

# Package ‘rebus.datetimes’

August 29, 2016

**Title** Date and Time Extensions for the 'rebus' Package

**Version** 0.0-1

**Date** 2015-12-16

**Author** Richard Cotton [aut, cre]

**Maintainer** Richard Cotton <richierocks@gmail.com>

**Description** Build regular expressions piece by piece using human readable code.  
This package contains date and time functionality, and is primarily intended to be used by package developers.

**Depends** R (>= 3.1.0)

**Imports** rebus.base

**Suggests** stats, testthat

**License** Unlimited

**LazyData** true

**RoxygenNote** 5.0.1

**Collate** 'imports.R' 'datetime-constants.R' 'datetime.R'  
'iso-classes.R'

**NeedsCompilation** no

**Repository** CRAN

**Date/Publication** 2015-12-16 09:36:53

## R topics documented:

get_weekdays . . . . .	2
IsoClasses . . . . .	3
OPT_LEADING_0 . . . . .	4

<b>Index</b>	<b>10</b>
--------------	-----------

---

get_weekdays	<i>Get the days of the week</i>
--------------	---------------------------------

---

### Description

Get the names of the days of the week in a given locale.

### Usage

```
get_weekdays(abbreviate = FALSE, locale = NULL, from = Sys.Date())
```

```
get_months(abbreviate = FALSE, locale = NULL, from = Sys.Date())
```

### Arguments

abbreviate	A logical value indicating whether or not to return abbreviated names (if they are available).
locale	A string denoting the name of the locale to retrieve names in, or NULL to use the current locale.
from	Date to use for the first day/month.

### Value

A string containing a regular expression of the names of the days of the week, separated by pipes. The first day of the week will be the current day.

### Note

See [Sys.setlocale](#) and <http://stackoverflow.com/q/20960821/134830> and <http://stackoverflow.com/q/26603564/134830> for how to specify the locale.

### Examples

```
get_weekdays()
get_weekdays(TRUE)
get_months()
get_months(TRUE)
## Not run:
if(.Platform$OS.type == "windows")
{
  get_weekdays(locale = "French_France")
  get_weekdays(TRUE, locale = "French_France")
  get_weekdays(locale = "Arabic_Qatar")
  get_weekdays(TRUE, locale = "Arabic_Qatar")
} else
{
  get_weekdays(locale = "fr_FR.utf8")
  get_weekdays(TRUE, locale = "fr_FR.utf8")
}
```

```
    get_weekdays(locale = "ar_QA.utf8")
    get_weekdays(TRUE, locale = "ar_QA.utf8")
  }

  ## End(Not run)
```

---

IsoClasses

*ISO 8601 date-time classes*

---

## Description

Match ISO 8601 date and time classes.

## Usage

```
iso_date(lo, hi, char_class = TRUE)

iso_time(lo, hi, char_class = TRUE)

iso_datetime(lo, hi, char_class = TRUE)
```

## Arguments

lo	A non-negative integer. Minimum number of repeats, when grouped.
hi	positive integer. Maximum number of repeats, when grouped.
char_class	TRUE or FALSE. Should the values be wrapped into a character class?

## Value

A character vector representing part or all of a regular expression.

## References

<http://www.iso.org/iso/iso8601>

## Examples

```
iso_date()
iso_time()
iso_datetime()

# With repetition
iso_date(3, 6)
iso_time(1, Inf)
iso_datetime(0, Inf)

# Without a class wrapper
iso_date(char_class = FALSE)
```

---

OPT_LEADING_0	<i>Date-time regexes</i>
---------------	--------------------------

---

**Description**

Compound regex constants for matching ISO 8601 dates and times.

**Usage**

OPT\_LEADING\_0

DTSEP

CENTURY

CENTURY\_IN

YEAR

YEAR2

YEAR4

MONTH

MONTH\_IN

WEEK\_OF\_YEAR

WEEK\_OF\_YEAR\_IN

DAY

DAY\_IN

DAY\_SINGLE

DAY\_OF\_YEAR

DAY\_OF\_YEAR\_IN

WEEKDAY1

WEEKDAY0

HOUR24

HOUR24\_SINGLE

HOUR24\_IN

HOUR12

HOUR12\_SINGLE

HOUR12\_IN

MINUTE

MINUTE\_IN

SECOND

SECOND\_IN

FRACTIONAL\_SECOND

FRACTIONAL\_SECOND\_IN

AM\_PM

TIMEZONE\_OFFSET

TIMEZONE

ISO\_DATE

ISO\_DATE\_IN

ISO\_TIME

ISO\_TIME\_IN

ISO\_DATETIME

ISO\_DATETIME\_IN

YMD

YMD\_IN

YDM

YDM\_IN

MYD

MYD\_IN

MDY

MDY\_IN

DYM

DYM\_IN

DMY

DMY\_IN

HMS

HMS\_IN

HM

HM\_IN

MS

MS\_IN

```
datetime(x, locale = NULL, io = c("output", "input"))
```

### Arguments

x	A <code>strptime</code> -style date-time format string.
locale	A string specifying a locale.
io	Are you trying to match output or input? The latter is less strict about leading zeroes and spaces.

