# Package 'nvctr'

October 13, 2022

**Type** Package

**Title** The n-vector Approach to Geographical Position Calculations using an Ellipsoidal Model of Earth

**Version** 0.1.4

**Maintainer** Enrico Spinielli <enrico.spinielli@eurocontrol.int>

**Description** The n-vector framework uses the normal vector to the Earth ellipsoid (called n-vector) as a non-singular position representation that turns out to be very convenient for practical position calculations. The n-vector is simple to use and gives exact answers for all global positions, and all distances, for both ellipsoidal and spherical Earth models. This package is a translation of the 'Matlab' library from FFI, the Norwegian Defence Research Establishment, as described in Gade (2010) <doi:10.1017/S0373463309990415>.

**License** MIT + file LICENSE

**URL** https://github.com/euctrl-pru/nvctr

**BugReports** https://github.com/euctrl-pru/nvctr/issues

**Imports** magrittr, pracma

**Suggests** bookdown, covr, geosphere, knitr, png, rmarkdown, spelling, testthat

**VignetteBuilder** knitr

**Encoding** UTF-8

**Language** en-US

**LazyData** true

**RoxygenNote** 7.1.1

**NeedsCompilation** no

**Author** Enrico Spinielli [aut, cre] (<https://orcid.org/0000-0001-8584-9131>), EUROCONTROL [cph, fnd]

**Repository** CRAN

**Date/Publication** 2020-10-28 14:00:02 UTC

## R topics documented:

along_track_distance . . . 2
altitude_azimuth_distance . . . 3
cross_track_distance . . . 4
cross_track_intersection . . . 5
deg . . . 6
lat_lon2n_E . . . 7
nvctr . . . 7
n_E2lat_lon . . . 8
n_E2R_EN . . . 9
n_EA_E_and_n_EB_E2p_AB_E . . . 9
n_EA_E_and_p_AB_E2n_EB_E . . . 11
n_EB_E2p_EB_E . . . 12
n_E_and_wa2R_EL . . . 13
p_EB_E2n_EB_E . . . 14
R2xyz . . . 15
R2zyx . . . 16
rad . . . 17
R_Ee . . . 17
R_EL2n_E . . . 18
R_EN2n_E . . . 19
unit . . . 20
xyz2R . . . 21
zyx2R . . . 22

**Index** **23**

---

`along_track_distance` *Compute the along-track distance from a great circle arc*

---

### Description

Compute the along-track distances of a body, ‘b‘ (for example a ground level projection position of an aircraft), from two geographical coordinates, ‘a1‘ and ‘a2‘ (for example an airport’s runway thresholds), of a great circle arc.

### Usage

```
along_track_distance(b, a1, a2)
```

### Arguments

| | |
|---|---|
| `b` | the geographical coordinates (WGS84) of a body: a vector of longitude, latitude (in decimal degrees) and eventually altitude (in meters) |
| `a1` | the geographical coordinates (WGS84) of one end of a great circle arc: a vector of longitude, latitude (in decimal degrees) and eventually altitude (in meters) |

| | |
|---|---|
| `a2` | the geographical coordinates (WGS84) of the other end of a great circle arc: a vector of longitude, latitude (in decimal degrees) and eventually altitude (in meters) |

### Value

the surface along-track distances from ‘b‘’s cross-track intersection to ‘a1‘ - ‘a2‘

### See Also

Other utilities: `altitude_azimuth_distance()`, `cross_track_distance()`, `cross_track_intersection()`

### Examples

```
## Not run:
b <- c(8.086135, 49.973942, 6401)
# EDDF: 07R (longitude, latitude, altitude)
a1 <- c(8.53417, 50.0275, 328)
# EDDF: 25L
a2 <- c(8.58653, 50.0401, 362)
along_track_distance(b, a1, a2)

## End(Not run)
```

---

## `altitude_azimuth_distance` *Calculate the altitude, azimuth and distance of B from A*

---

### Description

The altitude (elevation from the horizon), azimuth and distance of a point B from A are the coordinates of the Topocentric Coordinate System as typically used in astronomy to aim your telescope to a heavenly body. It can be also of use to know where an airplane is in the sky with respect to an observer on Earth.

