

# xts FAQ

xts Development Team

August 28, 2013

## Contents

What is <b>xts</b> ? . . . . .	2
Why should I use <b>xts</b> rather than <b>zoo</b> or another time-series package? . . . . .	2
How do I install <b>xts</b> ? . . . . .	2
I have multiple .csv time-series files that I need to load in a single <b>xts</b> matrix. What is the most efficient way to import the files? . . . . .	2
Why is <b>xts</b> implemented as a matrix rather than a data frame? . . . . .	2
How can I simplify the syntax of my <b>xts</b> matrix column names? . . . . .	2
How can I replace the 0s in an <b>xts</b> object with the last non-zero value in the series? . . . . .	3
How do I create an <b>xts</b> index with millisecond precision? . . . . .	3
OK, so now I have my millisecond series but I still can't see the milliseconds displayed. What went wrong? . . . . .	3
I set <code>digits.sec = 3</code> , but <b>R</b> doesn't show the values correctly. . . . .	4
I am using <code>apply()</code> to run a custom function on my <b>xts</b> series. Why the returned matrix has different dimensions than the original one? . . . . .	4
I have an <b>xts</b> matrix with multiple days of data at various frequencies. For example, day 1 might contain 10 different rows of 1 minute observations, while day 2 contains 20 observations. How can I process all observations for each day and return the summary daily statistics in a new matrix? . . . . .	4
How can I process daily data for a specific time subset? . . . . .	4
How can I process my data in 3-hour blocks, regardless of the begin/end time? I also want to add observations at the beginning and end of each discrete period if missing from the original time-series object. . . . .	5
Why do I get a <b>zoo</b> object when I call <code>transform()</code> on my <b>xts</b> matrix? . . . . .	5
Why can't I use the <code>&amp;</code> operator in <b>xts</b> objects when querying dates? . . . . .	5
How do I subset an <b>xts</b> object to only include weekdays (excluding Saturday and Sundays)? . . . . .	5
I need to quickly convert a data-frame that contains the time-stamps in one of the columns. Using <code>as.xts(q)</code> returns an error. How do I build my <b>xts</b> object? . . . . .	8
I have two time-series with different frequency. I want to combine the data into a single data frame, but the times are not exactly aligned. I want to have one row in the data frame for each ten minute period, with the time index showing the beginning of the time period. . . . .	9

## What is xts?

**xts** is an **R** package offering a number of functionalities to work on time-indexed data. **xts** extends **zoo**, another popular package for time-series analysis.

## Why should I use xts rather than zoo or another time-series package?

The main benefit of **xts** is its seamless compatibility with other packages using different time-series classes (**timeSeries**, **zoo**, ...). In addition **xts** allows the user to add custom attributes to any object. For more information check the **xts** Vignette Introduction.

## How do I install xts?

**xts** depends on **zoo** and some other packages. You should be able to install **xts** and all the other required components by simply calling `install.packages('pkg')` from your **R** prompt.

## I have multiple .csv time-series files that I need to load in a single xts matrix. What is the most efficient way to import the files?

If the files series have the same format, load them with `read.csv()` and then call `rbind()` to join the series together:

```
> filenames <- c("a.csv", "b.csv", "c.csv")
> l <- lapply(filenames, read.csv)
> do.call("rbind", l)
```

## Why is xts implemented as a matrix rather than a data frame?

**xts** uses a matrix rather than `data.frame` because:

1. It is a subclass of **zoo**, and that's how **zoo** objects are structured; and
2. matrix objects have much better performance than `data.frames`.

