Package 'parzer'

November 6, 2026

Type Package

Title Parse Messy Geographic Coordinates

Description Parse messy geographic coordinates from various character formats to decimal degree numeric values. Parse coordinates into their parts (degree, minutes, seconds); calculate hemisphere from coordinates; pull out individually degrees, minutes, or seconds; add and subtract degrees, minutes, and seconds. C++ code herein originally inspired from code written by Jeffrey D. Bogan, but then completely re-written.

Version 0.5.0

Maintainer Alban Sagouis <sagouis@pm.me>

License MIT + file LICENSE

URL https://github.com/ropensci/parzer, https://docs.ropensci.org/parzer/

BugReports https://github.com/ropensci/parzer/issues

Imports Rcpp (>= 1.0.2)

Suggests bench, testthat (>= 3.0.0), quarto, sf, leaflet, callr, pkgbuild, pkgload, withr

Config/testthat/edition 3

Config/testthat/parallel false

Config/testthat/start-first *utils*, *parse_parts*, *parse*, *dms*, *benchmarks*

LinkingTo Rcpp

VignetteBuilder quarto

Encoding UTF-8

Language en-US

X-schema.org-applicationCategory Geospatial

X-schema.org-keywords geospatial, data, latitude, longitude, parser, coordinates

X-schema.org-isPartOf https://ropensci.org

Config/roxygen2/version 8.1.1

NeedsCompilation yes

Author Scott Chamberlain [aut] (ORCID:
<https://orcid.org/0000-0003-1444-9135>),
Alban Sagouis [aut, cre] (ORCID:
<https://orcid.org/0000-0002-3827-1063>),
Alec L. Robitaille [ctb] (ORCID:
<https://orcid.org/0000-0002-4706-1762>),
Maëlle Salmon [ctb] (ORCID: <https://orcid.org/0000-0002-2815-0399>),
Hiroaki Yutani [ctb],
Jeffrey Bogan [ctb] (C++ code originally from Jeffrey Bogan, but
completely re-written),
Julien Brun [rev] (ORCID: <https://orcid.org/0000-0002-7751-6238>,
Julien Brun reviewed the package, see
https://github.com/ropensci/onboarding/issues/341),
Maria Munafó [rev] (Maria Munafó reviewed the package, see
https://github.com/ropensci/onboarding/issues/341),
rOpenSci [fnd] (ROR: <https://ror.org/019jywm96>),
German Centre for Integrative Biodiversity Research (iDiv)
Halle-Jena-Leipzig [fnd] (ROR: <https://ror.org/01jty7g66>)

Repository CRAN

Date/Publication 2026-11-06 13:00:02 UTC

Contents

parzer-package 2
dms 4
parse_hemisphere 5
parse_lat 6
parse_llstr 8
parse_lon 9
parse_lon_lat 10
parse_parts 11

Index **13**

`parzer-package` *parzer: Parse Messy Geographic Coordinates*

Description

Parse messy geographic coordinates from various character formats to decimal degree numeric values. Parse coordinates into their parts (degree, minutes, seconds); calculate hemisphere from coordinates; pull out individually degrees, minutes, or seconds; add and subtract degrees, minutes, and seconds. C++ code herein originally inspired from code written by Jeffrey D. Bogan, but then completely re-written.

Details

[Stable]

parse geographic coordinates

Author(s)

Maintainer: Alban Sagouis <sagouis@pm.me> (ORCID)

Authors:

- Alban Sagouis <sagouis@pm.me> (ORCID)
- Scott Chamberlain <sckott@protonmail.com> (ORCID)

Other contributors:

- Alec L. Robitaille (ORCID) [contributor]
- Maëlle Salmon (ORCID) [contributor]
- Hiroaki Yutani [contributor]
- Jeffrey Bogan (C++ code originally from Jeffrey Bogan, but completely re-written) [contributor]
- Julien Brun (ORCID) (Julien Brun reviewed the package, see https://github.com/ropensci/onboarding/issues/341) [reviewer]
- Maria Munafó (Maria Munafó reviewed the package, see https://github.com/ropensci/onboarding/issues/341) [reviewer]
- rOpenSci (ROR) [funder]
- German Centre for Integrative Biodiversity Research (iDiv) Halle-Jena-Leipzig (ROR) [funder]

