Introducing xts and zoo objects

Contents


```
knitr::opts_chunk$set(warning = F, message = F,
                      out.width = "400px", out.height = "400px")
```

```
library(dplyr)
library(xts)
```

xts is an eXtensive Time Series.They have the main objects called "zoo" objects= A matrix object + a vector of times. This means an observation of the objects in the matrix in the times in that vector.

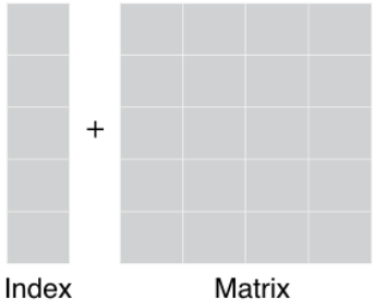

Figure 1: A matrix of objects are observed in that vector of times in index

XTS = matrix + index

Matrix of obsevations

```
x <- matrix(1:4, ncol=2, nrow = 2)
x
```

```
##      [,1] [,2]
## [1,]    1    3
## [2,]    2    4
```

Index (vector) of times These must be a time object.

```
idx <- as.Date(c("2015-01-01", "2015-02-01"))
idx
```

```
## [1] "2015-01-01" "2015-02-01"
```

XTS

```
X <- xts(x,               #x must be matrix or a vector
         order.by = idx) #idx must be increasing time object same length as x
X
```

```
##            [,1] [,2]
## 2015-01-01    1    3
## 2015-02-01    2    4
```

Birinci gün vektöründe 1 ile 3'üncü gözlemler, ikinci gün vektöründe 2 ile 4'üncü gözlemler yapılmış.

```
xts(x = NULL,             #an object containing the time series data
    order.by = index(x), #order by vector of times in increasing order of time
    frequency = NULL,     #
    unique = TRUE,        #force times to be unique
    tzone = Sys.getenv("TZ"), #set the time zone
    )
```

Interestingly, if you take a subset from the matrix, xts will preserve the timestamps if there are any.

XTS içinden matrixi `zoo::coredata(x, fmt=F)`, dateleri almak için `zoo::index(x)` kullanılır.

Examples

```
data <- rnorm(15) #create a vector of random 15 numbers

dates <- seq(as.Date("2020-01-01"), #create  a sequence of dates from
             length = 15,           #for 15 times
             by = "days")           #in days (15 days)


ex <- xts(x = data,             #create a xts from the data matrix
          order.by = dates,     #order by the dates as index
          bday = as.POSIXct("2020-01-01")) #special bday marker in that day

ex
```

```
##                  [,1]
## 2020-01-01  0.6116791
## 2020-01-02  0.2978292
## 2020-01-03  1.1075513
## 2020-01-04 -0.7816579
## 2020-01-05  0.5002649
## 2020-01-06 -1.7996792
## 2020-01-07  0.3763857
## 2020-01-08 -1.9680027
## 2020-01-09 -0.6850166
## 2020-01-10 -0.3333967
## 2020-01-11 -0.2828041
## 2020-01-12 -1.1821579
## 2020-01-13  0.7145466
## 2020-01-14  1.2693460
## 2020-01-15 -1.1446270
```

Extract the data and the date

```
data_extract <- coredata(ex)
data_extract
```

```
##             [,1]
##  [1,]  0.6116791
##  [2,]  0.2978292
##  [3,]  1.1075513
##  [4,] -0.7816579
##  [5,]  0.5002649
##  [6,] -1.7996792
##  [7,]  0.3763857
##  [8,] -1.9680027
##  [9,] -0.6850166
## [10,] -0.3333967
## [11,] -0.2828041
## [12,] -1.1821579
## [13,]  0.7145466
## [14,]  1.2693460
## [15,] -1.1446270
```

```
date_extract <- index(ex)
date_extract
```

```
##  [1] "2020-01-01" "2020-01-02" "2020-01-03" "2020-01-04" "2020-01-05"
##  [6] "2020-01-06" "2020-01-07" "2020-01-08" "2020-01-09" "2020-01-10"
## [11] "2020-01-11" "2020-01-12" "2020-01-13" "2020-01-14" "2020-01-15"
```