## How can I simplify the syntax of my xts matrix column names?

`with()` allows to enter the matrix name avoiding the full square brackets syntax. For example:

```
> lm(myxts[, "Res"] ~ myxts[, "ThisVar"] + myxts[, "ThatVar"])
```

can be converted to

```
> with(myxts, lm(Res ~ ThisVar + ThatVar))
```

## How can I replace the 0s in an xts object with the last non-zero value in the series?

Use `na.locf`:

```
> x <- .xts(c(1, 2, 3, 0, 0, 0), 1:6)
> x[x==0] <- NA
> na.locf(x)
```

```
          [,1]
1970-01-01 00:00:01    1
1970-01-01 00:00:02    2
1970-01-01 00:00:03    3
1970-01-01 00:00:04    3
1970-01-01 00:00:05    3
1970-01-01 00:00:06    3
```

```
> x
```

```
          [,1]
1970-01-01 00:00:01    1
1970-01-01 00:00:02    2
1970-01-01 00:00:03    3
1970-01-01 00:00:04   NA
1970-01-01 00:00:05   NA
1970-01-01 00:00:06   NA
```

## How do I create an xts index with millisecond precision?

Milliseconds in **xts** are stored as decimal values. This example builds a series spaced by 100 milliseconds, starting at the current system time:

```
> data(sample_matrix)
> sample.xts = xts(sample_matrix, Sys.time() + seq(0, by = 0.1, length = 180))
```

## OK, so now I have my millisecond series but I still can't see the milliseconds displayed. What went wrong?

Set the `digits.secs` option to some sub-second precision. Continuing from the previous example, if you are interested in milliseconds:

```
> options(digits.secs = 3)
> head(sample.xts)
```

	Open	High	Low	Close
2013-08-28 04:23:48.360	50.03978	50.11778	49.95041	50.11778
2013-08-28 04:23:48.460	50.23050	50.42188	50.23050	50.39767
2013-08-28 04:23:48.560	50.42096	50.42096	50.26414	50.33236
2013-08-28 04:23:48.660	50.37347	50.37347	50.22103	50.33459
2013-08-28 04:23:48.760	50.24433	50.24433	50.11121	50.18112
2013-08-28 04:23:48.860	50.13211	50.21561	49.99185	49.99185

**I set `digits.sec = 3`, but R doesn't show the values correctly.**

Sub-second values are stored in floating point format with microseconds precision. Setting the precision to only 3 decimal hides the full index value in microseconds and might be tricky to interpret depending how the machine rounds the millisecond (3rd) digit. Set the `digits.secs` options to a value higher than 3 or use the `as.numeric()` 'digits' parameter to display the full value. For example:

```
> print(as.numeric(as.POSIXlt("2012-03-20 09:02:50.001")), digits = 20)
[1] 1332234170.0009999275
```

**I am using `apply()` to run a custom function on my `xts` series. Why the returned matrix has different dimensions than the original one?**

When working on rows, `apply()` returns a transposed version of the original matrix. Simply call `t()` on the returned matrix to restore the original dimensions:

```
> myxts.2 <- xts(t(apply(myxts, 1, myfun)), index(myxts))
```

**I have an `xts` matrix with multiple days of data at various frequencies. For example, day 1 might contain 10 different rows of 1 minute observations, while day 2 contains 20 observations. How can I process all observations for each day and return the summary daily statistics in a new matrix?**

First split the source matrix in day subsets, then call `rbind()` to join the processed day statistics together:

```
> do.call(rbind, lapply(split(myxts, "days"), myfun))
```

**How can I process daily data for a specific time subset?**

First extract the time range you want to work on, then apply the daily function:

```
> rt <- r['T16:00/T17:00', 'Value']
> rd <- apply.daily(rt, function(x) xts(t(quantile(x, 0.9)), end(x)))
```

How can I process my data in 3-hour blocks, regardless of the begin/end time? I also want to add observations at the beginning and end of each discrete period if missing from the original time-series object.

Use `align.time()` to set indexes in the periods you are interested in, then call `period.apply` to run your processing function:

```
> # align index into 3-hour blocks
> a <- align.time(s, n=60*60*3)
> # find the number of obs in each block
> count <- period.apply(a, endpoints(a, "hours", 3), length)
> # create an empty \pkg{xts} object with the desired index
> e <- xts(,seq(start(a),end(a),by="3 hours"))
> # merge the counts with the empty object and fill with zeros
> out <- merge(e,count,fill=0)
```

Why do I get a `zoo` object when I call `transform()` on my `xts` matrix?

