Package 'GeodesiCL'

May 25, 2021

Type Package

Title Geometric Geodesy Functions

Version 1.0.0

Date 2020-12-10

Description Geometric geodesy functions applied to most common ellipsoids. This package was created to streamline and facilitate their work for surveyors, geographers, and everything related to geosciences.

License GPL-3

URL https://github.com/diegoalarc/GeodesiCL

BugReports https://github.com/diegoalarc/GeodesiCL/issues

Depends R (>= 3.5.0)

Encoding UTF-8

Language en-US

LazyData true

RoxygenNote 7.1.1

Imports sp, readr, dplyr, rgdal, utils, magrittr, methods, tibble, leafpop, htmltools, leaflet, profvis, mapview

Suggests testthat (>= 3.0.0), rmarkdown, knitr, covr

Config/testthat/edition 3

VignetteBuilder knitr

NeedsCompilation no

Author Diego Alonso Alarcon Diaz [cre, aut], Karla Francisca Riquelme Carrillo [aut]

Maintainer Diego Alonso Alarcon Diaz <diego.alarcondiaz@gmail.com>

Repository CRAN

Date/Publication 2021-05-25 12:20:02 UTC

R topics documented:

`arch` *Rope reduction to elliptical arch.*

Description

With this function it is possible to perform a reduction from chord to elliptical bow.

Usage

```
arch(x, digits = 4)
```

Arguments

`x`	Rope
`digits`	Number of digits the seconds are `rounded` to. DEFAULT: 4

Value

value

Examples

```
# Test data
ROPE <- 50000

value <- arch(ROPE, digits = 4)
print(value)
```

`cartesian` *To convert from Geographic coordinate to Cartesian coordinate.*

Description

With this function it is possible to convert from Geographic coordinate to Cartesian coordinate and obtain the decimal precision that you assign.

Usage

```
cartesian(a, longlat_df, digits = 4)
```

Arguments

`a`	Selection of Ellipsoid.
`longlat_df`	Point name, Sexagesimal longitude and latitude as dataframe.
`digits`	Number of digits the seconds are `round`ed to. DEFAULT: 4

Value

data.frame with the data in the following order: "Pt", "X", "Y", "Z".

Note

create data frame of epsg codes by epsg <- rgdal::make_EPSG()

References

https://github.com/OSGeo/PROJ & https://github.com/cran/rgdal

Examples

```
# Point name
Pto <- "St1"

# Longitude
g <- -71
m <- 18
s <- 44.86475

# Value in sexagesimal
```

```
sexa_long <- sexagesimal(g,m,s)

# Latitude
g <- -33
m <- 38
s <- 30.123456

# Value in sexagesimal
sexa_lat <- sexagesimal(g, m, s)
print(sexa_lat)

# ELLIPSOIDAL HEIGHT (h)
h <- 31.885

# Longitude and Latitude as data.frame
longlat_df <- data.frame(Pto, sexa_long, sexa_lat, h)

# To know the ellipsoids and the order open the Ellipsoids in the package and look for it number
Ellip <- Ellipsoids
#View(Ellip)

# We choose the number 5 which is GRS80
value <- cartesian(5, longlat_df, digits = 4)
print(value)
```

E2 *To calculate 1-e^2.*

Description

To calculate the value for the function 1-e^2.

Usage

```
E2(x, digits = 4)
```

Arguments

x	Selection of Ellipsoid.
digits	Number of digits the seconds are rounded to. DEFAULT: 4

Value

value

Note

create data frame of epsg codes by epsg <- rgdal::make_EPSG()

References

https://github.com/OSGeo/PROJ & https://github.com/cran/rgdal

Examples

```
# To know the ellipsoids and the order open the Ellipsoids in the package and look for it number
Ellip <- Ellipsoids
#View(Ellip)

# We choose the number 5 which is GRS80
value <- E2(5, digits = 4)
print(value)
```

E3 *To calculate 1-e^2*sen(lat)^2.*

Description

To calculate the value for the function 1-e^2*sen(lat)^2.

