Special aggregation for growth timeseries in package regts

Rob van Harrevelt and Anita van der Roest

2019-09-11

Contents

1 Introduction

This vignette is an extension of the `regts` vignette but can be read independent.

First a general introduction to aggregation in package `regts` is given. Then a special set of aggregation functions is introduced for timeseries representing growth rates, the so called cumulative growth functions.

2 Aggregation

Aggregation of timeseries is the conversion of a timeseries to a lower frequency. For example to convert a monthly timeseries to a quarterly timeseries, or a quarterly timeseries to a yearly timeseries.

In package `regts` a particular function `aggregate` is available to perform an aggregation. An example with a monthly timeseries:

```
> regtm1 <- regts(1:24, start = "2016M1")
> regtm1
```

```
     Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
2016   1   2   3   4   5   6   7   8   9  10  11  12
2017  13  14  15  16  17  18  19  20  21  22  23  24
```

```
> aggregate(regtm1, FUN = mean, nfrequency = 4)
```

```
     Qtr1 Qtr2 Qtr3 Qtr4
2016    2    5    8   11
2017   14   17   20   23
```

And another with a quarterly timeseries:

```
> regt1 <- regts(1:10, start = "2016q1")
> regt1
```

```
     Qtr1 Qtr2 Qtr3 Qtr4
2016    1    2    3    4
2017    5    6    7    8
2018    9   10
```

```
> aggregate(regt1, FUN = sum)
```

The result is a yearly timeseries:

```
Time Series: Start = 2016  End = 2017  Frequency = 1  [1] 10 26
```

The quarters in 2016 add up to 10, the quarters in 2017 to 26. Note that the 2 quarters in 2018 are ignored. The `aggregate` function skips the incomplete years at the end.

A special situation occurs when for instance a quarterly timeseries doesn't start in the first quarter:

```
> regt2 <- regts(2:10, start = "2016q2")
> regt2
```

```
     Qtr1 Qtr2 Qtr3 Qtr4
2016         2    3    4
2017    5    6    7    8
2018    9   10
```

```
> aggregate(regt2, FUN = sum)
```

Now the result is:

```
Time Series: Start = 2017  End = 2017  Frequency = 1  [1] 26
```

The quarters in 2017 add up to 26 as before. But now the information for 2016 is also not complete, and therefore ignored. This is what you would expect: for `regts` objects the `aggregate` function skips all years for which not all quarters are present.

Usually the `aggregate` function works the same for `regts` and `ts` objects, but in this case the result is very different. The same example with a `ts` timeseries:

```
> t2    <- ts(2:10, start = c(2016,2), frequency = 4)
> aggregate(t2, FUN = sum)
```

The result is:

```
Time Series: Start = 2016.25  End = 2017.25  Frequency = 1  [1] 14 30
```

Now the first year (2016) is not ignored. This leads to a shifted result period (the start is `2016.25`) and shifted input ($14 = 2 + 3 + 4 + 5$), which makes the results peculiar.[1]

So the function `aggregate` operates differently for a `regts` or a `ts` object, if the first period does not start at a subperiod for the new frequency.

3 Aggregation for growth timeseries

Another case is the treatment of timeseries that contain absolute, relative or percentage changes. For this case the function `aggregate_gr` is developed with special aggregation algorithms, the so called 'cumulative growth'-methods. A more detailed description of these methods can be found in the next subsection. This function can be employed for both `regts` and `ts` timeseries.

All methods convert input timeseries with high frequency to low frequency outputseries. There are four different type of methods for different types of input timeseries: `dif1`, `dif1s`, `rel` and `pct`.

The `dif1s` and `dif1` methods assume that the input timeseries is a first difference of length 1 in the input frequency (for `dif1s` the input is also scaled). They calculate a first difference of length 1 in the output frequency.

The `rel` and `pct` methods assume that the input timeseries is a one-period relative or percentage change and calculate the exact relative or percentage change for the output timeseries.

These methods are described in detail in the next section.

First a timeseries is created containing differences:

```
> xdif <- diff(regts(c(1,3,4,3,6,2,4,1,3,2), start = "2015q4"))
> xdif
```

```
     Qtr1 Qtr2 Qtr3 Qtr4
2016    2    1   -1    3
2017   -4    2   -3    2
2018   -1
```

Two examples of aggregation methods are shown, the results are yearly timeseries:

[1] This problem does not occur when a quarterly timeseries ends before the last quarter, or before the end of the quarter in case of a monthly timeseries. Then the last year or quarter is ignored.

```
> aggregate_gr(xdif, method = "dif1s", nfrequency = 1)  # result is also scaled
```

```
2017 -1.5
```

```
> aggregate_gr(xdif, method = "dif1")
```

```
2017 -6
```

4 The cumulative growth methods

The so called growth timeseries require special methods for aggregation, a simple averaging of the subperiods in the high frequency observations will not return the correct answer. In this section a more detailed description of the four previously defined frequency conversion methods is given.

