# Apply functions by time

MANIPULATING TIME SERIES DATA WITH XTS AND ZOO IN R

**Jeffrey Ryan**
Creator of xts and quantmod

## Topics

- Applying functions on discrete periods or intervals
- Two main approaches
    - `period.apply()`
    - `split()`

## Apply by period

```
period.apply(x, INDEX, FUN, ...)
```

- `period.apply()` extends R's apply functions to time
- Pass an object `x` to be modified
- `INDEX` is the vector of **end points** of a period
- `FUN` is the function to apply
- Additional arguments are passed to `FUN` (if needed)

# Finding endpoints

```
endpoints(x, on = "years")
```

- Defined as the index of the last observation per interval
- Intervals are defined with the on argument
  - Uses "days", "years", "quarters", etc.
- **Always** starts on 0 and ends on the last observation

## period.apply() in action

```
edhec_4yr <- edhec["1997/2001"]
ep <- endpoints(edhec_4yr, "years")
period.apply(edhec_4yr, INDEX = ep, FUN = mean)
```

```
           Convertible Arbitrage
1997-12-31            0.01159167
1998-12-31            0.00270000
1999-12-31            0.01251667
2000-12-31            0.01377500
2001-12-31            0.01086667
```

Shortcut functions: `apply.monthly()` , `apply.yearly()` , `apply.quarterly()` , etc.

## split.xts

- Split data into chunks of time
- Great control for discrete periods
- Uses standard period names

```
# S3 method for class xts
split(x, f = "months")
edhec.qtrs <- split(edhec[, 1], f = "quarters")
edhec.qtrs[[3]]
```

```
           Convertible Arbitrage
1997-07-31                0.0193
1997-08-31                0.0134
1997-09-30                0.0122
```

## Let's practice!

# Converting periodicity

MANIPULATING TIME SERIES DATA WITH XTS AND ZOO IN R

**Jeffrey Ryan**
Creator of xts and quantmod

## Time series aggregation

- Useful to convert a univariate series to range bars
  - OHLC: Open, High, Low, and Close
- Summary of a particular period
  - Starting, maximum, minimum and ending value

# Aggregate using xts

```r
to.period(x,
          period = "months",
          k = 1,
          indexAt,
          name = NULL,
          OHLC = TRUE,
          ...)
```

- `period` controls aggregation period
- `name` string renames column roots
- `indexAt` allows for index alignment

## Aggregate OHLC

```r
to.period(edhec["1997/2001", 1], "years",
          name = "EDHEC")
```

```
           EDHEC.Open EDHEC.High EDHEC.Low EDHEC.Close
1997-12-31     0.0119     0.0212    0.0000      0.0068
1998-12-31     0.0145     0.0269   -0.0319      0.0113
1999-12-31     0.0219     0.0243    0.0045      0.0140
2000-12-31     0.0227     0.0267   -0.0081     -0.0002
2001-12-31     0.0344     0.0344   -0.0094     -0.0094
```

## Aggregate OHLC (cont.)

```r
to.period(edhec["1997/2001", 1], "years",
          name = "EDHEC", indexAt = "firstof")
```

```
           EDHEC.Open EDHEC.High EDHEC.Low EDHEC.Close
1997-12-01     0.0119     0.0212    0.0000      0.0068
1998-12-01     0.0145     0.0269   -0.0319      0.0113
1999-12-01     0.0219     0.0243    0.0045      0.0140
2000-12-01     0.0227     0.0267   -0.0081     -0.0002
2001-12-01     0.0344     0.0344   -0.0094     -0.0094
```

## Aggregate without range bars

- You can aggregate without range bars
- xts offers two main methods for this
  - Force a univariate series in `to.period()`

```
# OHLC = FALSE
to.period(object[, j], period = "years",
             name = "NAME", OHLC = FALSE)
```

- Extract the period values directly

```
# Extract directly
object[endpoints(object, "years"), j]
```

```r
# Using OHLC = FALSE
to.period(edhec[, 1], period = "years", name = "EDHEC",
          OHLC = FALSE)
```

```
           Convertible Arbitrage
1997-12-31                0.0068
1998-12-31                0.0113
1999-12-31                0.0140
```

```r
# Extract directly
edhec[endpoints(edhec, "years"), 1]
```

```
           Convertible Arbitrage
1997-12-31                0.0068
1998-12-31                0.0113
1999-12-31                0.0140
```