Importing, exporting and converting time series

Sunspot verisini `as.xts()` ile xts'e çevirebiliriz.

```
sunspots_xts <- as.xts(sunspots)
head(sunspots_xts)
```

```
##          [,1]
## Oca 1749 58.0
## Şub 1749 62.6
## Mar 1749 70.0
## Nis 1749 55.7
## May 1749 85.0
## Haz 1749 83.5
```

Bir dosyayı okurken de `as.xts(read.csv())` formatıyla doğrudan xts olarak yükleyebiliriz.

```
# Create data by reading tmp_file
data <- read.csv(tmp_file)

# Convert data into xts
xts(data, order.by = as.Date(rownames(dat), "%m/%d/%Y"))

# Read tmp_file using read.zoo and as.xts
data_zoo <- read.zoo(tmp_file, index.column = 0, sep = ",", format = "%m/%d/%Y")

# Read tmp_file using read.zoo and as.xts
data_xts <- as.xts(data_zoo)
```

The ISO-8601 standard

Bu standart metodda soldan başlanarak sağa doğru tarih büyükten küçüğe yazılır. "YYYY-MM-DD-HH-MM-SS"

```
edhec <- PerformanceAnalytics::edhec
periodicity(edhec)
```

```
## Monthly periodicity from 1997-01-31 to 2019-11-30
```

```
head(edhec)
```

```
##            Convertible Arbitrage CTA Global Distressed Securities
## 1997-01-31                0.0119     0.0393                0.0178
## 1997-02-28                0.0123     0.0298                0.0122
## 1997-03-31                0.0078    -0.0021               -0.0012
## 1997-04-30                0.0086    -0.0170                0.0030
## 1997-05-31                0.0156    -0.0015                0.0233
## 1997-06-30                0.0212     0.0085                0.0217
##            Emerging Markets Equity Market Neutral Event Driven
## 1997-01-31           0.0791                0.0189       0.0213
## 1997-02-28           0.0525                0.0101       0.0084
## 1997-03-31          -0.0120                0.0016      -0.0023
## 1997-04-30           0.0119                0.0119      -0.0005
## 1997-05-31           0.0315                0.0189       0.0346
```

```
## 1997-06-30           0.0581                0.0165       0.0258
##            Fixed Income Arbitrage Global Macro Long/Short Equity
## 1997-01-31                 0.0191       0.0573            0.0281
## 1997-02-28                 0.0122       0.0175           -0.0006
## 1997-03-31                 0.0109      -0.0119           -0.0084
## 1997-04-30                 0.0130       0.0172            0.0084
## 1997-05-31                 0.0118       0.0108            0.0394
## 1997-06-30                 0.0108       0.0218            0.0223
##            Merger Arbitrage Relative Value Short Selling Funds of Funds
## 1997-01-31           0.0150         0.0180       -0.0166         0.0317
## 1997-02-28           0.0034         0.0118        0.0426         0.0106
## 1997-03-31           0.0060         0.0010        0.0778        -0.0077
## 1997-04-30          -0.0001         0.0122       -0.0129         0.0009
## 1997-05-31           0.0197         0.0173       -0.0737         0.0275
## 1997-06-30           0.0231         0.0198       -0.0065         0.0225
```

```r
A["20090825"]       ## Aug 25, 2009
A["201203/201212"]  ## Mar to Dec 2012
A["201601/"]        ## Up to and including January 2016

# Extract all data between 8AM and 10AM
morn_2010 <- irreg["T08:00/T10:00"]

# Extract the observations for January 13th, 2010
morn_2010["2010-01-13"]

#Ama tüm tarihler data["YYYY-MM-DD"] olarak daha net anlaşılır.
```

Row selection with time objects

POSIXct günü ve saati tutan bir date objesi türüdür.

