Returns calculations in R

Bernt Arne Ødegaard

1 Introduction

In finance, one of the standard task is to use data on *prices* **P** of financial assets to calculate *returns* **R**.

$$R_{it} = \frac{P_{i,t} - P_{i,t-1}}{P_{i,t-1}}$$

Typically one want to calculate the returns at fixed frequencies, such as monthly. The recommended procedure is as follows:

1. Make sure the price series is a time series object, such as a `zoo` or `xts` series.
2. Use the return calculations in the `quantmod` package to get returns at the desired frequency, such as daily, monthly, quarterly, yearly (the most common cases).

2 Example

Consider the following price sequence of annual observations.

$$\mathbf{P} = \{1, 2, 4\}$$

The returns here are

$$\mathbf{R} = \{1, 1\} = \{100\%, 100\%\}$$

or, each period involves a 100% return.

To illustrate the calculation in R, first create the time series (this example uses the `zoo` time series package.

```
> library(zoo)
> dates <- c(as.Date("2001-12-31"),
             as.Date("2002-12-31"),
             as.Date("2003-12-31"))
> prices <- zoo(c(1,2,4),order.by=dates)
> prices
2001-12-31 2002-12-31 2003-12-31
         1          2          4
```

Calculate annual returns using the `quantmod` calculation of annual returns:

```
> library(quantmod)
> returns <- annualReturn(prices,leading=FALSE)
> returns
           yearly.returns
2002-12-31              1
2003-12-31              1
```

3 Example - Ford's annual returns

As a more realworld example, download the daily stock price history of Ford (ticker `F`) from yahoo finance, and calculate annual returns for Ford in the period 2010-2019.

Getting data using the `getSymbols` routine:

```
> ford <- getSymbols("F",
+                    source="yahoo",
+                    auto.assign=FALSE,
+                    from=as.Date("2010-01-01"))

> head(ford)
           F.Open F.High F.Low F.Close  F.Volume F.Adjusted
2010-01-04  10.17  10.28 10.05   10.28  60855800   7.238358
2010-01-05  10.45  11.24 10.40   10.96 215620200   7.717158
2010-01-06  11.21  11.46 11.13   11.37 200070600   8.005848
2010-01-07  11.46  11.69 11.32   11.66 130201700   8.210042
2010-01-08  11.67  11.74 11.46   11.69 130463000   8.231167
2010-01-11  11.90  12.14 11.78   12.11 170626200   8.526896
```

Note that the the prices used for calculating returns are the prices adjusted for dividends, stock splits, etc, in the column `F.Adjusted`:

```
> ford_monthly_returns <- monthlyReturn(ford$F.Adjusted)
> ford_annual_returns <- annualReturn(ford$F.Adjusted)
> print(ford_annual_returns)
           yearly.returns
2010-12-31     0.63326807
2011-12-30    -0.35914251
2012-12-31     0.22600501
2013-12-31     0.22316542
2014-12-31     0.03696820
2015-12-31    -0.05403286
2016-12-30    -0.09806912
2017-12-29     0.08296394
2018-12-31    -0.35202864
2019-12-31     0.29639731
```

4 What can go wrong

The reason why the above is the recommended procedure, and not a more simple approach involving first differences, is that while the returns calculation seem like a trivial exercise, it is very easy to "shoot yourself in the foot" with simpler shortcuts.

A procedure you would think produces returns is to take first differences and divide by all but the last price. That may work, but only if you are very careful. Let us look at the same simple example.

4.1 the basic `ts` object

Base `R` contains a time series object `ts` where you specify a frequency together with the observations. For example, `frequency=1` is annual

```
> prices <- ts(c(1,2,4),frequency=1)
> returns <- diff(prices)/prices[-length(prices)]
```

```
> returns
Time Series:
Start = 2
End = 3
Frequency = 1
[1] 1 1
```

So in this case the calculation of dividing the differences by all but the last element of the price vector works.

4.2 the zoo library

But does it work when we use the zoo library?

```
> library(zoo)
> dates <- c(as.Date("2001-12-31"),
             as.Date("2002-12-31"),
             as.Date("2003-12-31"))
> prices <- zoo(c(1,2,4),order.by=dates)
> prices
2001-12-31 2002-12-31 2003-12-31
         1          2          4
> returns <- diff(prices)/prices[-length(prices)]
> returns
2002-12-31
      0.5
```

Why does this not work?

Check what we are actually dividing here

```
> print(diff(prices))
2002-12-31 2003-12-31
         1          2
> print(prices[-length(prices)])
2001-12-31 2002-12-31
         1          2
```

So, zoo silently matches date when doing the above calculation, which is not what was desired.

If we want to keep using zoo, but still require returns, one way to fix this is to remove the dates from the divisor:

```
> returns <- diff(prices)/as.vector(prices[-length(prices)])
> returns
2002-12-31 2003-12-31
         1          1
```

4.3 The xts library

Let us also check what xts does:

```
> prices <- xts(c(1,2,4),order.by=dates)
> prices
           [,1]
2001-12-31    1
2002-12-31    2
```

```
2003-12-31    4
> returns <- diff(prices)/prices[-length(prices)]
> returns
            e1
2001-12-31  NA
2002-12-31 0.5
```

which is also very wrong.

```
> print(diff(prices))
           [,1]
2001-12-31   NA
2002-12-31    1
2003-12-31    2
> print(prices[-length(prices)])
           [,1]
2001-12-31    1
2002-12-31    2
```

Here, the trick of replacing the prices does not work either

```
> returns <- diff(prices)/as.vector(prices[-length(prices)])
Warning message:
In `/.default`(diff(prices), as.vector(prices[-length(prices)])) :
  longer object length is not a multiple of shorter object length
> returns
           [,1]
2001-12-31   NA
2002-12-31  0.5
2003-12-31  2.0
```

But we can get the correct answer as follows

```
> returns <- na.omit(diff(prices))/coredata(prices[-length(prices)])
> returns
           [,1]
2002-12-31    1
2003-12-31    1
```

Well, it seems like getting returns are actually less trivial than one should think, so just get used to the `quantmod` package.