Package ‘lutz’

October 17, 2023

Type Package
Title Look Up Time Zones of Point Coordinates
Version 0.3.2
Description Input latitude and longitude values or an 'sf/sfc' POINT object and get back the time zone in which they exist. Two methods are implemented. One is very fast and uses 'Rcpp' in conjunction with data from the 'Javascript' library (<https://github.com/darkskyapp/tz-lookup-oss/>). This method also works outside of countries’ borders and in international waters, however speed comes at the cost of accuracy - near time zone borders away from populated centres there is a chance that it will return the incorrect time zone. The other method is slower but more accurate - it uses the 'sf' package to intersect points with a detailed map of time zones from here: <https://github.com/evansiroky/timezone-boundary-builder/>. The package also contains several utility functions for helping to understand and visualize time zones, such as listing of world time zones, including information about daylight savings times and their offsets from UTC. You can also plot a time zone to visualize the UTC offset over a year and when daylight savings times are in effect.

License MIT + file LICENSE


BugReports https://github.com/ateucher/lutz/issues

Depends R (>= 3.2)
Imports stats, Rcpp, lubridate
Suggests testthat (>= 2.1.0), sf (>= 0.7), sp, datasets, covr, ggplot2
Encoding UTF-8

RoxygenNote 7.2.3

LinkingTo Rcpp

NeedsCompilation yes

Author Andy Teucher [aut, cre] (<https://orcid.org/0000-0002-7840-692X>), Bob Rudis [ctb] (<https://orcid.org/0000-0001-5670-2640>)

Maintainer Andy Teucher <andy.teucher@gmail.com>
Repository  CRAN
Date/Publication  2023-10-17 20:30:02 UTC

R topics documented:

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tz_list</td>
<td>Create a list of Time Zones</td>
</tr>
<tr>
<td>tz_lookup</td>
<td>Lookup time zones of sf or sp points</td>
</tr>
</tbody>
</table>

Description

- **tz_list**: Output a list of time zone names, with daylight savings time and utc offset

Usage

tz_list()

Value

A data.frame of all time zones on your system. Columns:
- tz_name: the name of the time zone
- zone: time zone
- is_dst: is the time zone in daylight savings time
- utc_offset_h: offset from UTC (in hours)

Description

- **tz_lookup**: Lookup time zones of sf or sp points

Usage

tz_lookup(x, crs = NULL, method = "fast", warn = TRUE)
Arguments

x either an sfc or sf points or SpatialPoints(DataFrame) object

crs the coordinate reference system: integer with the EPSG code, or character with proj4string. If not specified (i.e., NULL) and x has no existing crs, EPSG: 4326 is assumed (lat/long).

method method by which to do the lookup. Either "fast" (default) or "accurate".

warn By default, if method = "fast" a warning is issued about the potential for inaccurate results. Set warn to FALSE to turn this off.

Details

Note that there are some regions in the world where a single point can land in two different overlapping time zones. The "accurate" method includes these, and when they are encountered they are concatenated in a single string, separated by a semicolon. The data used in the "fast" method does not include overlapping time zones at this time.

Value

character vector the same length as x specifying the time zone of the points.

Examples

```r
if (require("sf")) {

state_pts <- lapply(seq_along(state.center$x), function(i) {
  st_point(c(state.center$x[i], state.center$y[i]))
})

state_centers_sf <- st_sf(st_sfc(state_pts))

state_centers_sf$tz <- tz_lookup(state_centers_sf)

plot(state_centers_sf[, "tz")
}
```

tz_lookup_coords       Lookup time zones of lat/long pairs

Description

There are two methods - "fast", and "accurate". The "fast" version can look up many thousands of points very quickly, however when a point is near a time zone boundary and not near a populated centre, it may return the incorrect time zone. If accuracy is more important than speed, use method = "accurate".
Usage

tz_lookup_coords(lat, lon, method = "fast", warn = TRUE)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>lat</td>
<td>numeric vector of latitudes</td>
</tr>
<tr>
<td>lon</td>
<td>numeric vector of longitudes the same length as x</td>
</tr>
<tr>
<td>method</td>
<td>method by which to do the lookup. Either &quot;fast&quot; (default) or &quot;accurate&quot;.</td>
</tr>
<tr>
<td>warn</td>
<td>By default, if method = &quot;fast&quot; a warning is issued about the potential for inaccurate results. Set warn to FALSE to turn this off.</td>
</tr>
</tbody>
</table>

Value

character vector the same length as x and y specifying the time zone of the points.

Examples

tz_lookup_coords(42, -123)
tz_lookup_coords(lat = c(48.9, 38.5, 63.1, -25), lon = c(-123.5, -110.2, -95.0, 130))

tz_offset dt, tz = ""

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dt</td>
<td>Date, POSIXx or date-like character string</td>
</tr>
<tr>
<td>tz</td>
<td>A time zone name from base::OlsonNames(). Not required if dt is a POSIXt object with a time zone component.</td>
</tr>
</tbody>
</table>

Value

a one-row data frame with details of the time zone

Examples

tz_offset("2018-06-12", "America/Moncton")
tz_plot

**Description**

Make a circular plot of a time zone, visualizing the UTC offset over the course of the year, including Daylight Savings times.

**Usage**

```r
tz_plot(tz)
```

**Arguments**

- `tz` a valid time zone name. See `OlsonNames()`

**Value**

A `ggplot2` object

**Examples**

```r
tz_plot("America/Vancouver")
```
Index

base::OlsonNames(), 4
OlsonNames(), 5
tz_list, 2
tz_lookup, 2
tz_lookup_coords, 3
tz_offset, 4
tz_plot, 5