### Usage

```
altitude_azimuth_distance(a, b)
```

### Arguments

| | |
|---|---|
| `a` | the observer position: a vector of longitude, latitude (in decimal degrees) and altitude (in meters) in WGS84 |
| `b` | the observed position: a vector of longitude, latitude (in decimal degrees) and altitude (in meters) in WGS84 |

## Value

the coordinates in North-East-Up of the observed, B, with respect to the observer A. A vector of altitude (elevation from the horizon) in decimal degrees, azimuth) in decimal degrees and distance in meters.

## See Also

Other utilities: `along_track_distance()`, `cross_track_distance()`, `cross_track_intersection()`

## Examples

```
## Not run:
# sensor (longitude, latitude, altitude)
a <- c(49.47, 7.697, 274)
# aircraft (longitude, latitude, altitude)
b <- c(49.52, 7.803, 6401)
altitude_azimuth_distance(a, b)

## End(Not run)
```

---

`cross_track_distance` *Compute the cross-track distance from a great circle arc*

---

## Description

Compute the cross-track distance of a body, ‘b‘ (for example a ground level projection position of an aircraft), from a great circle arc determined by two geographical coordinates, ‘a1‘ and ‘a2‘ (for example an airport’s runway thresholds).

## Usage

```
cross_track_distance(b, a1, a2)
```

## Arguments

| | |
|---|---|
| `b` | the geographical coordinates (WGS84) of a body: a vector of longitude, latitude (in decimal degrees) and eventually altitude (in meters) |
| `a1` | the geographical coordinates (WGS84) of one end of a great circle arc: a vector of longitude, latitude (in decimal degrees) and eventually altitude (in meters) |
| `a2` | the geographical coordinates (WGS84) of the other end of a great circle arc: a vector of longitude, latitude (in decimal degrees) and eventually altitude (in meters) |

## Value

the surface cross-track distance from ‘b‘ to the arc ‘a1‘ - ‘a2‘

### See Also

Other utilities: `along_track_distance()`, `altitude_azimuth_distance()`, `cross_track_intersection()`

### Examples

```
## Not run:
b <- c(8.086135, 49.973942, 6401)
# EDDF: 07R (longitude, latitude, altitude)
a1 <- c(8.53417, 50.0275, 328)
# EDDF: 25L
a2 <- c(8.58653, 50.0401, 362)
cross_track_distance(b, a1, a2)

## End(Not run)
```

## `cross_track_intersection` *Calculate cross-track intersection*

### Description

Calculate the cross-track intersection between the position of a body (i.e. an aircraft) and a great circle arc as defined by two points (i.e. the runway's thresholds).

### Usage

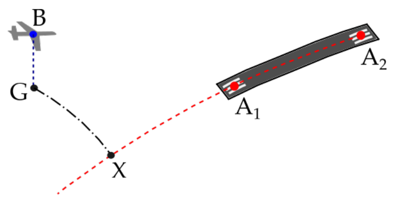
```
cross_track_intersection(b, a1, a2)
```

### Arguments

| | |
|---|---|
| `b` | coordinates of the body, b: a vector of longitude, latitude (in decimal degrees) and altitude (in meters) in WGS84 |
| `a1` | first coordinate of a great circle arc: a vector of longitude, latitude (in decimal degrees) and elevation (in meters) in WGS84 |
| `a2` | second coordinate of a great circle arc: a vector of longitude, latitude (in decimal degrees) and elevation (in meters) in WGS84 |

### Details

The cross-track intersection between the position of a body, B, (i.e. an aircraft) and a great circle arc as defined by two points, A1 and A2, (i.e. the runway's thresholds) is the intersection, X, of the above arc with the great circle arc passing through the ground projection of B, G, and perpendicular to A1-A2.



## Value

a WGS84 vector with longitude and latitude (decimal degrees)

## See Also

Other utilities: `along_track_distance()`, `altitude_azimuth_distance()`, `cross_track_distance()`

## Examples

```
## Not run:
# aircraft (longitude, latitude, altitude)
b <- c(8.086135, 49.973942, 6401)
# EDDF: 07R (longitude, latitude, altitude)
a1 <- c(8.53417, 50.0275, 328)
# EDDF: 25L
a2 <- c(8.58653, 50.0401, 362)
cross_track_intersection(b, a1, a2)

## End(Not run)
```

---

## deg — *Convert angle in radians to degrees*

---

## Description

Convert angle in radians to degrees

## Usage

```
deg(radians)
```

## Arguments

| | |
|---|---|
| radians | angle in radians. |

## Value

angle in degrees.