## Let's practice!

MANIPULATING TIME SERIES DATA WITH XTS AND ZOO IN R

# Rolling functions

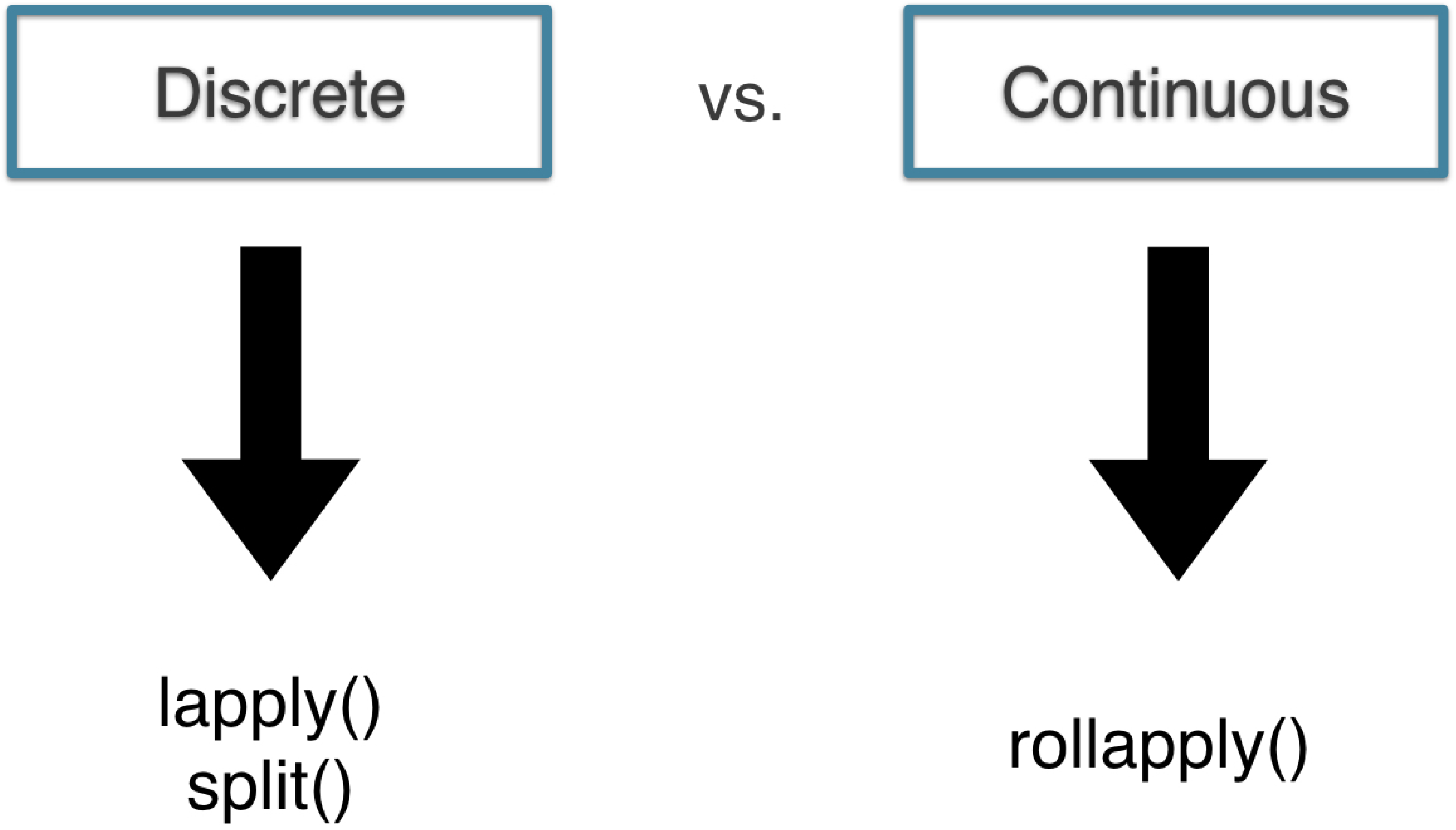
MANIPULATING TIME SERIES DATA WITH XTS AND ZOO IN R

