Package 'RcppCCTZ'

January 20, 2025

Type Package

Title 'Rcpp' Bindings for the 'CCTZ' Library

Version 0.2.13

Date 2024-12-11

Description 'Rcpp' Access to the 'CCTZ' timezone library is provided. 'CCTZ' is
a C++ library for translating between absolute and civil times using the rules
of a time zone. The 'CCTZ' source code, released under the Apache 2.0 License,
is included in this package. See <https://github.com/google/cctz> for more
details.

License GPL (>= 2)

Imports Rcpp (>= 0.11.0)

Suggests tinytest

LinkingTo Rcpp

SystemRequirements A 64-bit POSIX OS such as Linux or OS X with IANA
time zone data in /usr/share/zoneinfo as well as a
recent-enough C++11 compiler (such as g++-4.9 or later which is
preferred, g++-4.8 works too). On Windows the zoneinfo included
with R is used; and time parsing support is enabled via a
backport of std::get_time from the LLVM libc++ library.

URL <https://github.com/eddelbuettel/rcppcctz>,
<https://dirk.eddelbuettel.com/code/rcpp.cctz.html>

BugReports <https://github.com/eddelbuettel/rcppcctz/issues>

RoxygenNote 6.0.1

NeedsCompilation yes

Author Dirk Eddelbuettel [aut, cre] (<https://orcid.org/0000-0001-6419-907X>),
Bradley White [aut] (Principal author of CCTZ)

Maintainer Dirk Eddelbuettel <edd@debian.org>

Repository CRAN

Date/Publication 2024-12-12 04:40:08 UTC

Contents

RcppCCTZ-package   *A Simple Wrapper to the CCTZ Library for Time Zone Calculations*

Description

CCTZ contains two underlying libraries which build on the C++11 library chrono. The first covers *civil time* for computing with human-scale time such as dates and time. It is header-only. The second covers time zones and allow translation between absolute time and civil time.

RcppCCTZ brings CCTZ to R by means of Rcpp.

Details

CCTZ requires a valid timezone library as well as recent-enough compiler to cope with C++11.

Windows is supported since version 0.2.0 via the g++-4.9 compiler, but note that it provides an *incomplete* C++11 library. The std::get_time function was ported from libc++ of the LLVM to enable this. However, string formatting is more limited as the libc++ library used by g++-4.9 does not provide complete C++11 semantics. As one example, CCTZ frequently uses "%F %T" which do not work on Windows; one has to use "%Y-%m-%d %H:%M:%S".

Author(s)

Dirk Eddelbuettel wrote the package; Dan Dillon ported std::get_time from LLVM's libc++; Bradley White and Greg Miller wrote the underlying CCTZ library.

Maintainer: Dirk Eddelbuettel <edd@debian.org>

References

The CCZT repository at https://github.com/google/cctz has additional information.

Examples

```
helloMoon()
```

---

formatDatetime	*Format a Datetime vector as a string vector*

Description

Format a Datetime vector

Usage

```
formatDatetime(dtv, fmt = "%Y-%m-%dT%H:%M:%E*S%Ez", lcltzstr = "UTC",
  tgttzstr = "UTC")

formatDouble(secv, nanov, fmt = "%Y-%m-%dT%H:%M:%E*S%Ez",
  tgttzstr = "UTC")
```

Arguments

dtv	A Datetime vector object to be formatted
fmt	A string with the format, which is based on `strftime` with some extensions; see the CCTZ documentation for details.
lcltzstr	The local timezone object for creation the CCTZ timepoint
tgttzstr	The target timezone for the desired format
secv	A numeric vector with seconds since the epoch
nanov	A numeric vector with nanoseconds since the epoch, complementing `secv`.

Details

An alternative to `format.POSIXct` based on the CCTZ library. The `formatDouble` variant uses two vectors for seconds since the epoch and fractional nanoseconds, respectively, to provide fuller resolution.

Value

A string vector with the requested format of the datetime objects

Note

Windows is now supported via the `g++-4.9` compiler, but note that it provides an *incomplete* C++11 library. This means we had to port a time parsing routine, and that string formatting is more limited. As one example, CCTZ frequently uses `"%F %T"` which do not work on Windows; one has to use `"%Y-%m-%d %H:%M:%S"`.