## See Also

`rad`.

Other helpers: `rad()`, `unit()`

## Examples

```
deg(pi/2)
```

## `lat_lon2n_E` *Convert (geodetic) latitude and longitude to n-vector*

### Description

Convert (geodetic) latitude and longitude to n-vector

### Usage

```
lat_lon2n_E(latitude, longitude)
```

### Arguments

| | |
|---|---|
| `latitude` | Geodetic latitude (rad) |
| `longitude` | Geodetic longitude (rad) |

### Value

n-vector decomposed in E (3x1 vector) (no unit)

### References

Kenneth Gade A Nonsingular Horizontal Position Representation. *The Journal of Navigation*, Volume 63, Issue 03, pp 395-417, July 2010.

### See Also

`n_E2lat_lon`.

### Examples

```
lat_lon2n_E(rad(1), rad(2))
```

## `nvctr` *nvctr: non-singular geographical position calculations*

### Description

nvctr provides functions to calculate geographical positions for both the ellipsoidal and spherical Earth models.

### Author(s)

**Maintainer**: Enrico Spinielli <`enrico.spinielli@eurocontrol.int`> (ORCID)

Other contributors:

- EUROCONTROL [copyright holder, funder]

## References

Kenneth Gade A Nonsingular Horizontal Position Representation. *The Journal of Navigation*, Volume 63, Issue 03, pp 395-417, July 2010.

## See Also

Useful links:

- `https://github.com/euctrl-pru/nvctr`
- Report bugs at `https://github.com/euctrl-pru/nvctr/issues`

---

`n_E2lat_lon` *Convert n-vector to latitude and longitude*

---

## Description

Convert n-vector to latitude and longitude

## Usage

```
n_E2lat_lon(n_E)
```

## Arguments

| | |
|---|---|
| `n_E` | n-vector decomposed in E (3x1 vector) (no unit) |

## Value

A vector of geodetic latitude and longitude (rad)

## References

Kenneth Gade A Nonsingular Horizontal Position Representation. *The Journal of Navigation*, Volume 63, Issue 03, pp 395-417, July 2010.

## See Also

`lat_lon2n_E`.

## Examples

```
n_E2lat_lon(c(1, 0, 0))
```

`n_E2R_EN` *Find the rotation matrix R_EN from n-vector*

## Description

Find the rotation matrix R_EN from n-vector

## Usage

```
n_E2R_EN(n_E)
```

## Arguments

| | |
|---|---|
| `n_E` | n-vector decomposed in E (3x1 vector) (no unit) |

## Value

The resulting rotation matrix (direction cosine matrix) (no unit)

## References

Kenneth Gade A Nonsingular Horizontal Position Representation. *The Journal of Navigation*, Volume 63, Issue 03, pp 395-417, July 2010.

## See Also

`R_EN2n_E`, `n_E_and_wa2R_EL` and `R_EL2n_E`.

## Examples

```
n_E2R_EN(c(1, 0, 0))
```

`n_EA_E_and_n_EB_E2p_AB_E` *Find the delta position from two positions A and B*

## Description

Given the n-vectors for positions A (`n_EA_E`) and B (`n_EB_E`), the output is the delta vector from A to B (`p_AB_E`).

