Package ‘udunits2’

October 12, 2022

Type Package
Title Udunits-2 Bindings for R
Version 0.13.2.1
Date 2022-03-11
Description Provides simple bindings to Unidata's udunits library.
URL https://github.com/pacificclimate/Rudunits2
https://www.unidata.ucar.edu/software/udunits/
SystemRequirements udunits (>= 2) from
https://www.unidata.ucar.edu/software/udunits/
License GPL-2
LazyLoad yes
Depends R (>= 2.10.0)
NeedsCompilation yes
X-CRAN-Comment Orphaned on 2022-01-06 as installation problems were not corrected.
Maintainer ORPHANED
Repository CRAN
Date/Publication 2022-03-11 20:00:50 UTC
Author James Hiebert [aut, cre],
CRAN team [ctb, cph] (revisions/corrections in 2022)
X-CRAN-Original-Maintainer <ripley@stats.ox.ac.uk>

R topics documented:

udunits2-package ......................................................... 2
ud.are.convertible ..................................................... 3
ud.convert ............................................................... 4
ud.get.name ............................................................. 5
ud.have.unit.system ................................................... 6
ud.is.parseable ........................................................... 7
ud.set.encoding ........................................................ 8
Index 10

udunits2-package  

udunits-2 bindings for R

Description

This package provides simple bindings to version 2 of Unidata’s udunits library.

Details

<table>
<thead>
<tr>
<th>Package</th>
<th>udunits2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Package</td>
</tr>
<tr>
<td>Version</td>
<td>0.6</td>
</tr>
<tr>
<td>Date</td>
<td>2011-02-11</td>
</tr>
<tr>
<td>License</td>
<td>GPL-2</td>
</tr>
<tr>
<td>LazyLoad</td>
<td>yes</td>
</tr>
</tbody>
</table>

This package provides simple bindings to the version 2 API of Unidata’s udunits library. While the entire API is not supported, we have chosen to boil it down to a few simple functions to be able to exploit the most useful functionality that the library provides. This package provides the following functions:

- `ud.is.parseable`
- `ud.get.name`
- `ud.get.symbol`
- `ud.are.convertible`
- `ud.convert`

Please see the respective function help pages for further details and usage.

Author(s)

James Hiebert <hiebert@uvic.ca>

Maintainer: James Hiebert <hiebert@uvic.ca>

References

Unidata’s udunits web page: https://www.unidata.ucar.edu/software/udunits/

See Also

`ud.is.parseable` `ud.get.name` `ud.get.symbol` `ud.are.convertible` `ud.convert`
Determine whether two units may be converted between each other

Description

This function takes udunits compatible strings and determines whether or not it is possible to convert between them.

Usage

ud.are.convertible(u1, u2)

Arguments

u1 A character string which is parseable into a udunits compatible unit.
 u2 Another character string which is also parseable into a udunits compatible unit.

Details

Even if two units are parseable and recognized by the udunits library, it may or may not be possible to convert from one to another. For example, it makes sense to convert from celsius to kelvin, however not from celsius to kilograms. This function allows the user to check if two units are of the same system and if there exists a defined conversion between the two.

Value

Returns a logical: True if the units can be converted between each other, False if either of the arguments is not parseable by udunits, or if no conversion is possible.

Author(s)

James Hiebert <hiebert@uvic.ca>

References

See the udunits function ut_are_convertible: https://www.unidata.ucar.edu/software/udunits/udunits-2.1.24/udunits2lib.html#ut_005fare_005fconvertible_0028_0029 and the main ununits webpage: https://www.unidata.ucar.edu/software/udunits/

See Also

ud.is.parseable

Examples

ud.are.convertible("miles", "km") # TRUE
ud.are.convertible("grams", "kilograms") # TRUE
ud.are.convertible("celsius", "grams") # FALSE
ud.are.convertible("not", "parseable") # FALSE
**ud.convert**

Convert numeric types from one unit to another

**Description**

This function takes the numeric argument \( x \), quantified in units \( u1 \) and converts it to be of units \( u2 \).

**Usage**

\[
\text{ud.convert}(x, u1, u2)
\]

**Arguments**

- \( x \) Some argument which is convertible to a numeric type by `as.double`.
- \( u1 \) A character string which is parseable into a udunits compatible unit.
- \( u2 \) Another character string which is also parseable into a udunits compatible unit and for which there exists a defined transformation from the units represented by \( u1 \).