Author(s)

Dirk Eddelbuettel

Examples

```
## Not run:
now <- Sys.time()
formatDatetime(now)              # current (UTC) time, in full precision RFC3339
formatDatetime(now, tgttzstr="America/New_York")  # same but in NY
formatDatetime(now + 0:4)    # vectorised

## End(Not run)
```

---

  parseDatetime   *Parse a Datetime vector from a string vector*

Description

Parse a Datetime vector

Usage

```
parseDatetime(svec, fmt = "%Y-%m-%dT%H:%M:%E*S%Ez", tzstr = "UTC")

parseDouble(svec, fmt = "%Y-%m-%dT%H:%M:%E*S%Ez", tzstr = "UTC")
```

Arguments

svec	A string vector from which a Datetime vector is to be parsed
fmt	A string with the format, which is based on `strftime` with some extensions; see the CCTZ documentation for details.
tzstr	The local timezone for the desired format

Details

An alternative to `as.POSIXct` based on the CCTZ library

Value

A Datetime vector object for `parseDatetime`, a numeric matrix with two columns for seconds and nanoseconds for `parseDouble`

Author(s)

Dirk Eddelbuettel

Examples

```
ds <- getOption("digits.secs")
options(digits.secs=6) # max value
parseDatetime("2016-12-07 10:11:12",        "%Y-%m-%d %H:%M:%S")   # full seconds
parseDatetime("2016-12-07 10:11:12.123456", "%Y-%m-%d %H:%M:%E*S") # fractional seconds
parseDatetime("2016-12-07T10:11:12.123456-00:00")  ## default RFC3339 format
parseDatetime("20161207 101112.123456",     "%E4Y%m%d %H%M%E*S")   # fractional seconds
now <- trunc(Sys.time())
parseDatetime(formatDatetime(now + 0:4))     # vectorised
options(digits.secs=ds)
```

---

toTz   *Shift datetime object from one timezone to another*

Description

Change from one given timezone to another.

Usage

```
toTz(dtv, tzfrom, tzto, verbose = FALSE)
```

Arguments

dtv	A DatetimeVector object specifying when the difference is to be computed.
tzfrom	The first time zone as a character vector.
tzto	The second time zone as a character vector.
verbose	A boolean toggle indicating whether more verbose operations are desired, default is FALSE.

Details

Time zone offsets vary by date, and this helper function converts a Datetime object from one given timezone to another.

Value

A DatetimeVector object with the given (civil time) determined by the incoming object (and its timezone) shifted to the target timezone.

Author(s)

Dirk Eddelbuettel

Examples

```
## Not run:
toTz(Sys.time(), "America/New_York", "Europe/London")
# this redoes the 'Armstrong on the moon in NYC and Sydney' example
toTz(ISOdatetime(1969,7,20,22,56,0,tz="UTC"), "America/New_York", "Australia/Sydney", verbose=TRUE)
# we can also explicitly format for Sydney time
format(toTz(ISOdatetime(1969,7,20,22,56,0,tz="UTC"),
            "America/New_York", "Australia/Sydney", verbose=TRUE),
       tz="Australia/Sydney")

## End(Not run)
```

---

tzDiff   *Return difference between two time zones at a given date.*

Description

Difference between two given timezones at a specified date.

Usage

```
tzDiff(tzfrom, tzto, dt, verbose = FALSE)
```

Arguments

tzfrom	The first time zone as a character vector.
tzto	The second time zone as a character vector.
dt	A Datetime object specifying when the difference is to be computed.
verbose	A boolean toggle indicating whether more verbose operations are desired, default is FALSE.

Details

Time zone offsets vary by date, and this helper function computes the difference (in hours) between two time zones for a given date time.

Value

A numeric value with the difference (in hours) between the first and second time zone at the given date

Author(s)

Dirk Eddelbuettel

Examples

```
## Not run:
# simple call: difference now
tzDiff("America/New_York", "Europe/London", Sys.time())
# tabulate difference for every week of the year
table(sapply(0:52, function(d) tzDiff("America/New_York", "Europe/London",
                                    as.POSIXct(as.Date("2016-01-01") + d*7))))

## End(Not run)
```

Index