Usage

```
E3(x, y, digits = 4)
```

Arguments

x	Selection of Ellipsoid.
y	Latitude in radians.
digits	Number of digits the seconds are rounded to. DEFAULT: 4

Value

value

Note

create data frame of epsg codes by epsg <- rgdal::make_EPSG()

References

https://github.com/OSGeo/PROJ & https://github.com/cran/rgdal

Examples

```
# Lat
g <- -33
m <- 38
s <- 30.123456

rad_lat <- radians(g, m, s)
print(rad_lat)

# To know the ellipsoids and the order open the Ellipsoids in the package and look for it number
Ellip <- Ellipsoids
#View(Ellip)

# We choose the number 5 which is GRS80
value <- E3(5, rad_lat, digits = 4)
print(value)
```

`Ellipsoids` *Contains the values for conversion of the Ellipsoids PSAD-56, SAD-69, WGS-84, GRS-80 SIRGAS.*

Description

Contains the values for conversion of the Ellipsoids PSAD-56, SAD-69, WGS-84, GRS-80 SIRGAS.

Usage

```
Ellipsoids
```

Format

A data frame with 20 columns:

Description Contains the description of each ellipsoid.

Ellipsoids Contains the names of each ellipsoid.

a Semi major axis of each ellipsoid.

1/f 1/Flattening of each ellipsoid.

f Flattening of each ellipsoid.

b Semi minor axis of each ellipsoid.

e^2 1st eccentricity squared of each ellipsoid.

e'^2 2nd eccentricity squared of each ellipsoid.

A A Constant.

B B Constant.

C C Constant.

D D Constant.

E E Constant.

F F Constant.

Alfa Alfa Constant.

Beta Beta Constant.

Gamma Gamma Constant.

Delta Delta Constant.

Epsilon Epsilon Constant.

Zeta Zeta Constant.

References

https://github.com/diegoalarc/GeodesiCL

Examples

```
## Not run:
 Ellipsoids

## End(Not run)
```

`geodesic` *To convert from Cartesian coordinate to Geographic coordinate.*

Description

With this function it is possible to convert from Cartesian coordinate to Geographic coordinate and obtain the decimal precision that you assign.

Usage

```
geodesic(a, XYZ_df, digits = 4)
```

Arguments

`a`	Selection of Ellipsoid.
`XYZ_df`	Sexagesimal longitude and latitude as dataframe.
`digits`	Number of digits the seconds are `round`ed to. DEFAULT: 4

Value

data.frame with the data in the following order: "Pt", "Lat", "Lon", "H".

Note

create data frame of epsg codes by epsg <- rgdal::make_EPSG()

References

https://github.com/OSGeo/PROJ & https://github.com/cran/rgdal

Examples

```
# Point name
Pto <- "St1"

# Cartesian data
X <- 1711591.78090565
Y <- -5060304.1659587
Z <- -3473256.69328603

# Pto, X, Y and Z as data.frame
XYZ_df <- as.data.frame(cbind(Pto, X, Y, Z))

# To know the ellipsoids and the order open the Ellipsoids in the package and look for it number
Ellip <- Ellipsoids
#View(Ellip)

# We choose the number 5 which is GRS80
value <- geodesic(5, XYZ_df, digits = 4)
print(value)
```

`geodis` *Reduction of the horizontal distance to the Ellipsoid.*

Description

With this function it is possible to perform a reduction of the horizontal distance to the Ellipsoid.

Usage

```
geodis(Data_fm, digits = 4)
```

Arguments

`Data_fm`	Point name, Ellipsoidal height and Horizontal distance as dataframe.
`digits`	Number of digits the seconds are `round`ed to. DEFAULT: 4

Value

data.frame with the data in the following order: "Pt", "Kh(h)", "GEODESIC DISTANCE".

Examples

```
# Point name
Pto <- "St1"

# Ellipsoidal height
h <- 2500

# Horizontal distance
Dhz <- 728.5

# Ellipsoidal height and Horizontal distance as data.frame
Ellips_Horzdist_df <- data.frame(Pto, h, Dhz)

value <- geodis(Ellips_Horzdist_df, digits = 4)
print(value)
```

`LongLatToUTM` *To convert from Geographic coordinate to UTM coordinate.*

Description

With this function it is possible to convert from Geographic coordinate to UTM coordinate. It is also possible to convert from other coordinate reference systems by selecting their EPGS number to UTM coordinate. Review notes and references.