**Details**

This function uses the udunits function `cv_convert_doubles` to convert the argument from one set of units to another.

**Value**

Returns a numeric type having converted from one unit to another. The attributes of the original argument \( x \) (e.g. class, dimensions, etc.) are preserved and then re-applied to the return value of the transformation as such: \( \text{attributes(rv)} \leftarrow \text{attributes(x)} \) If either of unit \( u1 \) or \( u2 \) is unparsable, or there does not exist a conversion from one to the other the function raises an error.

**Author(s)**

James Hiebert <hiebert@uvic.ca>

**References**


**See Also**

`ud.are.convertible`
Examples

```r
x <- seq(10)
ud.convert(x, "miles", "km")  # c(1.609344, 3.218688, 4.828032, ...)
x <- c(-40, 0, 100)
ud.convert(x, "celsius", "degree_fahrenheit")  # c(-40, 32, 212)
err <- try(ud.convert(100,"miles", "grams"))  # Error
err <- try(ud.convert(NA, "not", "parseable"))  # Error
```

```
ud.get.name  Retrieve the udunits name or symbol from the database for a given
units string

Description
Retrieve the udunits name or symbol from the database for a given units string.

Usage
ud.get.name(unit.string)

Arguments
unit.string  A character string which is parseable into a udunits compatible unit.

Details
This function retrieves the udunits name or symbol from the udunits database and returns it. It uses the udunits functions ut_get_name and ut_get_symbol respectively.

Value
Returns a character string stating the udunits’s name/symbol for the given unit, or an empty character string if the unit does not map to a name/symbol for the default character set. If the unit is unparseable, the function raises an error.

Note
More often than not units do not have names or symbols that are returned by the base functions. This depends entirely on what is defined in the units data base, which is—as of API version 2—an XML database which ships with the library. See Unidata’s website for more information about the XML database: https://www.unidata.ucar.edu/software/udunits/udunits-2-units.html. All in all, don’t put too much stock in them, for they are for convenience only. If your application requires certain names and symbols to be present, the XML database is local and editable.

Author(s)
James Hiebert <hiebert@uvic.ca>
ud.have.unit.system

Determine whether udunits has loaded its units database

Description

This function checks whether udunits has successfully found and loaded its run-time XML units database.

Usage

ud.have.unit.system()

Details

At package load time, Rudunits attempts to load a unit system from an XML units database from the file system. This might be installed with the system library (e.g. through apt or yum), or the user can use their own. The file-system location is configured using the UDUNITS2_XML_PATH environment variable.

This package will attempt to load the path contained in UDUNIT2_XML_PATH. If it's empty, it will attempt to load it from the system library. Failing that it will attempt to load its own XML database that ships with the package (from udunits source).

One can call ud.have.unit.system to confirm that the units database has been loaded successfully.

Value

Returns a logical: True if udunits has successfully found and loaded the XML units database, False otherwise.

References


Examples

units.to.display <- c("celsius", # has no name, messed up symbol (maybe a bug in R?"
"kg",
"hr", # has no symbol
"K",
"degrees",
"m",
"ohm")

for (u in units.to.display) {
  print(ud.get.name(u))
  print(ud.get.symbol(u))
}

---

ud.have.unit.system

Determine whether udunits has loaded its units database

Description

This function checks whether udunits has successfully found and loaded its run-time XML units database.

Usage

ud.have.unit.system()

Details

At package load time, Rudunits attempts to load a unit system from an XML units database from the file system. This might be installed with the system library (e.g. through apt or yum), or the user can use their own. The file-system location is configured using the UDUNITS2_XML_PATH environment variable.

This package will attempt to load the path contained in UDUNIT2_XML_PATH. If it's empty, it will attempt to load it from the system library. Failing that it will attempt to load its own XML database that ships with the package (from udunits source).

One can call ud.have.unit.system to confirm that the units database has been loaded successfully.

Value

Returns a logical: True if udunits has successfully found and loaded the XML units database, False otherwise.

References


Examples

units.to.display <- c("celsius", # has no name, messed up symbol (maybe a bug in R?"
"kg",
"hr", # has no symbol
"K",
"degrees",
"m",
"ohm")

for (u in units.to.display) {
  print(ud.get.name(u))
  print(ud.get.symbol(u))
}

---

ud.have.unit.system

Determine whether udunits has loaded its units database

Description

This function checks whether udunits has successfully found and loaded its run-time XML units database.

Usage

ud.have.unit.system()

Details

At package load time, Rudunits attempts to load a unit system from an XML units database from the file system. This might be installed with the system library (e.g. through apt or yum), or the user can use their own. The file-system location is configured using the UDUNITS2_XML_PATH environment variable.

This package will attempt to load the path contained in UDUNIT2_XML_PATH. If it's empty, it will attempt to load it from the system library. Failing that it will attempt to load its own XML database that ships with the package (from udunits source).

One can call ud.have.unit.system to confirm that the units database has been loaded successfully.

Value

Returns a logical: True if udunits has successfully found and loaded the XML units database, False otherwise.
ud.is.parseable

**Author(s)**
James Hiebert <hiebert@uvic.ca>

**Examples**