Head ve tail fonksiyonlarındaki gibi ilk ve son n veriyi `xts::first()` ve `xts::last()` ile alabiliriz.

```r
first(edhec$`Funds of Funds`, #first
      "3 months")             #3 months
```

```
##            Funds of Funds
## 1997-01-31         0.0317
## 1997-02-28         0.0106
## 1997-03-31        -0.0077
```

```r
last(as.xts(edhec$`Funds of Funds`), #last
     n = 4)                          #4 data
```

```
##            Funds of Funds
## 2019-08-31        -0.0063
## 2019-09-30        -0.0033
## 2019-10-31         0.0035
## 2019-11-30         0.0071
```

```
last(edhec$`Funds of Funds`, #last
     "3 years")              #3 years
```

```
##            Funds of Funds
## 2017-01-31         0.0095
## 2017-02-28         0.0080
## 2017-03-31         0.0041
## 2017-04-30         0.0057
## 2017-05-31         0.0033
## 2017-06-30        -0.0006
## 2017-07-31         0.0097
## 2017-08-31         0.0075
## 2017-09-30         0.0039
## 2017-10-31         0.0113
## 2017-11-30        -0.0007
## 2017-12-31         0.0064
## 2018-01-31         0.0209
## 2018-02-28        -0.0127
## 2018-03-31        -0.0044
## 2018-04-30         0.0022
## 2018-05-31         0.0073
## 2018-06-30        -0.0037
## 2018-07-31         0.0018
## 2018-08-31         0.0015
## 2018-09-30        -0.0022
## 2018-10-31        -0.0269
## 2018-11-30        -0.0071
## 2018-12-31        -0.0163
## 2019-01-31         0.0233
## 2019-02-28         0.0090
## 2019-03-31         0.0075
## 2019-04-30         0.0086
## 2019-05-31        -0.0083
## 2019-06-30         0.0137
## 2019-07-31         0.0037
## 2019-08-31        -0.0063
## 2019-09-30        -0.0033
## 2019-10-31         0.0035
## 2019-11-30         0.0071
```

```
first(as.xts(edhec$`Funds of Funds`),
      "-20 years") #everything but the first 20 years
```

```
##            Funds of Funds
## 2017-01-31         0.0095
## 2017-02-28         0.0080
## 2017-03-31         0.0041
## 2017-04-30         0.0057
## 2017-05-31         0.0033
## 2017-06-30        -0.0006
## 2017-07-31         0.0097
## 2017-08-31         0.0075
```

```
## 2017-09-30         0.0039
## 2017-10-31         0.0113
## 2017-11-30        -0.0007
## 2017-12-31         0.0064
## 2018-01-31         0.0209
## 2018-02-28        -0.0127
## 2018-03-31        -0.0044
## 2018-04-30         0.0022
## 2018-05-31         0.0073
## 2018-06-30        -0.0037
## 2018-07-31         0.0018
## 2018-08-31         0.0015
## 2018-09-30        -0.0022
## 2018-10-31        -0.0269
## 2018-11-30        -0.0071
## 2018-12-31        -0.0163
## 2019-01-31         0.0233
## 2019-02-28         0.0090
## 2019-03-31         0.0075
## 2019-04-30         0.0086
## 2019-05-31        -0.0083
## 2019-06-30         0.0137
## 2019-07-31         0.0037
## 2019-08-31        -0.0063
## 2019-09-30        -0.0033
## 2019-10-31         0.0035
## 2019-11-30         0.0071
```

```
# Extract the first three days of the second week of temps
first(last(first(dat, "2 weeks"), "1 week"), "3 days")
```

```
#Her 1 yılda bir değer göster
edhec[endpoints(edhec,              #son değerler
                k = 1,              #her 1
                on = "years"), 13] #yılda bir
```

```
##            Funds of Funds
## 1997-12-31         0.0089
## 1998-12-31         0.0222
## 1999-12-31         0.0622
## 2000-12-31         0.0133
## 2001-12-31         0.0099
## 2002-12-31         0.0077
## 2003-12-31         0.0139
## 2004-12-31         0.0145
## 2005-12-31         0.0191
## 2006-12-31         0.0175
## 2007-12-31         0.0040
## 2008-12-31        -0.0119
## 2009-12-31         0.0066
## 2010-12-31         0.0205
## 2011-12-31        -0.0054
## 2012-12-31         0.0108
```

```
## 2013-12-31         0.0109
## 2014-12-31         0.0021
## 2015-12-31        -0.0059
## 2016-12-31         0.0090
## 2017-12-31         0.0064
## 2018-12-31        -0.0163
## 2019-11-30         0.0071
```

```
#Her 2 çeyrekteki bir değer göster
edhec[endpoints(edhec,                #son değerler
                k = 2,                #her 2
                on = "quarters"), 13] #çeyrekte bir
```

```
##            Funds of Funds
## 1997-06-30         0.0225
## 1997-12-31         0.0089
## 1998-06-30         0.0021
## 1998-12-31         0.0222
## 1999-06-30         0.0282
## 1999-12-31         0.0622
## 2000-06-30         0.0311
## 2000-12-31         0.0133
## 2001-06-30         0.0013
## 2001-12-31         0.0099
## 2002-06-30        -0.0095
## 2002-12-31         0.0077
## 2003-06-30         0.0068
## 2003-12-31         0.0139
## 2004-06-30         0.0034
## 2004-12-31         0.0145
## 2005-06-30         0.0131
## 2005-12-31         0.0191
## 2006-06-30        -0.0028
## 2006-12-31         0.0175
## 2007-06-30         0.0082
## 2007-12-31         0.0040
## 2008-06-30        -0.0068
## 2008-12-31        -0.0119
## 2009-06-30         0.0024
## 2009-12-31         0.0066
## 2010-06-30        -0.0079
## 2010-12-31         0.0205
## 2011-06-30        -0.0138
## 2011-12-31        -0.0054
## 2012-06-30        -0.0037
## 2012-12-31         0.0108
## 2013-06-30        -0.0133
## 2013-12-31         0.0109
## 2014-06-30         0.0091
## 2014-12-31         0.0021
## 2015-06-30        -0.0108
## 2015-12-31        -0.0059
## 2016-06-30        -0.0074
## 2016-12-31         0.0090
```

```
## 2017-06-30        -0.0006
## 2017-12-31         0.0064
## 2018-06-30        -0.0037
## 2018-12-31        -0.0163
## 2019-06-30         0.0137
## 2019-11-30         0.0071
```

```
head(index(edhec))
```

```
## [1] "1997-01-31" "1997-02-28" "1997-03-31" "1997-04-30" "1997-05-31"
## [6] "1997-06-30"
```

```
xts::tformat(edhec) <- "%d/%m/%Y %H:%M%:%S"
```

```
# Add a to b, and fill all missing rows of b with 0
a + merge(b, index(a), fill = 0)

# Add a to b and fill NAs with the last observation
a + merge(b, index(a), fill = na.locf)
```