Usage

```
LongLatToUTM(longlat_df, crs = 4326, units = "m", digits = 4)
```

Arguments

`longlat_df`	Point name, Sexagesimal longitude and latitude as dataframe.
`crs`	EPGS number of the new coordinate reference system to transform. DEFAULT: 4326 (WGS84)
`units`	Select units for UTM to work. DEFAULT: 'm'
`digits`	Number of digits the seconds are `rounded` to. DEFAULT: 4

Value

a list with a data.frame and leaflet map.

Note

create data frame of epsg codes by epsg <- rgdal::make_EPSG()

References

https://github.com/OSGeo/PROJ & https://github.com/cran/rgdal

See Also

data.frame

Examples

```
# Point name
Pto <- "St1"

# Longitude
g <- -71
m <- 18
s <- 44.86475

# Value in sexagesimal
sexa_long <- sexagesimal(g, m, s)

# Latitude
g1 <- -33
m1 <- 12
s1 <- 27.11457

# Value in sexagesimal
sexa_lat <- sexagesimal(g1, m1, s1)

# Longitude and Latitude as data.frame
longlat_df <- data.frame(Pto,sexa_long,sexa_lat)

value <- LongLatToUTM(longlat_df, crs = 4326, units = 'm', digits = 4)
print(value)
```

M *To calculate the value of M.*

Description

To calculate the value for the function of M.

Usage

```
M(x, y, digits = 4)
```

Arguments

x	Selection of Ellipsoid.
y	Latitude in radians.
digits	Number of digits the seconds are rounded to. DEFAULT: 4

Value

value

Note

create data frame of epsg codes by epsg <- rgdal::make_EPSG()

References

https://github.com/OSGeo/PROJ & https://github.com/cran/rgdal

Examples

```
# Latitude
g <- -33
m <- 38
s <- 30.123456

rad_lat <- radians(g, m, s)
print(rad_lat)

# To know the ellipsoids and the order open the Ellipsoids in the package and look for it number
Ellip <- Ellipsoids
#View(Ellip)

# We choose the number 5 which is GRS80
value <- M(5, rad_lat, digits = 4)
print(value)
```

N *To calculate the value of N.*

Description

To calculate the value for the function of N.

Usage

```
N(x, y, digits = 4)
```

Arguments

x	Selection of Ellipsoid.
y	Latitude in radians.
digits	Number of digits the seconds are rounded to. DEFAULT: 4

Value

value

Note

create data frame of epsg codes by epsg <- rgdal::make_EPSG()

References

https://github.com/OSGeo/PROJ & https://github.com/cran/rgdal

Examples

```
# Latitude
g <- -33
m <- 38
s <- 30.123456

rad_lat <- radians(g, m, s)
print(rad_lat)

# To know the ellipsoids and the order open the Ellipsoids in the package and look for it number
Ellip <- Ellipsoids
#View(Ellip)

# We choose the number 5 which is GRS80
value <- N(5, rad_lat, digits = 4)
print(value)
```

PlotLongLat *To plot using Geographic coordinate WGS84.*

Description

With this function it is possible to plot using Longitude and Latitude (Geographic coordinate WGS84).

Usage

```
PlotLongLat(longlat_df)
```

Arguments

longlat_df	Point name, Sexagesimal longitude and latitude as dataframe.

Value

leaflet map.

See Also

data.frame

Examples

```
# Point name
Pto <- "St1"

# Longitude
g <- -71
m <- 18
s <- 44.86475

# Value in sexagesimal
sexa_long <- sexagesimal(g, m, s)

# Latitude
g1 <- -33
m1 <- 12
s1 <- 27.11457

# Value in sexagesimal
sexa_lat <- sexagesimal(g1, m1, s1)

# Longitude and Latitude as data.frame
longlat_df <- data.frame(Pto,sexa_long,sexa_lat)

value <- PlotLongLat(longlat_df)
print(value)
```

`radians` *To convert separated data in Degrees Minutes and Seconds to Radians.*

Description

With this function it is possible to convert separated data in Degrees Minutes and Seconds to Radians.