```
ud.have.unit.system() # TRUE
```

---

**Description**

Determine whether a unit string is parseable and recognized by the udunits library.

**Usage**

```
ud.is.parseable(unit.string)
```

**Arguments**

- `unit.string`: A character string representing a type of units which may be parseable by the udunits library

**Details**

`ud.is.parseable` uses udunit's function `ut_parse` to determine whether or not the given unit string is parseable. If `ut_parse` returns NULL, then `ud.is.parseable` will return `FALSE`.

**Value**

Returns a logical: `True` if the units is parseable and recognized by the udunits library, `False` otherwise.

**Note**

There is a note in the `ut_parse` docs about how the argument string must have no leading or trailing whitespace. We make sure in this package to always call `ut_trim` on any strings before they are passed to `ut_parse`. The package user need not strip whitespace before-hand.

**Author(s)**
James Hiebert <hiebert@uvic.ca>

**References**

See Also

ud.are.convertible

Examples

```r
ud.is.parseable("K") # TRUE
ud.is.parseable(" K ") # TRUE
ud.is.parseable("miles") # TRUE
ud.is.parseable("Not parseable") # FALSE
```

---

**ud.set.encoding**

*Set the udunits package level encoding type*

Description

This function sets the encoding type parameter which is global to the R udunits2 package.

Usage

```r
ud.set.encoding(enc.string)
```

Arguments

- `enc.string`: A character string representing the encoding type. Valid strings are `utf8`, `ascii`, `iso-8859-1`, and `latin1` (an alias for ISO-8859-1).

Details

Encoding type is a parameter to nearly all of the functions in the udunits library. By default, the R udunits2 package sets the encoding type to UTF-8, however this package allows the user to set other encoding types which are supported by the udunits library. It presently supports UTF-8, ASCII, and ISO-8859-1.

Value

Returns no value. Raises an error if it is not given a valid encoding string.

Author(s)

James Hiebert <hiebert@uvic.ca>

References

Unidata’s udunits reference: [https://www.unidata.ucar.edu/software/udunits/](https://www.unidata.ucar.edu/software/udunits/) API guide chapter on data types: [https://www.unidata.ucar.edu/software/udunits/udunits-2.1.24/udunits2lib.html#Types](https://www.unidata.ucar.edu/software/udunits/udunits-2.1.24/udunits2lib.html#Types)
Examples

valid.enc.strings <- c('utf8', 'ascii', 'iso-8859-1', 'latin1')
lapply(valid.enc.strings, ud.set.encoding)
err <- try(ud.set.encoding("This will fail"))
# Index

- **climate**  
  udunits2-package, 2
- **meteorology**  
  udunits2-package, 2
- **unitdata**  
  udunits2-package, 2
- **units**  
  udunits2-package, 2

ud.are.convertible, 2, 3, 4, 8  
ud.convert, 2, 4  
ud.get.name, 2, 5  
ud.get.symbol, 2  
ud.get.symbol(ud.get.name), 5  
ud.have.unit.system, 6  
ud.is.parseable, 2, 3, 7  
ud.set.encoding, 8  
uudunits2 (udunits2-package), 2  
uudunits2-package, 2