See Also

Useful links:

- https://github.com/ropensci/parzer
- https://docs.ropensci.org/parzer/
- Report bugs at https://github.com/ropensci/parzer/issues

dms *extract degree, minutes, and seconds*

Description

extract degree, minutes, and seconds

Usage

```
pz_degree(lon = NULL, lat = NULL)

pz_minute(lon = NULL, lat = NULL)

pz_second(lon = NULL, lat = NULL)

## S3 method for class 'pz'
print(x, ...)

pz_d(x)

pz_m(x)

pz_s(x)

## S3 method for class 'pz'
e1 + e2

## S3 method for class 'pz'
e1 - e2

## S3 method for class 'pz'
e1 / e2

## S3 method for class 'pz'
e1 * e2
```

Arguments

lon, lat	(numeric/integer/character) one or more longitude or latitude values. values are internally validated. only one of lon or lat accepted
x	(integer) an integer representing a degree, minute or second
...	print dots
e1, e2	objects of class pz, from using pz_d(), pz_m(), or pz_s()

Details

Mathematics operators are exported for +, -, /, and *, but / and * are only exported with a stop message to say it's not supported; otherwise you'd be allow to divide degrees by minutes, leading to nonsense.

Value

pz_degree: integer, pz_minute: integer, pz_second: numeric, pz_d: numeric, pz_m: numeric, pz_s: numeric (adding/subtracting these also gives numeric)

Examples

```
# extract parts of a coordinate value
pz_degree(-45.23323)
pz_minute(-45.23323)
pz_second(-45.23323)

pz_degree(lon = 178.23423)
pz_minute(lon = 178.23423)
pz_second(lon = 178.23423)
## Not run:
pz_degree(lat = c(45.23323, "40:25:6N", "40° 25´ 5.994 S"))
pz_minute(lat = c(45.23323, "40:25:6N", "40° 25´ 5.994 S"))
pz_second(lat = c(45.23323, "40:25:6N", "40° 25´ 5.994 S"))

# invalid
pz_degree(445.23323)

# add or subtract
pz_d(31)
pz_m(44)
pz_s(3)
pz_d(31) + pz_m(44)
pz_d(-31) - pz_m(44)
pz_d(-31) + pz_m(44) + pz_s(59)
pz_d(31) - pz_m(44) + pz_s(59)
pz_d(-121) + pz_m(1) + pz_s(33)
unclass(pz_d(31) + pz_m(44) + pz_s(59))

## End(Not run)
```

---

parse_hemisphere *get hemisphere from long/lat coordinates*

Description

BEWARE: EXPERIMENTAL

Usage

```
parse_hemisphere(lon, lat)
```

Arguments

lon	(character/numeric/integer) one or more longitude values
lat	(character/numeric/integer) one or more latitude values

Details

length(lon) == length(lat)

Value

character vector of quadrants, one of: NE, NW, SE, SW. if one of the coordinate values is invalid, and one is valid, you get a length 1 string. if both coordinate values are bad, you get a zero length string.

Warnings are thrown on invalid values

Examples

```
# NE
parse_hemisphere("74.123E", "45N54.2356")
## Not run:
# NW
parse_hemisphere(-120, 40.4183318)
# SW
parse_hemisphere(-120, -40.4183318)
# SE
parse_hemisphere(120, -40.4183318)

# bad inputs, get one of the two strings
parse_hemisphere(-181, -40.4183318)
parse_hemisphere(-120, -192.4183318)

## End(Not run)
```

---

parse_lat *Parse latitude values*

Description

Parse latitude values

Usage

```
parse_lat(lat, format = NULL)
```

Arguments

`lat`	(numeric/integer/character) one or more latitude values
`format`	(character) format, default often works