Merging time series

Figure 2: Merge() function

rbind() only excepts dates

```
# Perform an inner join of a and b
merge(a, b, join = "inner")

# Perform a left-join of a and b, fill missing values with 0
merge(a, b, join = "left", fill = 0)
```

Handling missing values

l.o.c.f: last observation carried forward.

```
merge(x, c(2, 3, 4))
```

```
           x c.2..3..4.
2016-08-09 1          2
2016-08-10 1          3
2016-08-11 1          4
```

```
merge(x, 3)
```

```
           x X3
2016-08-09 1  3
2016-08-10 1  3
2016-08-11 1  3
```

```
merge(x, as.Date(c("2016-08-14")))
```

```
           x
2016-08-09  1
2016-08-10  1
2016-08-11  1
2016-08-14 NA
```

Figure 3: Merge() function

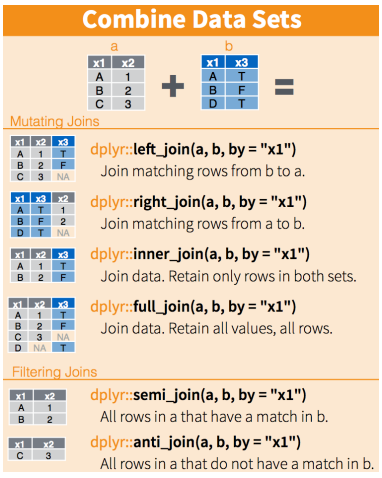

Figure 4: Join mantığı

```
zoo::na.locf(object,
        na.rm = TRUE,  #delete the observations that don't get filled (NAs followed by an NA)
        fromLast = FALSE, #reverse, take the last observation and carry before
        maxgap = 5)       #fill if > 5 consecutive NAs, < 5 will be unchanged
```

```
cbind(z, na.locf(z), na.locf(z, fromLast = TRUE))
```

```
           z z.1 z.2
2016-08-09 1   1   1
2016-08-10 NA  1   4
2016-08-11 NA  1   4
2016-08-12 4   4   4
```

Figure 5: z sütunundaki iki NA verisine z.1 sütunundaki düz locf ile NA'dan önceki 1, z.2 sütunundaki ters locf ile NA'dan sonraki 4 eklenmiş

Diğer NA seçenekleri

```
zoo::na.fill(object,          #bu objedeki tüm NA değerlerini
             fill = VALUE)    #VALUE ile doldur

zoo::na.trim(object,
             sides = c("left", "right", "both"), #left zaman, right gözlem trim ?
             is.na = c("any", "all"), #içindeki 1 tane değerde bile NA olan satırı silmek için any, tüm
             maxgap = n) # n kadar art arda gelen NA değerleri varsa tüm o satırları sil, yoksa bir şey
```

Leads and Lags

Lead bir gün ileriye almak (a day leading to today)
Lag bir gün geriye almak (a day lagging from today) demektir.