### Format

An object of class character of length 1.

### Value

A character vector representing part or all of a regular expression.

### Note

"%O[dHImMUVwWy]", "%E[cCyYxX]", "%x", "%X" and "%+" are supposed to be locale-dependent upon output, but implementing this in an OS-portable way seems to be much more effort than it's

worth.

### See Also

[strftime](#) that describes formatting codes, [ClassGroups](#), [Sys.setlocale](#)

### Examples

```

datetime("%m/%d/%Y")           # match US style dates
twelve_or_twentyfour <- rebus.base::or("%H", "%I%p")
datetime(twelve_or_twentyfour) # match hours in 24h or 12h format

## Not run:
# week days and months can be matched in any locale
if(.Platform$OS.type == "windows")
{
  fr_FR <- "French_France"
  ar_QA <- "Arabic_Qatar"
} else if(Sys.info()["sysname"] != "Darwin") # mac
{
  fr_FR <- "fr_FR"
  ar_QA <- "ar_QA"
} else if(Sys.info()["sysname"] != "Linux")
{
  fr_FR <- "fr_FR.utf8"
  ar_QA <- "ar_QA.utf8"
}
datetime("%a %A %b %B", fr_FR)
datetime("%a %A %b %B", ar_QA)

# All letter tokens. Lots of output.
x <- paste0("%", c(letters, LETTERS))
stats::setNames(datetime(x), x)

## End(Not run)

# Individual date-time components
DTSEP           # optional selected punctuation or space
CENTURY         # exactly two digits
YEAR           # one to four digits
YEAR2          # exactly two digits
YEAR4          # exactly four digits
MONTH          # number from 1 to 12, leading zero
WEEK_OF_YEAR   # number from 0 to 53, leading zero
DAY            # number from 1 to 31, leading zero
DAY_SINGLE     # leading space
HOUR24        # 24 hour clock, leading zero
HOUR12        # 12 hour clock, leading zero
HOUR24_SINGLE  # 24 hour clock, leading space
HOUR12_SINGLE  # 12 hour clock, leading space
MINUTE        # number from 0 to 59, leading zero
SECOND        # number from 0 to 61 (leap seconds), leading zero
FRACTIONAL_SECOND # a second optional decimal point and up to 6 digits

```

```

AM_PM          # AM or PM, any case
TIMEZONE_OFFSET # optional plus or minus, then four digits
TIMEZONE       # Any value returned by OlsonNames()
# ISO 8601 formats
ISO_DATE       # %Y-%m-%d
ISO_TIME       # %H:%M:%S
ISO_DATETIME   # ISO_DATE followed by ISO_TIME, separated by space or "T".
# Compound forms, separated by DTSEP
YMD
YDM
MYD
MDY
DYM
DMY
HMS
HM
MS

# Some forms have less strict alternatives for input (with an '_IN' suffix).
CENTURY_IN
MONTH_IN
WEEK_OF_YEAR_IN
DAY_IN
HOUR24_IN
HOUR12_IN
MINUTE_IN
SECOND_IN
FRACTIONAL_SECOND_IN
ISO_DATE_IN
ISO_TIME_IN
ISO_DATETIME_IN
YMD_IN
YDM_IN
MYD_IN
MDY_IN
DYM_IN
DMY_IN
HMS_IN
HM_IN
MS_IN

dates <- seq(as.Date("2000-01-01"), as.Date("2001-01-01"), "1 day")
datetimes <- seq(as.POSIXct(Sys.Date()), as.POSIXct(Sys.Date() + 1), "1 sec")
times <- substring(datetimes, 12, 19)
stopifnot(
  all(grepl(ISO_DATE, dates)),
  all(grepl(ISO_TIME, times)),
  all(grepl(ISO_DATETIME, datetimes))
)
non_dates <- c(
  "2000-13-01", "2000-01-32", "2000-00-01", "2000-01-00"
)
non_times <- c(

