation type to another. http://www.mathworks.com/matlabcentral/fileexchange/41562-spinconv

See Also

Q2EV

Examples

```
EV <- c(-0.1995301, -0.8765382, -0.4380279, 114.4324 * (pi/180))
EV2Q(EV,1e-7)
#EV2Q(EV)
```

EVrandom — *Generate uniform random Euler Vectors*

Description

`EVrandom` generates uniform random Euler Vectors.

Usage

```
EVrandom(n=NA, tol = 10 * .Machine$double.eps, ignoreAllChk=FALSE)
```

Arguments

`n`	Optional integer for the number of generated Euler Vectors, default = 1.
`tol`	Tolerance from deviations from unity for the determinant of rotation matrices or the the vector length for unitary vectors.
`ignoreAllChk`	Logical, TRUE=disables all warnings and error checks (use with caution!).

Value

`EV`	Euler Vectors (EV).

Author(s)

Jose Gama

Examples

```
EVrandom()
EVrandom(5)
```

isPureRotationMatrix — *Determine if the variable is a pure rotation matrix*

Description

`isPureRotationMatrix` determines if a matrix is pure rotation matrix (proper orthogonal matrix) with det(m)==1. `isPureQuaternion` determines if a quaternion is a pure quaternion. `isRealQuaternion` determines if a quaternion is a real quaternion. `isUnitQuaternion` determines if a quaternion is a unit quaternion.

Usage

```
isPureRotationMatrix(DCM, tol = 0.01)
```

Arguments

`DCM`	Direction Cosine Matrix (DCM) is a rotation matrix 3x3 (N=1) or an array 3x3xN.
`tol`	Tolerance value.

Value

Logical, TRUE = matrix is pure rotation matrix.

Author(s)

Jose Gama

See Also

`Q2GL`

Examples

```
isPureRotationMatrix(matrix(rep(0,9),3,3,byrow=TRUE),.1)
isPureRotationMatrix(matrix(rep(1,9),3,3,byrow=TRUE),.1)
isPureRotationMatrix(matrix(c(0,0,-1,0,1,0,1,0,1),3,3,byrow=TRUE),.1)
DCMx10 <- DCMrandom(10)
isPureRotationMatrix(DCMx10)
```

`Q2DCM` *Convert from rotation Quaternions to Direction Cosine Matrix*

Description

`Q2DCM` converts from Quaternions to Direction Cosine Matrix (DCM).

Usage

```
Q2DCM(Q, tol = 10 * .Machine$double.eps, ichk = FALSE, ignoreAllChk = FALSE)
```

Arguments

`Q`	Quaternion (Q) vector [q1, q2, q3, q4].
`tol`	Tolerance from deviations from unity for the determinant of rotation matrices or the the vector length for unitary vectors.
`ichk`	Logical, FALSE=disables near-singularity warnings.
`ignoreAllChk`	Logical, TRUE=disables all warnings and error checks (use with caution!).

Value

Direction Cosine Matrix (DCM) 3x3xN.

Author(s)

Jose Gama

References

by John Fuller, 14 Jul 2008 SpinCalc, Function to Convert between DCM, Euler angles, Quaternions, and Euler vectors. http://www.mathworks.com/matlabcentral/fileexchange/20696-function-to-convert-b

Paolo de Leva, 01 May 2013 SpinConv, Conversion from a rotation representation type to another. http://www.mathworks.com/matlabcentral/fileexchange/41562-spinconv

See Also

DCM2Q

Examples

```
Q <- c(-0.1677489, -0.7369231, -0.3682588, 0.5414703)
Q2DCM(Q)
```

Q2EA *Convert from rotation Quaternions to Euler Angles*

Description

Q2EA converts from Quaternions (Q) to Euler Angles (EA) based on D. M. Henderson (1977). Q2EA.Xiao is the algorithm by J. Xiao (2013) for the Princeton Vision Toolkit - included here to allow reproducible research.

Usage

```
Q2EA(Q, EulerOrder='zyx', tol = 10 * .Machine$double.eps, ichk = FALSE,
ignoreAllChk = FALSE)
```

Arguments

Q	Quaternion (Q) vector [q1, q2, q3, q4].
EulerOrder	Euler Order (xyx, yzy, zxz, xzx, yxy, zyz, xyz, yzx, zxy, xzy, yxz, zyx).
tol	Tolerance from deviations from unity for the determinant of rotation matrices or the the vector length for unitary vectors.
ichk	Logical, FALSE=disables near-singularity warnings.
ignoreAllChk	Logical, TRUE=disables all warnings and error checks (use with caution!).