## Usage

```
n_EA_E_and_n_EB_E2p_AB_E(
  n_EA_E,
  n_EB_E,
  z_EA = 0,
  z_EB = 0,
  a = 6378137,
  f = 1/298.257223563
)
```

## Arguments

| | |
|---|---|
| `n_EA_E` | n-vector of position A, decomposed in E (3x1 vector) (no unit) |
| `n_EB_E` | n-vector of position B, decomposed in E (3x1 vector) (no unit) |
| `z_EA` | Depth of system A, relative to the ellipsoid (z_EA = -height) (m, default 0) |
| `z_EB` | Depth of system B, relative to the ellipsoid (z_EB = -height) (m, default 0) |
| `a` | Semi-major axis of the Earth ellipsoid (m, default [WGS-84] 6378137) |
| `f` | Flattening of the Earth ellipsoid (no unit, default [WGS-84] 1/298.257223563) |

## Details

The calculation is exact, taking the ellipticity of the Earth into account. It is also nonsingular as both n-vector and p-vector are nonsingular (except for the center of the Earth). The default ellipsoid model used is WGS-84, but other ellipsoids (or spheres) might be specified via the optional parameters `a` and `f`.

## Value

Position vector from A to B, decomposed in E (3x1 vector)

## References

Kenneth Gade A Nonsingular Horizontal Position Representation. *The Journal of Navigation*, Volume 63, Issue 03, pp 395-417, July 2010.

## See Also

`n_EA_E_and_p_AB_E2n_EB_E`, `p_EB_E2n_EB_E` and `n_EB_E2p_EB_E`

## Examples

```
lat_EA <- rad(1); lon_EA <- rad(2); z_EA   <- 3
lat_EB <- rad(4); lon_EB <- rad(5); z_EB   <- 6

n_EA_E <- lat_lon2n_E(lat_EA, lon_EA)
n_EB_E <- lat_lon2n_E(lat_EB, lon_EB)

n_EA_E_and_n_EB_E2p_AB_E(n_EA_E, n_EB_E, z_EA, z_EB)
```

## n_EA_E_and_p_AB_E2n_EB_E
*Find position B from position A and delta*

### Description

Given the n-vector for position A (n_EA_E) and the position-vector from position A to position B (p_AB_E), the output is the n-vector of position B (n_EB_E) and depth of B (z_EB).

### Usage

```
n_EA_E_and_p_AB_E2n_EB_E(
  n_EA_E,
  p_AB_E,
  z_EA = 0,
  a = 6378137,
  f = 1/298.257223563
)
```

### Arguments

| | |
|---|---|
| n_EA_E | n-vector of position A, decomposed in E (3x1 vector) (no unit) |
| p_AB_E | Position vector from A to B, decomposed in E (3x1 vector) (m) |
| z_EA | Depth of system A, relative to the ellipsoid (z_EA = -height) (m, default 0) |
| a | Semi-major axis of the Earth ellipsoid (m, default [WGS-84] 6378137) |
| f | Flattening of the Earth ellipsoid (no unit, default [WGS-84] 1/298.257223563) |

### Details

The calculation is exact, taking the ellipticity of the Earth into account.

It is also nonsingular as both n-vector and p-vector are nonsingular (except for the center of the Earth). The default ellipsoid model used is WGS-84, but other ellipsoids (or spheres) might be specified.

### Value

a list with n-vector of position B, decomposed in E (3x1 vector) (no unit) and the depth of system B, relative to the ellipsoid (z_EB = -height)

### References

Kenneth Gade A Nonsingular Horizontal Position Representation. *The Journal of Navigation*, Volume 63, Issue 03, pp 395-417, July 2010.

### See Also

n_EA_E_and_n_EB_E2p_AB_E, p_EB_E2n_EB_E and n_EB_E2p_EB_E

## Examples

```
p_BC_B <- c(3000, 2000, 100)

# Position and orientation of B is given:
n_EB_E <- unit(c(1,2,3))  # unit to get unit length of vector
z_EB <- -400
R_NB  <- zyx2R(rad(10), rad(20), rad(30)) # yaw, pitch, and roll
R_EN <- n_E2R_EN(n_EB_E)
R_EB <- R_EN %*% R_NB

# Decompose the delta vector in E:
p_BC_E <- (R_EB %*% p_BC_B) %>% as.vector() # no transpose of R_EB, since the vector is in B

# Find the position of C, using the functions that goes from one
# position and a delta, to a new position:
(n_EB_E <- n_EA_E_and_p_AB_E2n_EB_E(n_EB_E, p_BC_E, z_EB))
```

---

`n_EB_E2p_EB_E` *Convert n-vector to cartesian position vector in meters*

---

## Description

The function converts the position of B (typically body) relative to E (typically Earth), the n-vector `n_EB_E` to cartesian position vector ("ECEF-vector"), `p_EB_E`, in meters.