```



```
    "24:00:00", "23:60:59", "23:59:62", "23 59 59"  
  )  
  stopifnot(  
    all(!grepl(ISO_DATE, non_dates)),  
    all(!grepl(ISO_TIME, non_times))  
  )
```

# Index

## \*Topic **datasets**

- OPT\_LEADING\_0, 4
- AM\_PM (OPT\_LEADING\_0), 4
- CENTURY (OPT\_LEADING\_0), 4
- CENTURY\_IN (OPT\_LEADING\_0), 4
- ClassGroups, 7
- DateTime (OPT\_LEADING\_0), 4
- datetime (OPT\_LEADING\_0), 4
- DAY (OPT\_LEADING\_0), 4
- DAY\_IN (OPT\_LEADING\_0), 4
- DAY\_OF\_YEAR (OPT\_LEADING\_0), 4
- DAY\_OF\_YEAR\_IN (OPT\_LEADING\_0), 4
- DAY\_SINGLE (OPT\_LEADING\_0), 4
- DMY (OPT\_LEADING\_0), 4
- DMY\_IN (OPT\_LEADING\_0), 4
- DTSEP (OPT\_LEADING\_0), 4
- DYM (OPT\_LEADING\_0), 4
- DYM\_IN (OPT\_LEADING\_0), 4
- FRACTIONAL\_SECOND (OPT\_LEADING\_0), 4
- FRACTIONAL\_SECOND\_IN (OPT\_LEADING\_0), 4
- get\_months (get\_weekdays), 2
- get\_weekdays, 2
- HM (OPT\_LEADING\_0), 4
- HM\_IN (OPT\_LEADING\_0), 4
- HMS (OPT\_LEADING\_0), 4
- HMS\_IN (OPT\_LEADING\_0), 4
- HOUR12 (OPT\_LEADING\_0), 4
- HOUR12\_IN (OPT\_LEADING\_0), 4
- HOUR12\_SINGLE (OPT\_LEADING\_0), 4
- HOUR24 (OPT\_LEADING\_0), 4
- HOUR24\_IN (OPT\_LEADING\_0), 4
- HOUR24\_SINGLE (OPT\_LEADING\_0), 4
- ISO\_DATE (OPT\_LEADING\_0), 4
- iso\_date (IsoClasses), 3
- ISO\_DATE\_IN (OPT\_LEADING\_0), 4
- ISO\_DATETIME (OPT\_LEADING\_0), 4
- iso\_datetime (IsoClasses), 3
- ISO\_DATETIME\_IN (OPT\_LEADING\_0), 4
- ISO\_TIME (OPT\_LEADING\_0), 4
- iso\_time (IsoClasses), 3
- ISO\_TIME\_IN (OPT\_LEADING\_0), 4
- IsoClasses, 3
- IsoDateTime (IsoClasses), 3
- MDY (OPT\_LEADING\_0), 4
- MDY\_IN (OPT\_LEADING\_0), 4
- MINUTE (OPT\_LEADING\_0), 4
- MINUTE\_IN (OPT\_LEADING\_0), 4
- MONTH (OPT\_LEADING\_0), 4
- MONTH\_IN (OPT\_LEADING\_0), 4
- MS (OPT\_LEADING\_0), 4
- MS\_IN (OPT\_LEADING\_0), 4
- MYD (OPT\_LEADING\_0), 4
- MYD\_IN (OPT\_LEADING\_0), 4
- OPT\_LEADING\_0, 4
- SECOND (OPT\_LEADING\_0), 4
- SECOND\_IN (OPT\_LEADING\_0), 4
- strptime, 6, 7
- Sys.setlocale, 2, 7
- TIMEZONE (OPT\_LEADING\_0), 4
- TIMEZONE\_OFFSET (OPT\_LEADING\_0), 4
- WEEK\_OF\_YEAR (OPT\_LEADING\_0), 4
- WEEK\_OF\_YEAR\_IN (OPT\_LEADING\_0), 4
- WEEKDAY0 (OPT\_LEADING\_0), 4
- WEEKDAY1 (OPT\_LEADING\_0), 4
- YDM (OPT\_LEADING\_0), 4
- YDM\_IN (OPT\_LEADING\_0), 4
- YEAR (OPT\_LEADING\_0), 4
- YEAR2 (OPT\_LEADING\_0), 4
- YEAR4 (OPT\_LEADING\_0), 4

YMD (OPT\_LEADING\_0), 4  
YMD\_IN (OPT\_LEADING\_0), 4