# Package 'padr'

January 16, 2017

**Type** Package

**Title** Quickly Get Datetime Data Ready for Analysis

**Version** 0.1.0

**Author** Edwin Thoen

**Maintainer** Edwin Thoen <edwinthoen@gmail.com>

**Description** The padr package helps transforming datetime data into a format ready for analysis. It offers two functionalities; aggregating data to a higher level interval (thicken) and imputing records where observations were absent (pad). It also offers a few functions that assist with filling missing values after padding.

**License** MIT + file LICENSE

**Encoding** UTF-8

**LazyData** true

**Depends** R (>= 3.0.0)

**Imports** Rcpp

**Suggests** lubridate, ggplot2, testthat, knitr, rmarkdown, lazyeval, dplyr, tidyr, data.table

**RoxygenNote** 5.0.1

**LinkingTo** Rcpp

**VignetteBuilder** knitr

**URL** https://github.com/EdwinTh/padr

**BugReports** https://github.com/EdwinTh/padr/issues

## R topics documented:

## emergency *Emergency Calls for Montgomery County, PA*

### Description

The emergency calls coming in at Montgomery County, PA since 2015-12-10. Data set was created at 2016-10-17 16:15:40 CEST from the API and contains events untill 2016-10-17 09:47:03 EST. From the original set the columns desc and e are not included.

### Usage

```
emergency
```

### Format

A data frame with 120450 rows and 6 variables:

**lat** Latitude from Google maps, based on the address

**lng** Longitude from Google maps, based on the address

**zip** Zipcode from Google, when possible

**title** Title, emergency category

**time_stamp** YYYY-MM-DD HH:MM:SS

**twp** Township

### Source

https://storage.googleapis.com/montco-stats/tzr.csv

## fill_by_function *Fill missing values by a function of the nonmissings.*

### Description

For each specified column in x replace the missing values by a function of the nonmissing values.

### Usage

```
fill_by_function(x, ..., fun = mean)
```

### Arguments

| | |
|---|---|
| x | A data frame. |
| ... | The unquoted column names of the variables that should be filled. |
| fun | The function to apply on the nonmissing values. |

### Value

x with the altered columns.

## Examples

```
library(dplyr) # for the pipe operator
x <- seq(as.Date('2016-01-01'), by = 'day', length.out = 366)
x <- x[sample(1:366, 200)] %>% sort
x_df <- data_frame(x  = x,
                   y1 = runif(200, 10, 20) %>% round,
                   y2 = runif(200, 1, 50) %>% round)
x_df %>% pad %>% fill_by_function(y1, y2)
x_df %>% pad %>% fill_by_function(y1, y2, fun = median)
```

---

`fill_by_prevalent` *Fill missing values by the most prevalent nonnmissing value.*

---

## Description

For each specified column in `x` replace the missing values by the most prevalent nonmissing value.

## Usage

```
fill_by_prevalent(x, ...)
```

## Arguments

| | |
|---|---|
| `x` | A data frame. |
| `...` | The unquoted column names of the variables that should be filled. |

## Value

`x` with the altered columns.

## Examples

```
library(dplyr) # for the pipe operator
x <- seq(as.Date('2016-01-01'), by = 'day', length.out = 366)
x <- x[sample(1:366, 200)] %>% sort
x_df <- data_frame(x  = x,
                   y1 = rep(letters[1:3], c(80, 70, 50)) %>% sample,
                   y2 = rep(letters[2:5], c(60, 80, 40, 20)) %>% sample)
x_df %>% pad %>% fill_by_prevalent(y1, y2)
```

## fill_by_value *Fill missing values by a single value.*

### Description

Replace all missing values in the specified columns by the same value.

### Usage

```
fill_by_value(x, ..., value = 0)
```

### Arguments

| | |
|---|---|
| x | A data frame. |
| ... | The unquoted column names of the variables that should be filled. |
| value | The value to replace the missing values by. |

### Value

x with the altered columns.

### Examples

```
library(dplyr) # for the pipe operator
x <- seq(as.Date('2016-01-01'), by = 'day', length.out = 366)
x <- x[sample(1:366, 200)] %>% sort
x_df <- data_frame(x  = x,
                   y1 = runif(200, 10, 20) %>% round,
                   y2 = runif(200, 1, 50) %>% round,
                   y3 = runif(200, 20, 40) %>% round,
                   y4 = sample(letters[1:5], 200, replace = TRUE))
x_padded <- x_df %>% pad
x_padded %>% fill_by_value(y1)
x_df %>% pad %>% fill_by_value(y1, y2, value = 42)
```

## get_interval *Get the interval of a datetime variable.*

### Description

The interval is the highest time unit that can explain all instances of a variable of class Date, class POSIXct, or class POSIXct. This function will determine what the interval of the variable is.

### Usage

```
get_interval(x)
```

### Arguments

| | |
|---|---|
| x | A variable of class of class Date or of class POSIXt. |

## Details

See `vignette("padr")` for more information on intervals.

## Value

A character string indicating the interval of `x`.

## Examples

```
x_month <- seq(as.Date('2016-01-01'), as.Date('2016-05-01'), by = 'month')
get_interval(x_month)

x_sec <- seq(as.POSIXct('2016-01-01 00:00:00'), length.out = 100, by = 'sec')
get_interval(x_sec)
```

---

# pad *Pad the datetime column of a data frame.*

---

## Description

pad will fill the gaps in incomplete datetime variables, by figuring out what the interval of the data is and what instances are missing. It will insert a record for each of the missing time points. For all other variables in the data frame a missing value will be insterted at the padded rows.