**Jeffrey Ryan**
Creator of xts and quantmod

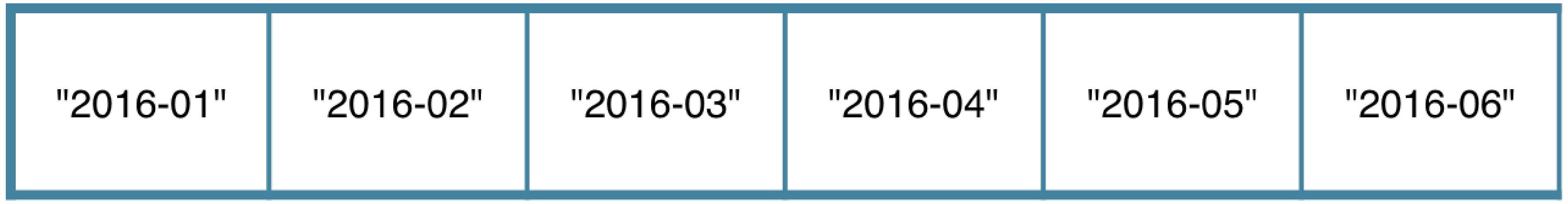
## Rolling windows

Discrete vs. Continuous

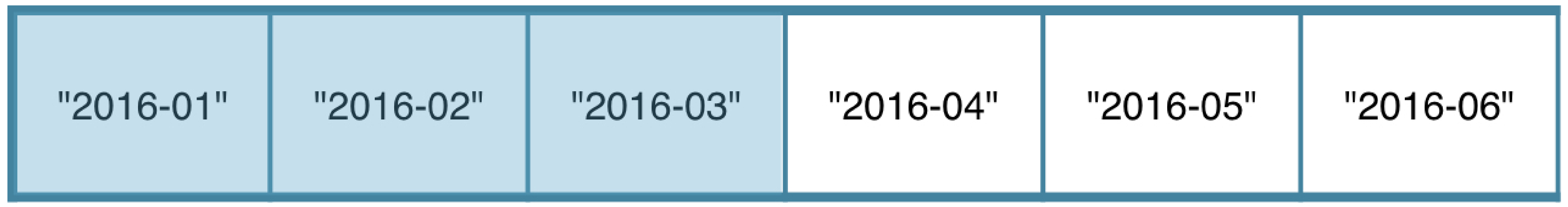
## Rolling windows



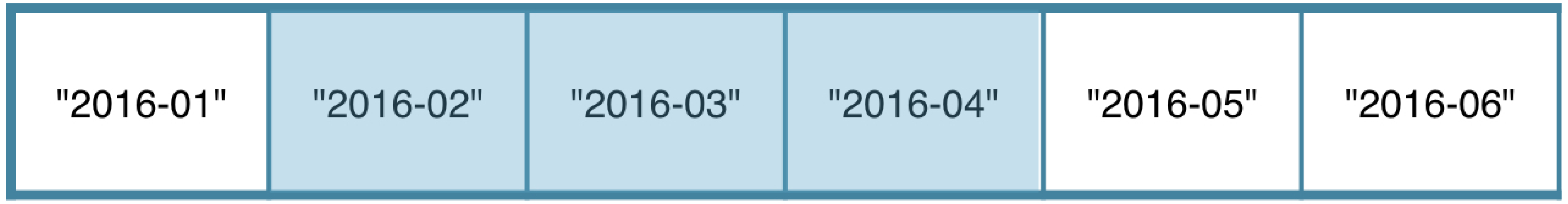
## Rolling windows



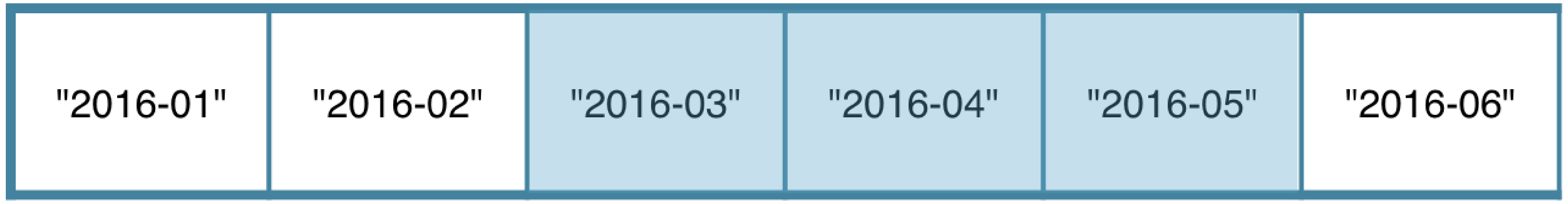
## Rolling windows



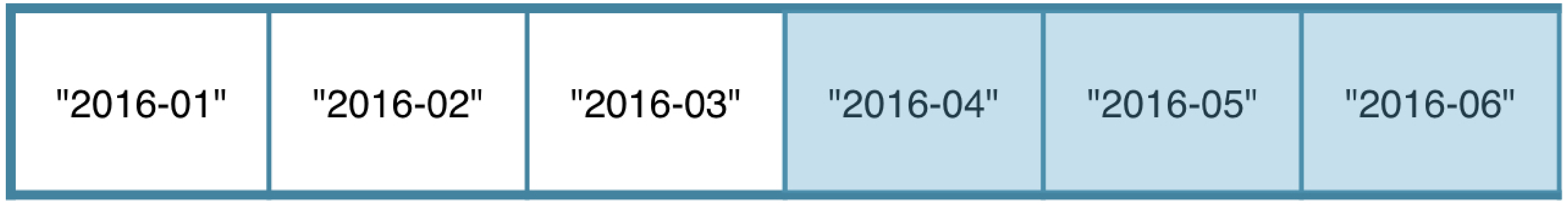
## Rolling windows



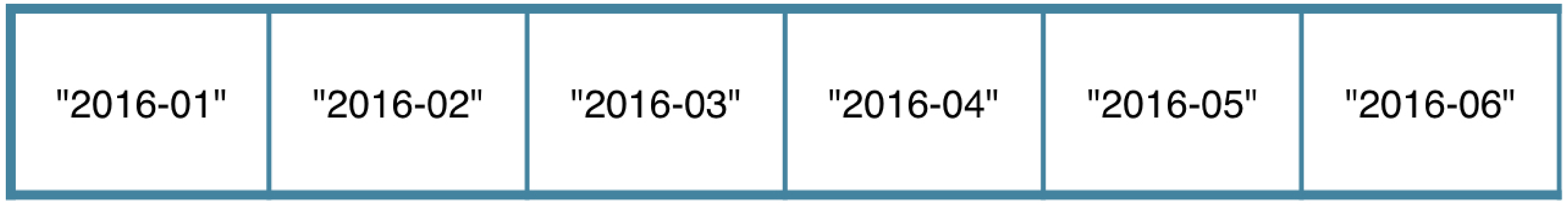
## Rolling windows



# Rolling windows



## Rolling windows



## Discrete rolling windows

- `split()` to break up by period
- `lapply()` **cumulative** functions
  - `cumsum()`, `cumprod()`, `cummin()`, `cummax()`

```
x <- xts(c(1, 2, 3), as.Date("2016-01-01") + 0:2)
cbind(x, cumsum(x))
```

```
           ..1 ..2
2016-01-01   1   1
2016-01-02   2   3
2016-01-03   3   6
```

# Discrete rolling windows

```r
edhec.yrs <- split(edhec[, 1], f = "years")
edhec.yrs <- lapply(edhec.yrs, cumsum)
edhec.ytd <- do.call(rbind, edhec.yrs)
```

```r
cbind(edhec.ytd, edhec[, 1])["2007-10/2008-03"]
```

```
           Convertible.Arbitrage Convertible.Arbitrage.1
2007-10-31                0.0594                  0.0177
2007-11-30                0.0463                 -0.0131
2007-12-31                0.0386                 -0.0077
2008-01-31               -0.0009                 -0.0009
2008-02-29               -0.0092                 -0.0083
2008-03-31               -0.0409                 -0.0317
```

## Continuous rolling windows

```
rollapply(data, width, FUN, ...,
          by = 1, by.column = TRUE,
          fill = if (na.pad) NA,
          na.pad = TRUE, partial = TRUE,
          align = c("right", "center", "left"))
```

- `data` is your xts object
- `width` is the the window size
- `FUN` is your function to apply
- Can add additional arguments to your function

## Continuous rolling windows

```r
rollapply(edhec["200701/08", 1], 3, mean)
```

```
           Convertible Arbitrage
2007-01-31                    NA
2007-02-28                    NA
2007-03-31           0.010233333
2007-04-30           0.006766667
2007-05-31           0.006533333
2007-06-30           0.004900000
2007-07-31           0.002266667
2007-08-31          -0.006233333
```

## Let's practice!

MANIPULATING TIME SERIES DATA WITH XTS AND ZOO IN R