Details

Euler Angles (EA) xyz <=> x(roll) y(pitch) z(yaw) Type 1 Rotations (Tait-Bryan angles): xyz - xzy - yxz - yzx - zyx - zxy Singular if second rotation angle is -90 or 90 degrees. Type 2 Rotations (proper Euler angles): xyx - xzx - yxy - yzy - zxz - zyz Singular if second rotation angle is 0 or 180 degrees.

Euler angles [psi, theta, phi] range from -90 to 90 degrees. Tait-Bryan angles [psi, theta, phi] range from 0 to 180 degrees. Angles about Euler vectors range from 0 to 180 degrees.

Value

Euler Angles (EA) vector [psi, theta, phi].

Author(s)

Jose Gama

References

D. M. Henderson, 1977 Shuttle Program. Euler Angles, Quaternions, and Transformation Matrices Working Relationships. National Aeronautics and Space Administration (NASA), N77-31234/6

J. Xiao, 2013 Princeton Vision Toolkit. Available from: http://vision.princeton.edu/code.html http://vision.princeton.edu/pvt/GCBreader/quaternion.m

John Fuller, 14 Jul 2008 SpinCalc, Function to Convert between DCM, Euler angles, Quaternions, and Euler vectors. http://www.mathworks.com/matlabcentral/fileexchange/20696-function-to-convert-betwee

Paolo de Leva, 01 May 2013 SpinConv, Conversion from a rotation representation type to another. http://www.mathworks.com/matlabcentral/fileexchange/41562-spinconv

See Also

EA2Q

Examples

```
## Not run:
Q <- c(-0.1677489, -0.7369231, -0.3682588, 0.5414703)
Q2EA(Q,'xyx')

## End(Not run)
```

`Q2EV` *Convert from rotation Quaternions to Euler Vectors*

Description

Q2EV converts from Quaternions (Q) to Euler Vectors (EV).

Usage

```
Q2EV(Q, tol = 10 * .Machine$double.eps, ichk = FALSE, ignoreAllChk = FALSE)
```

Arguments

`Q`	Quaternion (Q) vector [q1, q2, q3, q4].
`tol`	Tolerance from deviations from unity for the determinant of rotation matrices or the the vector length for unitary vectors.
`ichk`	Logical, FALSE=disables near-singularity warnings.
`ignoreAllChk`	Logical, TRUE=disables all warnings and error checks (use with caution!).

Value

Euler Vectors (EV) vector [m1, m2, m3, MU].

Author(s)

Jose Gama

References

by John Fuller, 14 Jul 2008 SpinCalc, Function to Convert between DCM, Euler angles, Quaternions, and Euler vectors. http://www.mathworks.com/matlabcentral/fileexchange/20696-function-to-convert-b

Paolo de Leva, 01 May 2013 SpinConv, Conversion from a rotation representation type to another. http://www.mathworks.com/matlabcentral/fileexchange/41562-spinconv

See Also

EV2Q

Examples

```
Q <- c(-0.1677489, -0.7369231, -0.3682588, 0.5414703)
Q2EV(Q)
```

`Q2GL` *Convert from rotation Quaternions to OpenGL rotation matrix*

Description

`DCM2EA` converts from Quaternions (Q) to OpenGL rotation matrix.

Usage

```
Q2GL(Q)
```

Arguments

`Q`	rotation Quaternions (Q) vector [q1, q2, q3, q4].

Value

OpenGL rotation matrix 4x4xN.

Author(s)

Jose Gama

References

Python - IMU Brick 2012 http://www.tinkerforge.com/doc/Software/Bricks/IMU_Brick_Python.html

See Also

`isPureRotationMatrix`

Examples

```
Q <- c(-0.1677489, -0.7369231, -0.3682588, 0.5414703)
Q2GL(Q)
```

QangularDifference — *Angular difference between 2 quaternions*

Description

`QangularDifference` returns the angular difference between 2 quaternions.

Usage

```
QangularDifference(Q1, Q2)
```

Arguments

`Q1`	Quaternion (Q) vector [q1, q2, q3, q4].
`Q2`	Quaternion (Q) vector [q1, q2, q3, q4].

Value

Real value = angular difference between 2 quaternions.