## Usage

```
n_EB_E2p_EB_E(n_EB_E, z_EB = 0, a = 6378137, f = 1/298.257223563)
```

## Arguments

| | |
|---|---|
| `n_EB_E` | n-vector of position B, decomposed in E (3x1 vector) (no unit) |
| `z_EB` | Depth of system B, relative to the ellipsoid (z_EB = -height) (m, default 0) |
| `a` | Semi-major axis of the Earth ellipsoid (m, default [WGS-84] 6378137) |
| `f` | Flattening of the Earth ellipsoid (no unit, default [WGS-84] 1/298.257223563) |

## Details

The calculation is exact, taking the ellipticity of the Earth into account.

It is also nonsingular as both n-vector and p-vector are nonsingular (except for the center of the Earth). The default ellipsoid model used is WGS-84, but other ellipsoids (or spheres) might be specified via the optional parameters `a` and `f`.

## Value

Cartesian position vector from E to B, decomposed in E (3x1 vector) (m)

## References

Kenneth Gade A Nonsingular Horizontal Position Representation. *The Journal of Navigation*, Volume 63, Issue 03, pp 395-417, July 2010.

## See Also

p_EB_E2n_EB_E, n_EA_E_and_p_AB_E2n_EB_E and n_EA_E_and_n_EB_E2p_AB_E.

## Examples

```
n_EB_E  <- lat_lon2n_E(rad(1), rad(2))
n_EB_E2p_EB_E(n_EB_E)
```

---

n_E_and_wa2R_EL *Find R_EL from n-vector and wander azimuth angle*

---

## Description

Calculate the rotation matrix (direction cosine matrix) R_EL using n-vector (n_E) and the wander azimuth angle. When wander_azimuth = 0, we have that N = L (See Table 2 in Gade (2010) for details)

## Usage

```
n_E_and_wa2R_EL(n_E, wander_azimuth)
```

## Arguments

| | |
|---|---|
| n_E | n-vector decomposed in E (3x1 vector) (no unit) |
| wander_azimuth | The angle between L's x-axis and north, positive about L's z-axis (rad) |

## Value

The resulting rotation matrix (3x3) (no unit)

## References

Kenneth Gade A Nonsingular Horizontal Position Representation. *The Journal of Navigation*, Volume 63, Issue 03, pp 395-417, July 2010.

## See Also

R_EL2n_E, R_EN2n_E and n_E2R_EN.

## Examples

```
# Calculates the rotation matrix (direction cosine matrix) R_EL
# using n-vector (n_E) and the wander azimuth angle.
n_E <- c(1, 0, 0)
(R_EL <-  n_E_and_wa2R_EL(n_E, wander_azimuth = pi / 2))
```

---

## p_EB_E2n_EB_E *Convert cartesian position vector in meters to n-vector*

---

### Description

The position of B (typically body) relative to E (typically Earth) is given as cartesian position vector p_EB_E, in meters ("ECEF-vector").

### Usage

```
p_EB_E2n_EB_E(p_EB_E, a = 6378137, f = 1/298.257223563)
```

### Arguments

| | |
|---|---|
| p_EB_E | Cartesian position vector from E to B, decomposed in E (3x1 vector) (m) |
| a | Semi-major axis of the Earth ellipsoid (m, default [WGS-84] 6378137) |
| f | Flattening of the Earth ellipsoid (no unit, default [WGS-84] 1/298.257223563) |

### Details

The function converts to n-vector, n_EB_E and its depth, z_EB.

The calculation is exact, taking the ellipticity of the Earth into account. It is also nonsingular as both n-vector and p-vector are nonsingular (except for the center of the Earth). The default ellipsoid model used is WGS-84, but other ellipsoids (or spheres) might be specified.

### Value

n-vector representation of position B, decomposed in E (3x1 vector) (no unit) and depth of system B relative to the ellipsoid (z_EB = -height)

### References

Kenneth Gade A Nonsingular Horizontal Position Representation. *The Journal of Navigation*, Volume 63, Issue 03, pp 395-417, July 2010.