Usage

```
radians(x, y, z, digits = 4)
```

Arguments

`x`	Value of Degree in Latitude or Longitude.
`y`	Value of Minute in Latitude or Longitude.
`z`	Value of Seconds in Latitude or Longitude.
`digits`	Number of digits the seconds are `round`ed to. DEFAULT: 4

Value

value

Examples

```
# Lat
g <- -33
m <- 38
s <- 30.123456
value <- radians(g, m, s, digits = 4)
print(value)
```

`read_data` *Read data in csv.*

Description

Read data in csv.

Usage

```
read_data(x, digits = 4)
```

Arguments

`x`	the name of the file which the data are to be read from. this is a `data.frame` in format csv.
`digits`	Number of digits the seconds are `round`ed to. DEFAULT: 4

Value

a `tibble`

See Also

Package 'tibble' as 'readr'

Examples

```
# Test data
csv <- system.file("extdata", "test.csv", package = "GeodesiCL")
data_test <- read_data(csv)
```

rSL *To calculate the value of r, S and L.*

Description

With this function it is possible to calculate the value of r, S and L.

Usage

```
rSL(x, pto_lat, digits = 4)
```

Arguments

x	Selection of Ellipsoid to work between 1 = 'PSAD-56', 2 = 'SAD-69', 3 = 'WGS-84', 4 ='GRS-80 (SIRGAS)'.
pto_lat	Point name and the latitude in radians as data.frame.
digits	Number of digits the seconds are rounded to. DEFAULT: 4

Value

data.frame with the data in the following order: "Pt", "r", "S", "L".

Examples

```
# Point name
Pto <- "St1"

# Latitude
g <- -33
m <- 38
s <- 30.123456

rad_lat <- radians(g, m, s)
print(rad_lat)

# Ellipsoidal height and Horizontal distance as data.frame
pto_lat <- data.frame(Pto, rad_lat)

# To know the ellipsoids and the order open the Ellipsoids in the package and look for it number
Ellip <- Ellipsoids
#View(Ellip)

# We choose the number 47 which is WGS84
value <- rSL(47, pto_lat, digits = 4)
print(value)
```

scalfactor *To calculate scale factor calculation.*

Description

With this function it is possible to calculate the scale factor.

Usage

```
scalfactor(EAST_Geodist_df, digits = 4)
```

Arguments

EAST_Geodist_df	Point name, East coordinate and Geodesic distance as data.frame.
digits	Number of digits the seconds are rounded to. DEFAULT: 4

Value

data.frame with the data in the following order: "Pt", "X", "K UTM","D UTM", "DIF D-S", "PPM".

Examples

```
# Point name
Pto <- "St1"

# East coordinate.
EAST <- 224200

# Ellipsoidal height
h <- 2500

# Horizontal distance
Dhz <- 728.5

# Ellipsoidal height and Horizontal distance as data.frame
Ellips_Horzdist_df <- as.data.frame(cbind(Pto, h, Dhz))

geodis <- geodis(Ellips_Horzdist_df, digits = 4)

# East coordinate and Geodesic distance as data.frame
EAST_Geodist_df <- as.data.frame(cbind(Pto, EAST, geodis[,3]))

value <- scalfactor(EAST_Geodist_df, digits = 4)
print(value)
```

`sexagesimal` *To convert separated data in Degrees Minutes and Seconds to Decimal degrees.*

Description

With this function it is possible to convert separated data in Degrees Minutes and Seconds to Decimal degrees.

Usage

```
sexagesimal(x, y, z, digits = 4)
```

Arguments

`x`	Value of Degree in Latitude or Longitude.
`y`	Value of Minute in Latitude or Longitude.
`z`	Value of Seconds in Latitude or Longitude.
`digits`	Number of digits the seconds are `round`ed to. DEFAULT: 4

Value

value

Examples

```
# Latitude
g <- -33
m <- 38
s <- 30.123456
value <- sexagesimal(g, m, s, digits = 4)
print(value)
```

`Sin_1` *Contains the values of sin 1".*

Description

Contains the values of sin 1".

Usage

```
Sin_1
```

Format

A numeric with the value of sin 1"., which is:

Sin_1 Value of sin 1".

