datacamp

R For Data Science

xts Cheat Sheet

Learn xts online at www.DataCamp.com

xts

eXtensible Time Series (xts) is a powerful package that provides an extensible time series class, enabling uniform handling of many R time series classes by extending zoo.

Load the package as follows:

```
> library(xts)
```

xts Objects

xts objects have three main components:
- **coredata:** always a matrix for xts objects, while it could also be a vector for zoo objects
- **index:** vector of any `Date`, `POSIXct`, `chron`, `yearmon`, `yearqtr`, or `DateTime` classes
- **xtsAttributes:** arbitrary attributes

> Creating xts Objects

```
> xts1 ← xts(x=1:10, order.by=Sys.Date()-1:10)
> data ← rnorm(5)
> dates ← seq(as.Date("2017-05-01"),length=5,by="days")
> xts2 ← xts(x=data, order.by=dates)
> xts3 ← xts(x=rnorm(10),
             order.by=as.POSIXct(Sys.Date()+1:10),
             born=as.POSIXct("1899-05-08"))
> xts4 ← xts(x=1:10, order.by=Sys.Date()+1:10)
```

Convert To And From xts

```
> data(AirPassengers)
> xts5 ← as.xts(AirPassengers)
```

Import From Files

```
> dat ← read.csv(tmp_file)
> xts(dat, order.by=as.Date(rownames(dat),"%m/%d/%Y"))
> dat_zoo ← read.zoo(tmp_file,
                     index.column=0,
                     sep=",", format="%m/%d/%Y")
> dat_zoo ← read.zoo(tmp,sep=",",FUN=as.yearmon)
> dat_xts ← as.xts(dat_zoo)
```

> Export xts Objects

```
> data_xts ← as.xts(matrix)
> tmp ← tempfile()
> write.zoo(data_xts,sep=",",file=tmp)
```

> Replace & Update

```
> xts2[dates] ← 0 #Replace values in xts2 on dates with 0
> xts5["1961"] ← NA #Replace dates from 1961 with NA
> xts2["2016-05-02"] ← NA #Replace the value at 1 specific index with NA
```

> Applying Functions

```
> ep1 ← endpoints(xts4,on="weeks",k=2) #Take index values by time
 [1] 0 5 10
> ep2 ← endpoints(xts5,on="years")
 [1] 0 12 24 36 48 60 72 84 96 108 120 132 144
> period.apply(xts5,INDEX=ep2,FUN=mean) #Calculate the yearly mean
> xts5_yearly ← split(xts5,f="years") #Split xts5 by year
> lapply(xts5_yearly,FUN=mean) #Create a list of yearly means
> do.call(rbind, #Find the last observation in each year in xts5
          lapply(split(xts5,"years"),
          function(w) last(w,n="1 month")))
> do.call(rbind, #Calculate cumulative annual passengers
          lapply(split(xts5,"years"),
          cumsum))
> rollapply(xts5, 3, sd) #Apply sd to rolling margins of xts5
```

> Selecting, Subsetting & Indexing

Select

```
> mar55 ← xts5["1955-03"] #Get value for March 1955
```

Subset

```
> xts5_1954 ← xts5["1954"] #Get all data from 1954
> xts5_janmarch ← xts5["1954/1954-03"] #Extract data from Jan to March '54
> xts5_janmarch ← xts5["/1954-03"] #Get all data until March '54
> xts4[ep1] #Subset xts4 using ep2
```

first() and last()

```
> first(xts4,'1 week') #Extract first 1 week
> first(last(xts4,'1 week'),'3 days') #Get first 3 days of the last week of data
```

Indexing

```
> xts2[index(xts3)] #Extract rows with the index of xts3
> days ← c("2017-05-03","2017-05-23")
> xts3[days] #Extract rows using the vector days
> xts2[as.POSIXct(days,tz="UTC")] #Extract rows using days as POSIXct
> index ← which(.indexwday(xts1)==0|.indexwday(xts1)==6) #Index of weekend days
> xts1[index] #Extract weekend days of xts1
```