Value

numeric vector

Errors

Throws warnings on parsing errors, and returns `NaN` in each case

Types of errors:

- invalid argument: e.g., letters passed instead of numbers, see https://en.cppreference.com/cpp/error/invalid_argument
- out of range: numbers of out acceptable range, see https://en.cppreference.com/cpp/error/out_of_range
- out of latitude range: not within -90/90 range
- minutes or seconds out of range (negative or 60 and above): not within 0/60 range, e.g. `"10 30,12345"` is read as 10 degrees, 30 minutes and 12345 seconds, so it gives a warning and `NA`
- a decimal degree or minute followed by another slot: only the last slot can be decimal, e.g. `"45.5 30"`

A comma or space is always a separator between degrees, minutes and seconds, never a decimal mark.

Examples

```
parse_lat("")
## Not run:
parse_lat("-91")
parse_lat("95")
parse_lat("asdfaf")

parse_lat("45")
parse_lat("-45")
parse_lat("-45.2323")

# out of range with std::stod?
parse_lat("-45.23232e24")
parse_lat("-45.23232e2")

# numeric input
parse_lat(1:10)
parse_lat(85:94)

# different formats
parse_lat("40.4183318 N")
parse_lat("40.4183318 S")
```

```
parse_lat("40 25 5.994") # => 40.41833

parse_lat("40.4183318N")
parse_lat("N40.4183318")
parse_lat("40.4183318S")
parse_lat("S40.4183318")

parse_lat("N 39 21.440") # => 39.35733
parse_lat("S 56 1.389") # => -56.02315

parse_lat("N40°25'5.994") # => 40.41833
parse_lat("40° 25´ 5.994\" N") # => 40.41833
parse_lat("40:25:6N")
parse_lat("40:25:5.994N")
parse_lat("40d 25' 6\" N")

## End(Not run)
```

`parse_llstr` *parse string with lat and lon together*

Description

parse string with lat and lon together

Usage

```
parse_llstr(str)
```

Arguments

`str`	(character) string with latitude and longitude, one or more in a vector.

Value

A data.frame with parsed latitude and longitude in decimal degrees.

Examples

```
parse_llstr("N 04.1683, E 101.5823")
parse_llstr("N04.82344, E101.61320")
parse_llstr("N 04.25164, E 101.70695")
parse_llstr("N05.03062, E101.75172")
parse_llstr("N05.03062,E101.75172")
parse_llstr("N4.9196, E101.345")
parse_llstr("N4.9196, E101.346")
parse_llstr("N4.9196, E101.347")
# no comma
parse_llstr("N4.9196 E101.347")
# no space
```

```
parse_llstr("N4.9196E101.347")

# DMS
parse_llstr("N4 51'36\", E101 34'7\"")
parse_llstr(c("4 51'36\"S, 101 34'7\"W", "N4 51'36\", E101 34'7\""))
```

`parse_lon` *Parse longitude values*

Description

Parse longitude values

Usage

```
parse_lon(lon, format = NULL)
```

Arguments

`lon`	(numeric/integer/character) one or more longitude values.
`format`	(character) format, not implemented yet.

Value

numeric vector

Errors

Throws warnings on parsing errors, and returns `NaN` in each case

Types of errors:

- invalid argument: e.g., letters passed instead of numbers, see https://en.cppreference.com/cpp/error/invalid_argument
- out of range: numbers of out acceptable range, see https://en.cppreference.com/cpp/error/out_of_range
- out of longitude range: not within -180/360 range
- minutes or seconds out of range (negative or 60 and above): not within 0/60 range, e.g. `"10 30,12345"` is read as 10 degrees, 30 minutes and 12345 seconds, so it gives a warning and `NA`
- a decimal degree or minute followed by another slot: only the last slot can be decimal, e.g. `"45.5 30"`

A comma or space is always a separator between degrees, minutes and seconds, never a decimal mark.