### See Also

n_EB_E2p_EB_E, n_EA_E_and_p_AB_E2n_EB_E and n_EA_E_and_n_EB_E2p_AB_E

## Examples

```
p_EB_E <- 6371e3 * c(0.9, -1, 1.1)
(n_EB_E <- p_EB_E2n_EB_E(p_EB_E))
```

---

`R2xyz` *Find the three rotation angles about new axes in the xyz order from a rotation matrix*

---

## Description

The angles (called Euler angles or Tait–Bryan angles) are defined by the following procedure of successive rotations: Given two arbitrary coordinate frames A and B, consider a temporary frame T that initially coincides with A. In order to make T align with B, we first rotate T an angle x about its x-axis (common axis for both A and T). Secondly, T is rotated an angle y about the NEW y-axis of T. Finally, T is rotated an angle z about its NEWEST z-axis. The final orientation of T now coincides with the orientation of B. The signs of the angles are given by the directions of the axes and the right hand rule.

## Usage

```
R2xyz(R_AB)
```

## Arguments

| | |
|---|---|
| `R_AB` | a 3x3 rotation matrix (direction cosine matrix) such that the relation between a vector v decomposed in A and B is given by: v_A = R_AB * v_B |

## Value

x,y,z Angles of rotation about new axes (rad)

## References

Kenneth Gade A Nonsingular Horizontal Position Representation. *The Journal of Navigation*, Volume 63, Issue 03, pp 395-417, July 2010.

## See Also

`xyz2R`, `R2zyx` and `zyx2R`.

## Examples

```
  R_AB <- matrix(
     c( 0.9980212 ,  0.05230407, -0.0348995 ,
       -0.05293623,  0.99844556, -0.01744177,
        0.03393297,  0.01925471,  0.99923861),
     nrow = 3, ncol = 3, byrow = TRUE)
R2xyz(R_AB)
```

`R2zyx` *Find the three angles about new axes in the zyx order from a rotation matrix*

## Description

The 3 angles `z`, `y`, `x` about new axes (intrinsic) in the order z-y-x are found from the rotation matrix `R_AB`. The angles (called Euler angles or Tait–Bryan angles) are defined by the following procedure of successive rotations:

1. Given two arbitrary coordinate frames A and B, consider a temporary frame T that initially coincides with A. In order to make T align with B, we first rotate T an angle z about its z-axis (common axis for both A and T).
2. Secondly, T is rotated an angle y about the NEW y-axis of T. Finally, T is rotated an angle x about its NEWEST x-axis.
3. The final orientation of T now coincides with the orientation of B.

The signs of the angles are given by the directions of the axes and the right hand rule.

## Usage

```
R2zyx(R_AB)
```

## Arguments

| | |
|---|---|
| `R_AB` | a 3x3 rotation matrix (direction cosine matrix) such that the relation between a vector v decomposed in A and B is given by: v_A = R_AB * v_B |

## Details

Note that if A is a north-east-down frame and B is a body frame, we have that z=yaw, y=pitch and x=roll.

## Value

z,y,x angles of rotation about new axes (rad)

## References

Kenneth Gade A Nonsingular Horizontal Position Representation. *The Journal of Navigation*, Volume 63, Issue 03, pp 395-417, July 2010.

## See Also

`zyx2R`, `xyz2R` and `R2xyz`.

## Examples

```
zyx2R(rad(1), rad(-2), rad(-3))
```

## rad *Convert angle in degrees to radians.*

### Description

Convert angle in degrees to radians.

### Usage

```
rad(degrees)
```

### Arguments

| | |
|---|---|
| `degrees` | angle in degrees. |

### Value

angle in radians

### See Also

`deg`.

Other helpers: `deg()`, `unit()`

### Examples

```
rad(30)
```

## R_Ee *Select the axes of the coordinate frame E*

### Description

This function returns the axes of the coordinate frame E (Earth-Centered, Earth-Fixed, ECEF).