> Missing Values

```
> na.omit(xts5) #Omit NA values in xts5
> xts_last ← na.locf(xts2) #Fill missing values in xts2 using last observation
> xts_last ← na.locf(xts2, #Fill missing values in xts2 using next observation
                     fromLast=TRUE)
> na.approx(xts2) #Interpolate NAs using linear approximation
```

> Arithmetic Operations

coredata()or as.numeric()

```
> xts3 + as.numeric(xts2) #Addition
> xts3 * as.numeric(xts4) #Multiplication
> coredata(xts4) - xts3 #Subtraction
> coredata(xts4) / xts3 #Division
```

Shifting Index Values

```
> xts5 - lag(xts5) #Period-over-period differences
> diff(xts5,lag=12,differences=1) #Lagged differences
```

Reindexing

```
> xts1 + merge(xts2,index(xts1),fill=0) #Addition
              e1
 2017-05-04 5.231538
 2017-05-05 5.829257
 2017-05-06 4.000000
 2017-05-07 3.000000
 2017-05-08 2.000000
 2017-05-09 1.000000
> xts1 - merge(xts2,index(xts1),fill=na.locf) #Subtraction
              e1
 2017-05-04 5.231538
 2017-05-05 5.829257
 2017-05-06 4.829257
 2017-05-07 3.829257
 2017-05-08 2.829257
 2017-05-09 1.829257
```

Reindexing

```
> merge(xts2,xts1,join='inner') #Inner join of xts2 and xts1
             xts2        xts1
 2017-05-05 -0.8382068  10
> merge(xts2,xts1,join='left',fill=0) #Left join of xts2 and xts1, fill empty spots with 0
            xts2        xts1
 2017-05-01 1.7482704   0
 2017-05-02 -0.2314678  0
 2017-05-03 0.1685517   0
 2017-05-04 1.1685649   0
 2017-05-05 -0.8382068  10
> rbind(xts1, xts4) #Combine xts1 and xts4 by rows
```

> Inspect your data

```
> core_data ← coredata(xts2) #Extract core data of objects
> index(xts1) #Extract index of objects
```

Class Attributes

```
> indexClass(xts2) #Get index class
> indexClass(convertIndex(xts,'POSIXct')) #Replacing index class
> indexTZ(xts5) #Get index class
> indexFormat(xts5) ← "%Y-%m-%d" #Change format of time display
```

Time Zones

```
> tzone(xts1) ← "Asia/Hong_Kong" #Change the time zone
> tzone(xts1) #Extract the current time zone
```

Periods, Periodicity and Timestamps

```
> periodicity(xts5) #Estimate frequency of observations
> to.yearly(xts5) #Convert xts5 to yearly OHLC
> to.monthly(xts3) #Convert xts3 to monthly OHLC
> to.quarterly(xts5) #Convert xts5 to quarterly OHLC
> to.period(xts5,period="quarters") #Convert to quarterly OHLC
> to.period(xts5,period="years") #Convert to yearly OHLC
> nmonths(xts5) #Count the months in xts5
> nquarters(xts5) #Count the quarters in xts5
> nyears(xts5) #Count the years in xts5
> make.index.unique(xts3,eps=1e-4) #Make index unique
> make.index.unique(xts3,drop=TRUE) #Remove duplicate times
> align.time(xts3,n=3600) #Round index time to the next n seconds
```

Other Useful Functions

```
> .index(xts4) #Extract raw numeric index of xts1
> .indexwday(xts3) #Value of week(day), starting on Sunday,in index of xts3
> .indexhour(xts3) #Value of hour in index of xts3
> start(xts3) #Extract first observation of xts3
> end(xts4) #Extract last observation of xts4
> str(xts3) #Display structure of xts3
> time(xts1) #Extract raw numeric index of xts1
> head(xts2) #First part of xts2
> tail(xts2) #Last part of xts2
```