Examples

```
parse_lon("")
## Not run:
parse_lon("-181")
parse_lon("-361")
parse_lon("95")
parse_lon("asdfaf")

parse_lon("45")
parse_lon("-45")
parse_lon("-45.2323")
parse_lon("334")

# out of range with std::stod?
parse_lon("-45.23232e24")
parse_lon("-45.23232e2")
parse_lon("-45.23232")

# numeric input
parse_lon(1:10)
parse_lon(85:94)

# different formats
parse_lon("40.4183318 E")
parse_lon("40.4183318 W")
parse_lon("40 25 5.994") # => 40.41833

parse_lon("40.4183318W")
parse_lon("W40.4183318")
parse_lon("E40.4183318")
parse_lon("40.4183318E")

parse_lon("E 39 21.440") # => 39.35733
parse_lon("W 56 1.389") # => -56.02315

parse_lon("E40°25'5.994") # => 40.41833
parse_lon("40° 25´ 5.994\" E") # => 40.41833
parse_lon("40:25:6E")
parse_lon("40:25:5.994E")
parse_lon("40d 25' 6\" E")

## End(Not run)
```

---

parse_lon_lat *parse longitude and latitude*

Description

parse longitude and latitude

Usage

```
parse_lon_lat(lon, lat)
```

Arguments

lon	(character/numeric/integer) one or more longitude values
lat	(character/numeric/integer) one or more latitude values

Details

length(lon) == length(lat)

Value

data.frame, with columns lon, lat. on an invalid values, an NA is returned. In addition, warnings are thrown on invalid values

Examples

```
parse_lon_lat("-120.43", "49.12")
## Not run:
parse_lon_lat("-120.43", "93")
parse_lon_lat("-190", "49.12")
parse_lon_lat("240", "49.12")
parse_lon_lat("-190", "92")
# many
lons <- c("45W54.2356", "181", 45, 45.234234, "-45.98739874")
lats <- c("40.123°", "40.123N74.123W", "191.89", 12, "N45 04.25764")
parse_lon_lat(lons, lats)

## End(Not run)
```

---

parse_parts *parse coordinates into degrees, minutes and seconds*

Description

parse coordinates into degrees, minutes and seconds

Usage

```
parse_parts_lon(str)

parse_parts_lat(str)
```

Arguments

str	(character) string including longitude or latitude

Value

data.frame with columns for:

- deg (integer)
- min (integer)
- sec (numeric)

For negative values the sign is carried by the first non-zero part: `-45.5` gives deg -45, min 30, but `-0.5` gives deg 0, min -30. Minutes are always below 60 and seconds below 60, in absolute value.

NA/NaN given upon error

Examples

```
parse_parts_lon("140.4183318")
## Not run:
parse_parts_lon("174.6411133")
parse_parts_lon("-45.98739874")
parse_parts_lon("40.123W")

parse_parts_lat("45N54.2356")
parse_parts_lat("40.4183318")
parse_parts_lat("-74.6411133")
parse_parts_lat("-45.98739874")
parse_parts_lat("40.123N")
parse_parts_lat("N40°25'5.994")

# not working, needs format input
parse_parts_lat("N455698735")

# multiple
x <- c("40.123°", "40.123N74.123W", "191.89", 12, "N45 04.25764")
parse_parts_lat(x)
system.time(parse_parts_lat(rep(x, 10^2)))

## End(Not run)
```

Index

*.pz *(dms)*, 4
+.pz *(dms)*, 4
-.pz *(dms)*, 4
/.pz *(dms)*, 4

dms, 4

parse_hemisphere, 5
parse_lat, 6
parse_llstr, 8
parse_lon, 9
parse_lon_lat, 10
parse_parts, 11
parse_parts_lat *(parse_parts)*, 11
parse_parts_lon *(parse_parts)*, 11
parzer *(parzer-package)*, 2
parzer-package, 2
print.pz *(dms)*, 4
pz_d *(dms)*, 4
pz_degree *(dms)*, 4
pz_m *(dms)*, 4
pz_minute *(dms)*, 4
pz_s *(dms)*, 4
pz_second *(dms)*, 4