There's no `xts` method for `transform`, so the `zoo` method is dispatched. The `zoo` method explicitly creates a new `zoo` object. To convert the transformed matrix back to an `xts` object wrap the `transform` call in `as.xts`:

```
> myxts = as.xts(transform(myxts, ABC = 1))
```

You might also have to reset the index timezone:

```
> indexTZ(myxts) = Sys.getenv("TZ")
```

Why can't I use the `&` operator in `xts` objects when querying dates?

"2011-09-21" is not a logical vector and cannot be coerced to a logical vector. See `?"&"` for details.

`xts`' ISO-8601 style subsetting is nice, but there's nothing we can do to change the behavior of `.Primitive("&")`. You can do something like this though:

```
> myts[myts$Symbol == "AAPL" & index(myts) == as.POSIXct("2011-09-21"),]
```

or:

```
> myts[myts$Symbol == "AAPL"]['2011-09-21']
```

How do I subset an `xts` object to only include weekdays (excluding Saturday and Sundays)?

Use `.indexwday()` to only include Mon-Fri days:

```

> data(sample_matrix)
> sample.xts <- as.xts(sample_matrix, descr='my new xts object')
> x <- sample.xts['2007']
> x[.indexwday(x) %in% 1:5]

```

	Open	High	Low	Close
2007-01-02	50.03978	50.11778	49.95041	50.11778
2007-01-03	50.23050	50.42188	50.23050	50.39767
2007-01-04	50.42096	50.42096	50.26414	50.33236
2007-01-05	50.37347	50.37347	50.22103	50.33459
2007-01-08	50.03555	50.10363	49.96971	49.98806
2007-01-09	49.99489	49.99489	49.80454	49.91333
2007-01-10	49.91228	50.13053	49.91228	49.97246
2007-01-11	49.88529	50.23910	49.88529	50.23910
2007-01-12	50.21258	50.35980	50.17176	50.28519
2007-01-15	50.61724	50.68583	50.47359	50.48912
2007-01-16	50.62024	50.73731	50.56627	50.67835
2007-01-17	50.74150	50.77336	50.44932	50.48644
2007-01-18	50.48051	50.60712	50.40269	50.57632
2007-01-19	50.41381	50.55627	50.41278	50.41278
2007-01-22	50.36008	50.43875	50.21129	50.21129
2007-01-23	50.03966	50.16961	50.03670	50.16961
2007-01-24	50.10953	50.26942	50.06387	50.23145
2007-01-25	50.20738	50.28268	50.12913	50.24334
2007-01-26	50.16008	50.16008	49.94052	50.07024
2007-01-29	49.85624	49.93038	49.76308	49.91875
2007-01-30	49.85477	50.02180	49.77242	50.02180
2007-01-31	50.07049	50.22578	50.07049	50.22578
2007-02-01	50.22448	50.41376	50.19101	50.35784
2007-02-02	50.44503	50.53490	50.36064	50.36928
2007-02-05	50.52389	50.69783	50.45977	50.69783
2007-02-06	50.71661	50.71661	50.49865	50.49865
2007-02-07	50.49322	50.69693	50.49322	50.60611
2007-02-08	50.58531	50.84734	50.58531	50.81383
2007-02-09	50.83331	50.89683	50.67686	50.67686
2007-02-12	50.88990	50.96653	50.83604	50.96653
2007-02-13	50.90056	51.00299	50.87935	50.90106
2007-02-14	50.95283	51.04699	50.80317	51.04699
2007-02-15	51.06330	51.11401	50.94681	51.05185
2007-02-16	51.12879	51.12879	51.00613	51.02164
2007-02-19	51.29502	51.32342	51.13524	51.17899
2007-02-20	51.13725	51.14940	50.93523	50.93523
2007-02-21	50.92940	50.92940	50.69880	50.77325
2007-02-22	50.72111	50.86597	50.65718	50.86597
2007-02-23	50.84392	50.96946	50.73060	50.76498
2007-02-26	50.88168	50.88168	50.75481	50.75481
2007-02-27	50.74333	50.78909	50.61874	50.69206
2007-02-28	50.69435	50.77091	50.59881	50.77091
2007-03-01	50.81620	50.81620	50.56451	50.57075