Examples

```
## Not run:
 Sin_1

## End(Not run)
```

TO_TM *To convert from Geographic coordinate to TM.*

Description

With this function it is possible to convert from Geographic coordinate to TM using the Central meridian, Scale factor Ko, False East, False North and obtain the decimal precision that you assign.

Usage

```
TO_TM(a = 47, longlat_df, d, e, f, g, digits = 4)
```

Arguments

a	Selection of Ellipsoid.
longlat_df	Sexagesimal longitude and latitude as dataframe.
d	Central meridian.
e	Scale factor Ko.
f	False East (FE).
g	False North (FN).
digits	Number of digits the seconds are rounded to. DEFAULT: 4

Value

data.frame with the data in the following order: "East", "North", "X", "Y".

Note

create data frame of epsg codes by epsg <- rgdal::make_EPSG()

References

https://github.com/OSGeo/PROJ & https://github.com/cran/rgdal

Examples

```
# Test data
CM <- -69.00000
SC_FACTOR_Ko <- 0.99960
FE <- 500000.00000
FN <- 10000000.00000

# Point name
Pto <- "St1"

# Longitude
g <- -71
m <- 18
s <- 44.86475

# Value in sexagesimal
sexa_long <- sexagesimal(g, m, s)

# Latitude
g1 <- -33
m1 <- 12
s1 <- 27.11457

# Value in sexagesimal
sexa_lat <- sexagesimal(g1, m1, s1)

# Longitude and Latitude as data.frame
longlat_df <- as.data.frame(cbind(Pto,sexa_long,sexa_lat))

# ELLIPSOIDAL HEIGHT (h)
h <- 31.885

# To know the ellipsoids and the order open the Ellipsoids in the package and look for it number
Ellip <- Ellipsoids
#View(Ellip)

# We choose the number 47 which is WGS84
value <- TO_TM(a = 47, longlat_df, CM, SC_FACTOR_Ko, FE, FN, digits = 4)
print(value)
```

UTMtoLongLat *To convert from UTM to Geographic coordinate.*

Description

With this function it is possible to convert from UTM coordinate to Geographic coordinate. It is also possible to convert to other coordinate reference systems by selecting their EPGS number. Review notes and references.

Usage

```
UTMtoLongLat(utm_df, zone, hemisphere = "south", crs = 4326, digits = 4)
```

Arguments

utm_df	Point name, East and North UTM in a data.frame.
zone	Select UTM zone to work.
hemisphere	select between north or south (written in lowercase). DEFAULT: "south"
crs	EPGS number of the new coordinate reference system to transform. DEFAULT: 4326 (WGS84)
digits	Number of digits the seconds are rounded to. DEFAULT: 4

Value

a list with a data.frame and leaflet map.

Note

create data frame of epsg codes by epsg <- rgdal::make_EPSG()

References

https://github.com/OSGeo/PROJ & https://github.com/cran/rgdal

See Also

data.frame

Examples

```
# Load test data from the package
csv <- system.file("extdata", "test.csv", package = "GeodesiCL")
data_test <- read_data(csv)

# Zone
zone <- 19

# Hemisphere could be "north" or "south"
hemisphere <- "south"

value <- UTMtoLongLat(data_test, zone, hemisphere = "south", crs = 4326, digits = 4)
print(value)
```

`UTM_zone_hemisphere` *To find the zone hemisphere from Longitude and Latitude the UTM zone.*

Description

With this function it is possible to find the zone hemisphere from Geographic coordinate to obtain the UTM zone.

Usage

```
UTM_zone_hemisphere(x, y)
```

Arguments

x	Sexagesimal longitude.
y	Sexagesimal latitude.

Value

value

Examples

```
#' # Longitude
g <- -71
m <- 18
s <- 44.86475

# Value in sexagesimal
sexa_long <- sexagesimal(g, m, s)

# Latitude
g1 <- -33
m1 <- 12
s1 <- 27.11457

# Value in sexagesimal
sexa_lat <- sexagesimal(g1, m1, s1)

value <- UTM_zone_hemisphere(sexa_long, sexa_lat)
print(value)
```

Index