### Usage

```
R_Ee(axes = "e")
```

## Arguments

| | |
|---|---|
| `axes` | Either 'e' or 'E'<br>• 'e': z-axis points to the North Pole along the Earth's rotation axis, x-axis points towards the point where latitude = longitude = 0. This choice is very common in many fields.<br>• 'E': x-axis points to the North Pole along the Earth's rotation axis, y-axis points towards longitude +90deg (east) and latitude = 0. (the yz-plane coincides with the equatorial plane). This choice of axis ensures that at zero latitude and longitude, frame N (North-East-Down) has the same orientation as frame E. If roll/pitch/yaw are zero, also frame B (forward-starboard-down) has this orientation. In this manner, the axes of frame E is chosen to correspond with the axes of frame N and B. The functions in this library originally used this option. |

## Details

There are two choices of E-axes that are described in Table 2 in Gade (2010):

- `e`: z-axis points to the North Pole and x-axis points to the point where latitude = longitude = 0. This choice is very common in many fields.
- `E`: x-axis points to the North Pole, y-axis points towards longitude +90deg (east) and latitude = 0. This choice of axis directions ensures that at zero latitude and longitude, N (North-East-Down) has the same orientation as E. If roll/pitch/yaw are zero, also B (Body, forward, starboard, down) has this orientation. In this manner, the axes of E is chosen to correspond with the axes of N and B.

## Value

rotation matrix defining the axes of the coordinate frame E as described in Table 2 in Gade (2010)

## References

Kenneth Gade (2010) A Nonsingular Horizontal Position Representation. *The Journal of Navigation*, Volume 63, Issue 03, pp 395-417, July 2010.

## Examples

```
R_Ee()
```

---

`R_EL2n_E` *Find n-vector from the rotation matrix (direction cosine matrix)* `R_EL`

---

## Description

Find n-vector from the rotation matrix (direction cosine matrix) `R_EL`

### Usage

```
R_EL2n_E(R_EL)
```

### Arguments

| | |
|---|---|
| `R_EL` | Rotation matrix (direction cosine matrix) (no unit) |

### Value

n-vector decomposed in E (3x1 vector) (no unit)

### References

Kenneth Gade A Nonsingular Horizontal Position Representation. *The Journal of Navigation*, Volume 63, Issue 03, pp 395-417, July 2010.

### See Also

`n_E2R_EN`, `R_EL2n_E` and `n_E_and_wa2R_EL`.

### Examples

```
R_EL <- matrix(
   c(-1, 0,  0,
      0, 1,  0,
      0, 0, -1),
   nrow = 3, ncol = 3, byrow = TRUE)
R_EL2n_E(R_EL)
```

---

## `R_EN2n_E` *Find n-vector from R_E*

---

### Description

Find n-vector from R_E

### Usage

```
R_EN2n_E(R_EN)
```

### Arguments

| | |
|---|---|
| `R_EN` | Rotation matrix (direction cosine matrix) (no unit) |

### Value

n-vector decomposed in E (3x1 vector) (no unit)

### References

Kenneth Gade A Nonsingular Horizontal Position Representation. *The Journal of Navigation*, Volume 63, Issue 03, pp 395-417, July 2010.

### See Also

`n_E2R_EN`, `R_EL2n_E` and `n_E_and_wa2R_EL`.

### Examples

```
R_EN <- matrix(
   c(-1, 0,  0,
      0, 1,  0,
      0, 0, -1),
   nrow = 3, ncol = 3, byrow = TRUE)
R_EL2n_E(R_EN)
```

---

## `unit` *Make input vector unit length, i.e. norm == 1*

---

### Description

Make input vector unit length, i.e. norm == 1

### Usage

```
unit(vector)
```

### Arguments

| | |
|---|---|
| `vector` | a vector |

### Value

a unit length vector

### See Also

Other helpers: `deg()`, `rad()`

### Examples

```
unit(c(1,2,3))
```

## xyz2R *Create a rotation matrix from 3 angles about new axes in the xyz order.*

### Description

The rotation matrix `R_AB` is created based on 3 angles `x`, `y` and `z` about new axes (intrinsic) in the order x-y-z. The angles (called Euler angles or Tait-Bryan angles) are defined by the following procedure of successive rotations:

1. Given two arbitrary coordinate frames `A` and `B`, consider a temporary frame `T` that initially coincides with `A`. In order to make `T` align with `B`, we first rotate `T` an angle `x` about its x-axis (common axis for both `A` and `T`).
2. Secondly, `T` is rotated an angle `y` about the NEW y-axis of `T`.
3. Finally, codeT is rotated an angle `z` about its NEWEST z-axis. The final orientation of `T` now coincides with the orientation of `B`.