2007-03-02 50.60980 50.72061 50.50808 50.61559  
 2007-03-05 50.26501 50.34050 50.26501 50.29567  
 2007-03-06 50.27464 50.32019 50.16380 50.16380  
 2007-03-07 50.14458 50.20278 49.91381 49.91381  
 2007-03-08 49.93149 50.00364 49.84893 49.91839  
 2007-03-09 49.92377 49.92377 49.74242 49.80712  
 2007-03-12 49.82763 49.90311 49.67049 49.74033  
 2007-03-13 49.69628 49.70863 49.37924 49.37924  
 2007-03-14 49.36270 49.53735 49.30746 49.53735  
 2007-03-15 49.57374 49.62310 49.39876 49.49600  
 2007-03-16 49.44900 49.65285 49.42416 49.59500  
 2007-03-19 49.62747 49.65407 49.51604 49.54590  
 2007-03-20 49.59529 49.62003 49.42321 49.50690  
 2007-03-21 49.49765 49.53961 49.41610 49.51807  
 2007-03-22 49.42306 49.42306 49.31184 49.39687  
 2007-03-23 49.27281 49.27281 48.93095 48.93095  
 2007-03-26 48.34210 48.44637 48.28969 48.28969  
 2007-03-27 48.25248 48.41572 48.23648 48.30851  
 2007-03-28 48.33090 48.53595 48.33090 48.53595  
 2007-03-29 48.59236 48.69988 48.57432 48.69988  
 2007-03-30 48.74562 49.00218 48.74562 48.93546  
 2007-04-02 48.90488 49.08400 48.90488 49.06316  
 2007-04-03 49.06071 49.24525 48.96928 49.24525  
 2007-04-04 49.22579 49.37335 49.19913 49.34736  
 2007-04-05 49.41435 49.41435 49.30641 49.33776  
 2007-04-06 49.33621 49.41900 49.33621 49.41900  
 2007-04-09 49.44429 49.50234 49.33828 49.50234  
 2007-04-10 49.55704 49.78776 49.55704 49.76984  
 2007-04-11 49.74550 49.81925 49.74550 49.74623  
 2007-04-12 49.75079 49.75470 49.61732 49.72996  
 2007-04-13 49.70708 49.85332 49.69245 49.73339  
 2007-04-16 49.74915 49.86289 49.71091 49.83886  
 2007-04-17 49.84698 49.95456 49.77754 49.95456  
 2007-04-18 49.93794 50.07208 49.92484 50.07208  
 2007-04-19 50.02441 50.02991 49.83945 49.83945  
 2007-04-20 49.76042 49.92847 49.69808 49.91103  
 2007-04-23 50.32009 50.32009 49.87574 49.88539  
 2007-04-24 49.87340 49.90184 49.72769 49.72769  
 2007-04-25 49.73385 49.88622 49.73385 49.88472  
 2007-04-26 49.89064 49.89064 49.74899 49.79201  
 2007-04-27 49.80530 49.80530 49.50814 49.50814  
 2007-04-30 49.13825 49.33974 49.11500 49.33974  
 2007-05-01 49.34572 49.52635 49.34572 49.47138  
 2007-05-02 49.47062 49.47062 49.34261 49.38521  
 2007-05-03 49.46328 49.69097 49.46328 49.58677  
 2007-05-04 49.59963 49.59963 49.41375 49.41375  
 2007-05-07 49.49188 49.49188 49.13572 49.13572  
 2007-05-08 49.13282 49.25507 49.13282 49.18930

```

2007-05-09 49.17739 49.17739 48.72708 48.72708
2007-05-10 48.83479 48.84549 48.38001 48.38001
2007-05-11 48.25456 48.25456 47.96904 47.96904
2007-05-14 47.64469 47.72505 47.58212 47.65930
2007-05-15 47.60647 47.74053 47.51796 47.72686
2007-05-16 47.72065 47.90717 47.70913 47.86683