Bugünün verisini dünün verisiyle karşılaştırmak için `lag()` operatörü kullanılır. Ne kadar geriye veya ileriye gidileceği `k` değeriyle belirtilir; negatif ise geri, pozitif ise ileri götürür. Birden fazla k değerini birleştirip o tarihlerin damgaları kıyaslanabilir (e.g., 0,1 ve 5 lag/leadleri).

R'ın kendi time series(ts) veya zoo verileri için de lag kavramları vardır ama bunlar XTS'e göre ters kodlanır; -1 ile bir ileri lead, 1 ile bir geri lag kodlanır. Bu yüzden ts veya zoo verisini XTS'e çevirdikten sonra ters kodla devam etmek gerekir.

Diff

XTS serisindeki farklılıkları (e.g., hisse senedindeki günlük fark) hesaplamak için `diff()` fonksiyonu kullanılır.

```
diff(xts object,          #
     lag = 1,             #kaç günlüklük fark çıkarılacak (2 gün ise 3-1,4-2,...)
     differences = 1,     #kaç kere uygulanacak
     arithmetic = FALSE,  #değerleri aritmatik dönüşüm
     log = FALSE,         #değerleri logaritmik dönüşüm
     na.pad )       #
```

`diff()` çalışırken bir önceki tarihin verisi bulmak için `lag()` da çalıştırır.

Seasonality: Kışın sıcaklıkların düşmesi gibi belli olayların belli dönemlerde görülmesi.
Stationarity: Bir olayın belli bir aralığa sıkışması.

```
lead_x ← lag(x, k = -1)

lag_x  ← lag(x, k = 1)

z ← merge(lead_x, x, lag_x)

z

            x x.1 x.2
2022-02-02  6   5  NA
2022-02-03  7   6   5
2022-02-04  8   7   6
2022-02-05  9   8   7
2022-02-06 NA   9   8
```

Figure 6: x lead (+1), x, x lag (-1)

```
head(
  merge(as.xts(AirPassengers),
        diff(as.xts(AirPassengers), #AirPassanger datasındaki her bir veriyi
             lag = 3,               #3 önceki veri ile
             differences = 1)),     #1 kere çıkararak ver
  10)
```

```
##          as.xts.AirPassengers.
## Oca 1949                   112
## Şub 1949                   118
## Mar 1949                   132
## Nis 1949                   129
## May 1949                   121
## Haz 1949                   135
## Tem 1949                   148
## Ağu 1949                   148
## Eyl 1949                   136
## Eki 1949                   119
##          diff.as.xts.AirPassengers...lag...3..differences...1.
## Oca 1949                                                    NA
## Şub 1949                                                    NA
## Mar 1949                                                    NA
## Nis 1949                                                    17
## May 1949                                                     3
## Haz 1949                                                     3
## Tem 1949                                                    19
## Ağu 1949                                                    27
## Eyl 1949                                                     1
## Eki 1949                                                   -29
```

Yukarıdaki `lag(k=3)` çıkarma işlemi satırlarda 4 - 1, 5 - 2 olarak devam ediyor.

```
periodicity(as.xts(AirPassengers))
```

```
## Monthly periodicity from Oca 1949 to Ara 1960
```

```
nseconds(x)
nminutes(x)
nhours(x)
ndays(x)
nweeks(x)
nmonths(x)
nquarters(x)
nyears(x)
```