Author(s)

Jose Gama

Examples

```
Q1 <- Qrandom()
Q2 <- Qrandom()
QangularDifference(Q1, Q2)
```

Qconj — *Quaternion conjugate*

Description

`Qconj` performs a quaternion conjugate operation.

Usage

```
Qconj(Q)
```

Arguments

`Q`	Quaternion (Q) vector [q1, q2, q3, q4].

Value

Q	Conjugate quaternion (Q) vector [q1, q2, q3, q4].

Author(s)

Jose Gama

Examples

```
Q <- c(-0.1677489, -0.7369231, -0.3682588, 0.5414703)
Qconj(Q)
```

Qinv — *Quaternion inverse*

Description

`Qinv` calculated the quaternion inverse.

Usage

```
Qinv(Q)
```

Arguments

Q	Quaternion (Q) vector [q1, q2, q3, q4].

Value

Q	Quaternion inverse (Q) vector [q1, q2, q3, q4].

Author(s)

Jose Gama

Examples

```
Q <- c(-0.1677489, -0.7369231, -0.3682588, 0.5414703)
Qinv(Q)
```

Qlerp *Linear quaternion interpolation*

Description

Qlerp linear quaternion interpolation. Qslerp spherical linear interpolation. QslerpNoInvert version of slerp, used by squad, that does not check for theta > 90. Qspline spherical cubic interpolation. Qsquad spherical and Quadrangle linear interpolation. Qbezier Shoemake-Bezier interpolation using De Castlejau algorithm. Qspline for 3 quaternions, qn-1,qn and qn+1, calculate a control point to be used in spline interpolation.

Usage

```
Qlerp(Q1, Q2, fracT)
```

Arguments

Q1	Quaternion (Q) vector [q1, q2, q3, q4].
Q2	Quaternion (Q) vector [q1, q2, q3, q4].
fracT	Fraction of .

Value

Q	Zero or one-valued quaternion (Q) vector [q1, q2, q3, q4] or matrix n x 4.

Author(s)

Jose Gama

Examples

```
Q1 <- Qrandom()
Q2 <- Qrandom()
Qlerp(Q1, Q2, 0.1)
```

Qlog *Quaternion logarithm*

Description

Qlog performs a quaternion logarithm operation. Qexp performs a quaternion exponential operation.

Usage

```
Qlog(Q)
Qexp(Q)
```

Arguments

Q	Quaternion (Q) vector [q1, q2, q3, q4].

Value

Q	Result quaternion (Q) vector [q1, q2, q3, q4].

Author(s)

Jose Gama

Examples

```
Q <- c(-0.1677489, -0.7369231, -0.3682588, 0.5414703)
Qlog(Q)
Qexp(Q)
```

Qnorm *Norm of a quaternion*

Description

`Qnorm` calculates the norm of a quaternion.

Usage

```
Qnorm(Q)
```

Arguments

Q	Quaternion (Q) vector [q1, q2, q3, q4].

Value

Norm of the quaternion.

Author(s)

Jose Gama

Examples

```
Q <- c(-0.1677489, -0.7369231, -0.3682588, 0.5414703)
Qnorm(Q)
```

`Qnormalize` *Quaternion normalization*

Description

`Qnormalize` performs a quaternion normalization.

Usage

```
Qnormalize(Q)
```

Arguments

`Q`	Quaternion (Q) vector [q1, q2, q3, q4].

Value

`Q`	Normalized quaternion (Q) vector [q1, q2, q3, q4].

Author(s)

Jose Gama

Examples

```
Q <- c(-0.1677489, -0.7369231, -0.3682588, 0.5414703)
Qnormalize(Q)
```

`Qrandom` *Generate uniform random unit quaternions*

Description

`Qrandom` generates uniform random unit quaternions.

Usage

```
Qrandom(n=NA)
```

Arguments

`n`	Optional integer for the number of generated quaternions, default = 1.

Value

`Q`	Uniform random unit quaternion (Q) vector [q1, q2, q3, q4] or matrix n x 4.

Author(s)

Jose Gama

Examples

```
Qrandom()
Qrandom(5)
```

`Qrot` *Updates current attitude quaternion*

Description

`Qrot` updates the current attitude quaternion.

Usage

```
Qrot(Q, w, dT)
```

Arguments

`Q`	Quaternion (Q) vector [q1, q2, q3, q4].
`w`	Angular rate values [wx, wy, wz].
`dT`	Inverse of update rate.