2007-05-17 47.79430 47.79430 47.55140 47.62938
2007-05-18 47.65013 47.75117 47.65013 47.68423
2007-05-21 47.96582 48.02903 47.78072 47.78072
2007-05-22 47.81830 47.94825 47.81155 47.82946
2007-05-23 47.93593 48.08242 47.88763 47.90068
2007-05-24 47.89041 48.03077 47.88413 48.01130
2007-05-25 47.98234 48.17543 47.94507 48.16058
2007-05-28 47.90142 47.93398 47.64718 47.64718
2007-05-29 47.65665 47.89342 47.65446 47.87252
2007-05-30 47.78866 47.93267 47.78866 47.83291
2007-05-31 47.82845 47.84044 47.73780 47.73780
2007-06-01 47.74432 47.74432 47.54820 47.65123
2007-06-04 47.51516 47.53545 47.32342 47.37642
2007-06-05 47.41090 47.48217 47.21116 47.22930
2007-06-06 47.36581 47.41233 47.23306 47.40048
2007-06-07 47.42099 47.50637 47.35320 47.45262
2007-06-08 47.48449 47.53089 47.42814 47.48360
2007-06-11 47.27807 47.30884 47.14660 47.14660
2007-06-12 47.19411 47.41834 47.18153 47.41834
2007-06-13 47.46135 47.52004 47.43083 47.43083
2007-06-14 47.43279 47.43279 47.33490 47.34884
2007-06-15 47.33306 47.40490 47.26157 47.36779
2007-06-18 47.43470 47.56336 47.36424 47.36424
2007-06-19 47.46055 47.73353 47.46055 47.67220
2007-06-20 47.71126 47.81759 47.66843 47.66843
2007-06-21 47.71012 47.71012 47.61106 47.62921
2007-06-22 47.56849 47.59266 47.32549 47.32549
2007-06-25 47.20471 47.42772 47.13405 47.42772
2007-06-26 47.44300 47.61611 47.44300 47.61611
2007-06-27 47.62323 47.71673 47.60015 47.62769
2007-06-28 47.67604 47.70460 47.57241 47.60716
2007-06-29 47.63629 47.77563 47.61733 47.66471

```

**I need to quickly convert a data-frame that contains the time-stamps in one of the columns. Using `as.xts(q)` returns an error. How do I build my xts object?**

The `xts()` constructor requires two arguments: a vector or a matrix carrying data and a vector of type `Date`, `POSIXt`, `chron`, ... supplying the time index information. If the time is set in one of the matrix columns, use this line:

```
> qxts = xts(q[,-1], order.by=q[,1])
```



I have two time-series with different frequency. I want to combine the data into a single data frame, but the times are not exactly aligned. I want to have one row in the data frame for each ten minute period, with the time index showing the beginning of the time period.

`align.time()` creates evenly spaced time-series from a set of indexes, `merge()` insure two time-series are combined in a single `xts` object with all original columns and indexes preserved. The new object has one entry for each timestamp from both series and values missing are replaced with NAs.

```
> xTemps <- align.time(xts(temps[,2],as.POSIXct(temps[,1])), n=600)
> xGas <- align.time(xts(gas[,2],as.POSIXct(gas[,1])), n=600)
> merge(xTemps,xGas)
```