Figure 7: Zamanların sayısı

```
last5 <- as.xts(AirPassengers)["1955/1960"]   #take the last 5 years
periodicity(last5)
```

```
## Monthly periodicity from Oca 1955 to Ara 1960
```

```
head(last5)
```

```
##          [,1]
## Oca 1955  242
## Şub 1955  233
## Mar 1955  267
## Nis 1955  269
## May 1955  270
## Haz 1955  315
```

```
ep <- endpoints(last5, on = "months", k = 6)  #a point every 6 months
ep #it is interesting that every six months is coded as 6*x instead of relevant dates
```

```
##  [1]  0  6 12 18 24 30 36 42 48 54 60 66 72
```

```
period.apply(last5,       #on the last5
             INDEX = ep,  #take the index points indicated periods of time in ep
             FUN = mean)  #take the mean of the values for that period
```

```
##              [,1]
## Haz 1955 266.0000
## Ara 1955 302.0000
## Haz 1956 313.8333
## Ara 1956 342.6667
## Haz 1957 349.5000
## Ara 1957 387.3333
## Haz 1958 361.0000
## Ara 1958 401.0000
## Haz 1959 399.3333
## Ara 1959 457.3333
## Haz 1960 449.1667
## Ara 1960 503.1667
```

```
#it'd still work if we just c(0,6,12,18,...)
xd <- c(0,6,12,18)
period.apply(last5, INDEX = xd, FUN = mean)
```

```
##              [,1]
## Haz 1955 266.0000
## Ara 1955 302.0000
## Haz 1956 313.8333
## Ara 1960 405.6111
```

Bütün bu işlemleri tek bir kodla yapmak için `xts::apply.monthly()` veya türevleri olan `xts::apply.yearly()` , `xts::apply.quarterly()`, .. vs kullanılabilir; endpoints işlemlerini kendiliğinden yapar.

```
apply.yearly(last5, mean) #Apply mean on every year point (yearly)
```

```
##              [,1]
## Ara 1955 284.0000
## Ara 1956 328.2500
## Ara 1957 368.4167
## Ara 1958 381.0000
## Ara 1959 428.3333
## Ara 1960 476.1667
```

split.xts

Datayı kendi içinde belli tarih aralıklarına (periodlara) bölmek için kullanılır.

```
head(split.xts(as.xts(AirPassengers), #AirPassanger datasını
          f = "years"))               #Kendi içinde yıllara ayır
```

```
## [[1]]
##          [,1]
## Oca 1949  112
```

```
## Şub 1949  118
## Mar 1949  132
## Nis 1949  129
## May 1949  121
## Haz 1949  135
## Tem 1949  148
## Ağu 1949  148
## Eyl 1949  136
## Eki 1949  119
## Kas 1949  104
## Ara 1949  118
##
## [[2]]
##          [,1]
## Oca 1950  115
## Şub 1950  126
## Mar 1950  141
## Nis 1950  135
## May 1950  125
## Haz 1950  149
## Tem 1950  170
## Ağu 1950  170
## Eyl 1950  158
## Eki 1950  133
## Kas 1950  114
## Ara 1950  140
##
## [[3]]
##          [,1]
## Oca 1951  145
## Şub 1951  150
## Mar 1951  178
## Nis 1951  163
## May 1951  172
## Haz 1951  178
## Tem 1951  199
## Ağu 1951  199
## Eyl 1951  184
## Eki 1951  162
## Kas 1951  146
## Ara 1951  166
##
## [[4]]
##          [,1]
## Oca 1952  171
## Şub 1952  180
## Mar 1952  193
## Nis 1952  181
## May 1952  183
## Haz 1952  218
## Tem 1952  230
## Ağu 1952  242
## Eyl 1952  209
## Eki 1952  191
```

```
## Kas 1952  172
## Ara 1952  194
##
## [[5]]
##          [,1]
## Oca 1953  196
## Şub 1953  196
## Mar 1953  236
## Nis 1953  235
## May 1953  229
## Haz 1953  243
## Tem 1953  264
## Ağu 1953  272
## Eyl 1953  237
## Eki 1953  211
## Kas 1953  180
## Ara 1953  201
##
## [[6]]
##          [,1]
## Oca 1954  204
## Şub 1954  188
## Mar 1954  235
## Nis 1954  227
## May 1954  234
## Haz 1954  264
## Tem 1954  302
## Ağu 1954  293
## Eyl 1954  259
## Eki 1954  229
## Kas 1954  203
## Ara 1954  229
```

Time series aggregation

Belli periodlardaki verilerin aritmatik analizini yaparken mean, max, min vs gibi değerleri hesaplayıp incelemek gerekir.