## Usage

```
pad(x, interval = NULL, start_val = NULL, end_val = NULL, by = NULL)
```

## Arguments

| | |
|---|---|
| `x` | A data frame containing at least one variable of class `Date`, class `POSIXct` or class `POSIXlt`. |
| `interval` | The interval of the returned datetime variable. When NULL the the interval will be equal to the interval of the datetime variable. When specified it can only be lower than the interval of the input data. See Details. |
| `start_val` | An object of class `Date`, class `POSIXct` or class `POSIXlt` that specifies the start of the returned datetime variable. If NULL it will use the lowest value of the input variable. |
| `end_val` | An object of class `Date`, class `POSIXct` or class `POSIXlt` that specifies the end of returned datetime variable. If NULL it will use the highest value of the input variable. |
| `by` | Only needs to be specified when `x` contains multiple variables of class `Date`, class `POSIXct` or class `POSIXlt`. `by` indicates which variable to use for padding. |

## Details

The interval of a datetime variable is the time unit at which the observations occur. The eight intervals in `padr` are from high to low `year`, `quarter`, `month`, `week`, `day`, `hour`, `min`, and `sec`. `pad` will figure out the interval of the input variable and will fill the gaps for the instances that would be expected from the interval, but are missing in the input data. See `vignette("padr")` for more information on `pad`. See `vignette("padr_implementation")` for detailed information on daylight savings time, different timezones, and the implementation of `thicken`.

## Value

The data frame x with the datetime variable padded. All other variables in the data frame will have missing values at the rows that are padded.

## Examples

```
simple_df <- data.frame(day = as.Date(c('2016-04-01', '2016-04-03')),
                        some_value = c(3,4))
pad(simple_df)

library(dplyr) # for the pipe operator
month <- seq(as.Date('2016-04-01'), as.Date('2017-04-01'),
             by = 'month')[c(1, 4, 5, 7, 9, 10, 13)]
month_df <- data.frame(month = month,
                       y = runif(length(month), 10, 20) %>% round)
# forward fill the padded values with tidyr's fill
month_df %>% pad %>% tidyr::fill(y)

# or fill all y with 0
month_df %>% pad %>% fill_by_value(y)

# padding a data.frame on group level
day_var <- seq(as.Date('2016-01-01'), length.out = 12, by = 'month')
x_df_grp <- data.frame(grp  = rep(LETTERS[1:3], each =4),
                       y    = runif(12, 10, 20) %>% round(0),
                       date = sample(day_var, 12, TRUE)) %>%
 arrange(grp, date)

x_df_grp %>% group_by(grp) %>% do(pad(.)) %>% ungroup %>%
tidyr::fill(grp)
```

---

`thicken` *Add a variable of a higher interval to a data frame.*

---

## Description

`thicken` will take the datetime variable in a data frame and map this to a variable of a higher interval. The mapping is added to the data frame in a new variable. After applying `thicken` the user can aggregate the other variables in the data frame to the higher interval, for instance using `dplyr`.

## Usage

```
thicken(x, interval = c("level_up", "year", "quarter", "month", "week", "day",
  "hour", "min"), colname = NULL, rounding = c("down", "up"), by = NULL,
  start_val = NULL)
```

## Arguments

| | |
|---|---|
| x | A data frame containing at least one datetime variable of class `Date`, class `POSIXct` or class `POSIXlt`. |

| | |
|---|---|
| interval | The interval of the added datetime variable, which should be higher than the interval of the input datetime variable. If NULL it will be one level higher than the interval of the input datetime variable. |
| colname | The column name of the added variable. If NULL it will be the name of the original datetime variable with the interval name added to it, separeted by an underscore. |
| rounding | Should a value in the input datetime variable be mapped to the closest value that is lower (down) or that is higher (up) than itself. |
| by | Only needs to be specified when x contains multiple variables of class Date, class POSIXct or class POSIXlt. by indicates which to use. |
| start_val | By default the first instance of interval that is lower than the lowest value of the input datetime variable, with all time units on default value. Specify start_val as an offset if you want the range to be nonstandard. |

## Details

See vignette("padr") for more information on thicken. See vignette("padr_implementation") for detailed information on daylight savings time, different timezones, and the implementation of thicken.

## Value

The data frame x with the variable added to it.

## Examples

```
x_hour <- seq(lubridate::ymd_hms('20160302 000000'), by = 'hour',
              length.out = 200)
some_df <- data.frame(x_hour = x_hour)
thicken(some_df)
thicken(some_df, 'month')
thicken(some_df, start_val = lubridate::ymd_hms('20160301 120000'))

library(dplyr)
x_df <- data.frame(
  x = seq(lubridate::ymd(20130101), by = 'day', length.out = 1000) %>%
    sample(500),
  y = runif(500, 10, 50) %>% round) %>%
  arrange(x)

# get the max per month
x_df %>% thicken('month') %>% group_by(x_month) %>%
  summarise(y_max = max(y))

# get the average per week, but you want your week to start on Mondays
# instead of Sundays
min_x <- x_df$x %>% min
weekdays(min_x)
x_df %>% thicken(start_val = min_x - 1) %>%
  group_by(x_week) %>% summarise(y_avg = mean(y))
```

# Index