The signs of the angles are given by the directions of the axes and the right hand rule.

### Usage

```
xyz2R(x, y, z)
```

### Arguments

| | |
|---|---|
| `x` | Angle of rotation about new x axis (rad) |
| `y` | Angle of rotation about new y axis (rad) |
| `z` | Angle of rotation about new z axis (rad) |

### Value

3x3 rotation matrix (direction cosine matrix) such that the relation between a vector v decomposed in A and B is given by: v_A = R_AB * v_B

### References

Kenneth Gade A Nonsingular Horizontal Position Representation. *The Journal of Navigation*, Volume 63, Issue 03, pp 395-417, July 2010.

### See Also

`R2xyz`, `zyx2R` and `R2zyx`.

### Examples

```
xyz2R(rad(10), rad(20), rad(30))
```

## zyx2R *Create a rotation matrix from 3 angles about new axes in the zyx order.*

### Description

The rotation matrix `R_AB` is created based on 3 angles `z`, `y` and `x` about new axes (intrinsic) in the order z-y-x. The angles (called Euler angles or Tait–Bryan angles) are defined by the following procedure of successive rotations:

1. Given two arbitrary coordinate frames A and B, consider a temporary frame T that initially coincides with A. In order to make T align with B, we first rotate T an angle z about its z-axis (common axis for both A and T).
2. Secondly, T is rotated an angle y about the NEW y-axis of T.
3. Finally, T is rotated an angle x about its NEWEST x-axis. The final orientation of T now coincides with the orientation of B.

The signs of the angles are given by the directions of the axes and the right hand rule. Note that if A is a north-east-down frame and B is a body frame, we have that z=yaw, y=pitch and x=roll.

### Usage

```
zyx2R(z, y, x)
```

### Arguments

| | |
|---|---|
| `z` | Angle of rotation about new z axis |
| `y` | Angle of rotation about new y axis |
| `x` | Angle of rotation about new x axis |

### Value

3x3 rotation matrix R_AB (direction cosine matrix) such that the relation between a vector v decomposed in A and B is given by: v_A = R_AB * v_B

### References

Kenneth Gade A Nonsingular Horizontal Position Representation. *The Journal of Navigation*, Volume 63, Issue 03, pp 395-417, July 2010.

### See Also

`R2zyx`, `xyz2R` and `R2xyz`.

### Examples

```
zyx2R(rad(30), rad(20), rad(10))
```

# Index

∗ **helpers**
    deg, 6
    rad, 17
    unit, 20
∗ **utilities**
    along_track_distance, 2
    altitude_azimuth_distance, 3
    cross_track_distance, 4
    cross_track_intersection, 5

along_track_distance, 2, *4–6*
altitude_azimuth_distance, *3*, 3, *5, 6*

cross_track_distance, *3, 4*, 4, *6*
cross_track_intersection, *3–5*, 5

deg, 6, *17*, *20*

lat_lon2n_E, 7, *8*

n_E2lat_lon, *7*, 8
n_E2R_EN, 9, *13*, *19, 20*
n_E_and_wa2R_EL, *9*, 13, *19, 20*
n_EA_E_and_n_EB_E2p_AB_E, 9, *11*, *13, 14*
n_EA_E_and_p_AB_E2n_EB_E, *10*, 11, *13, 14*
n_EB_E2p_EB_E, *10, 11*, 12, *14*
nvctr, 7
nvctr-package *(*nvctr*)*, 7

p_EB_E2n_EB_E, *10, 11*, *13*, 14

R2xyz, 15, *16*, *21, 22*
R2zyx, *15*, 16, *21, 22*
R_Ee, 17
R_EL2n_E, *9*, *13*, 18, *19, 20*
R_EN2n_E, *9*, *13*, 19
rad, *6*, 17, *20*

unit, *6*, *17*, 20

xyz2R, *15, 16*, 21, *22*

zyx2R, *15, 16*, *21*, 22