- OHLC: Open, High, Low, and Close

  Time series hesaplamalarını yaparken `xts::to.period()` fonksiyonu kullanılır.

```
to.period(xtsobject,
          period = "months", #Hangi aralıklarla hesap yapılacak
          k = 1,             #
          indexAt = ,        #Index sırasını periyodun başlangıcında ceya bitişinde ayarlamak istersen
          name = NULL,       #OHLC verilerinin sütunlarının ismini değiştirmek istersen
          OHLC = TRUE)       #
```

```
to.period(edhec["1997/2001", 11], #97-01 edhec datasının ilk sütununda OHLC verisi
          period = "years",       #yıllık periodlarda
          name = "EDHEC")         #EDHEC sütun başlığıyla
```

```
##                     EDHEC.Open EDHEC.High EDHEC.Low EDHEC.Close
## 31/12/1997 00:00:00     0.0180     0.0198    0.0010      0.0082
## 31/12/1998 00:00:00     0.0132     0.0198   -0.0341      0.0164
## 31/12/1999 00:00:00     0.0195     0.0238    0.0062      0.0183
## 31/12/2000 00:00:00     0.0173     0.0185   -0.0004      0.0075
## 31/12/2001 00:00:00     0.0333     0.0333   -0.0221      0.0097
```

```
to.period(edhec["1997/2001", 11],
          period = "years",
          OHLC = F)             #OHLC = F sadece Close toplamını verir
```

```
##                     Relative Value
## 31/12/1997 00:00:00         0.0082
## 31/12/1998 00:00:00         0.0164
## 31/12/1999 00:00:00         0.0183
## 31/12/2000 00:00:00         0.0075
## 31/12/2001 00:00:00         0.0097
```

```
edhec[endpoints(edhec, "years"), 11]
```

```
##                     Relative Value
## 31/12/1997 00:00:00         0.0082
## 31/12/1998 00:00:00         0.0164
## 31/12/1999 00:00:00         0.0183
## 31/12/2000 00:00:00         0.0075
## 31/12/2001 00:00:00         0.0097
## 31/12/2002 00:00:00         0.0023
## 31/12/2003 00:00:00         0.0127
## 31/12/2004 00:00:00         0.0099
## 31/12/2005 00:00:00         0.0126
## 31/12/2006 00:00:00         0.0128
## 31/12/2007 00:00:00         0.0022
## 31/12/2008 00:00:00         0.0031
## 31/12/2009 00:00:00         0.0161
## 31/12/2010 00:00:00         0.0157
## 31/12/2011 00:00:00         0.0012
## 31/12/2012 00:00:00         0.0117
## 31/12/2013 00:00:00         0.0102
## 31/12/2014 00:00:00        -0.0016
## 31/12/2015 00:00:00        -0.0067
## 31/12/2016 00:00:00         0.0094
## 31/12/2017 00:00:00         0.0055
## 31/12/2018 00:00:00        -0.0090
## 30/11/2019 00:00:00         0.0068
```

```
to.period(edhec["1997/2001", 11], #OHLC bilgilerini
          period = "years",       #yıllık periodlarla
          indexAt = "firstof",    #periyodun başından başlayarak
          name = "EDD")           #EDD ismiyle
```

```
##            EDD.Open EDD.High EDD.Low EDD.Close
## 1997-12-01   0.0180   0.0198  0.0010    0.0082
```

```
## 1998-12-01   0.0132   0.0198 -0.0341    0.0164
## 1999-12-01   0.0195   0.0238  0.0062    0.0183
## 2000-12-01   0.0173   0.0185 -0.0004    0.0075
## 2001-12-01   0.0333   0.0333 -0.0221    0.0097
```

Her üç ayın verisini hesaplarken 1+2+3, 2+3+4, 3+4+5 diye devam eder.

`zoo::rollappy()`: Belli aralıklardaki hesapları tekrarlayarak yapmak

```
rollapply(head(edhec$`Funds of Funds`, 10), #edhec verisinde devinimli hesabı
          width = 3,  #her 3 veride
          FUN = mean) #ortalamasını al
```

```
##            Funds of Funds
## 1997-01-31             NA
## 1997-02-28             NA
## 1997-03-31    0.011533333
## 1997-04-30    0.001266667
## 1997-05-31    0.006900000
## 1997-06-30    0.016966667
## 1997-07-31    0.031166667
## 1997-08-31    0.023700000
## 1997-09-30    0.027333333
## 1997-10-31    0.009533333
```