Value

`Q`	Updated quaternion (Q) vector [q1, q2, q3, q4].

Author(s)

Jose Gama

Examples

```
Q <- c(-0.1677489, -0.7369231, -0.3682588, 0.5414703)
w <- c(0.1, 0.2, 0.3)
dT <- -.12
Qrot(Q,w,dT)
```

Qzero — *Generate zero-valued quaternions*

Description

`Qzero` generates zero-valued quaternions. `Qone` generates one-valued quaternions.

Usage

```
Qzero(n=NA)
```

Arguments

`n`	Optional integer for the number of generated quaternions, default = 1.

Value

`Q`	Zero or one-valued quaternion (Q) vector [q1, q2, q3, q4] or matrix n x 4.

Author(s)

Jose Gama

Examples

```
Qzero()
Qzero(5)
Qone()
Qone(5)
```

vectQrot — *Rotate a vector by a quaternion*

Description

`vectQrot` performs a vector rotation by a quaternion.

Usage

```
vectQrot(Q, rr)
```

Arguments

`Q`	Quaternion (Q) vector [q1, q2, q3, q4].
`rr`	Vector [x, y, z].

Value

Rotated vector [x, y, z].

Author(s)

Jose Gama

Examples

```
Q <- c(-0.1677489, -0.7369231, -0.3682588, 0.5414703)
v <- c(1, 2, 3)
vectQrot(Q, v)
```

---

z1 — *Quaternion multiplication*

Description

`%Q*%` performs a quaternion multiplication.

Usage

```
Q1 %Q*% Q2
```

Arguments

`Q1`	Quaternion (Q) vector [q1, q2, q3, q4].
`Q2`	Quaternion (Q) vector [q1, q2, q3, q4].

Value

`Q`	Quaternion result of multiplication (Q) vector [q1, q2, q3, q4].

Author(s)

Jose Gama

Examples

```
## Not run:
Q <- c(-0.1677489, -0.7369231, -0.3682588, 0.5414703)
Q %Q*% Q

## End(Not run)
```

z2 — *Quaternion division*

Description

`%Q/%` performs a quaternion division.

Usage

```
Q1 %Q/% Q2
```

Arguments

Q1	Quaternion (Q) vector [q1, q2, q3, q4].
Q2	Quaternion (Q) vector [q1, q2, q3, q4].

Value

Q	Quaternion result of division (Q) vector [q1, q2, q3, q4].

Author(s)

Jose Gama

Examples

```
## Not run:
Q <- c(-0.1677489, -0.7369231, -0.3682588, 0.5414703)
Q %Q/% Q

## End(Not run)
```

z3 — *Quaternion subtraction*

Description

`%Q-%` performs a quaternion subtraction.

Usage

```
Q1 %Q-% Q2
```

Arguments

Q1	Quaternion (Q) vector [q1, q2, q3, q4].
Q2	Quaternion (Q) vector [q1, q2, q3, q4].

Value

Q	Quaternion result of subtraction (Q) vector [q1, q2, q3, q4].

Author(s)

Jose Gama

Examples

```
## Not run:
Q <- c(-0.1677489, -0.7369231, -0.3682588, 0.5414703)
Q %Q-% Q

## End(Not run)
```

---

z4 *Quaternion addition*

Description

`%Q+%` performs a quaternion addition.

Usage

```
Q1 %Q+% Q2
```

Arguments

Q1	Quaternion (Q) vector [q1, q2, q3, q4].
Q2	Quaternion (Q) vector [q1, q2, q3, q4].

Value

Q	Quaternion sum (Q) vector [q1, q2, q3, q4].

Author(s)

Jose Gama

Examples

```
## Not run:
Q <- c(-0.1677489, -0.7369231, -0.3682588, 0.5414703)
Q %Q+% Q

## End(Not run)
```

z5 *Quaternion dot product*

Description

`%Q.%` performs a quaternion dot product.

Usage

```
Q1 %Q.% Q2
```

Arguments

Q1	Quaternion (Q) vector [q1, q2, q3, q4].
Q2	Quaternion (Q) vector [q1, q2, q3, q4].

Value

Q	Quaternion result of dot product (Q) vector [q1, q2, q3, q4].

Author(s)

Jose Gama

Examples

```
## Not run:
Q <- c(-0.1677489, -0.7369231, -0.3682588, 0.5414703)
Q %Q.% Q

## End(Not run)
```

Index