In advance some definitions are needed:

We are temporally aggregating a timeseries x with n subperiods to a timeseries X with a lower frequency. In other words we want to convert a timeseries x with a high frequency (monthly, quarterly) to a new timeseries with a lower frequency (quarterly, annual).

Let $x_{t,i}$ stand for the value of x in subperiod i of main period t. The index i takes on values in the range $1..n$. We always interpret $x_{t,0}$ to mean $x_{t-1,n}$. The time index t refers to the periods in the time domain of X. Many conversion methods only need the observations subperiod i of main period t. For example the `mean` method calculates X_t as follows

$$X_t = \sum_{i=1}^{i=n} x_{t,i}/n$$

`dif1s` method

Define x to be the first difference of z scaled to the output frequency. Thus

$$x_{t,i} = n(z_{t,i} - z_{t,i-1})$$

We can also define $z_{t,i}$ in terms of x as follows

$$\begin{aligned} z_{t,i} &= z_{t,i-1} + x_{t,i}/n \\ &= z_{t-1,n} + \sum_{j=1}^{i} x_{t,j}/n \end{aligned}$$

Now define Z_t as the level timeseries in time domain t corresponding to z

$$\begin{aligned} Z_t &= \sum_{i=1}^{n} z_{t,i} \\ &= \sum_{i=1}^{n} z_{t-1,n} + \sum_{i=1}^{n}\sum_{j=1}^{i} x_{t,j}/n \\ &= \sum_{i=1}^{n} z_{t-2,n} + \sum_{i=1}^{n}\sum_{j=1}^{n} x_{t-1,j}/n + \sum_{i=1}^{n}\sum_{j=1}^{i} x_{t,j}/n \end{aligned}$$

Then X in any period t can be calculated from

$$X_t = Z_t - Z_{t-1}$$

Since Z_{t-1} can be written as

$$Z_{t-1} = \sum_{i=1}^{n} z_{t-2,n} + \sum_{i=1}^{n}\sum_{j=1}^{i} x_{t-1,j}/n$$

it is easy to show that X_t does not depend on $z_{t-2,n}$. In any practical algorithm $z_{t-2,n}$ can be set to 0.

A compact equation for X_t can be derived by substituting the equations for Z_t and Z_{t-1} into the definition of X_t

$$X_t = \sum_{i=1}^{n} \left(\sum_{j=i+1}^{n} x_{t-1,j} + \sum_{j=1}^{i} x_{t,j} \right) /n$$

This expression can be futher simplified by changing the order of summation. For the first summation, we can write

$$\sum_{i=1}^{n}\sum_{j=i+1}^{n} x_{t-1,j} = \sum_{j=2}^{n}\sum_{i=1}^{j-1} x_{t-1,j} = \sum_{j=2}^{n}(j-1)x_{t-1,j}$$

Similarly,

$$\sum_{i=1}^{n}\sum_{j=1}^{i} x_{t,j} = \sum_{j=1}^{n}\sum_{i=j}^{n} x_{t,j} = \sum_{j=1}^{n}(n-j+1)x_{t,j}$$

The final equation is given by

$$X_t = \left(\sum_{j=2}^{n}(j-1)x_{t-1,j} + \sum_{j=1}^{n}(n-j+1)x_{t,j} \right) /n$$

`dif1` method

Define x to be the first difference of z. Thus

$$x_{t,i} = z_{t,i} - z_{t,i-1}$$

We can also define $z_{t,i}$ in terms of x as follows

$$\begin{aligned} z_{t,i} &= z_{t,i-1} + x_{t,i} \\ &= z_{t-1,n} + \sum_{j=1}^{i} x_{t,j} \end{aligned}$$

Further derivations are analogous to the case for the `dif1s` method described in the previous section.

`rel` method

Define x to be the relative change in z. Thus

$$x_{t,i} = (z_{t,i} - z_{t,i-1})/z_{t,i-1}$$

We can also define $z_{t,i}$ in terms of x as follows

$$\begin{aligned} z_{t,i} &= z_{t,i-1}(1 + x_{t,i}) \\ &= z_{t-1,n} \prod_{j=1}^{i}(1 + x_{t,j}) \\ &= z_{t-2,n} \prod_{j=1}^{n}(1 + x_{t-1,j}) \prod_{j=1}^{i}(1 + x_{t,j}) \end{aligned}$$

Now define Z_t as the level timeseries in time domain t corresponding to z

$$\begin{aligned} Z_t &= \sum_{i=1}^{n} z_{t,i} \\ &= z_{t-1,n} \sum_{i=1}^{n} \prod_{j=1}^{i} (1 + x_{t,j}) \\ &= z_{t-2,n} \prod_{j=1}^{n} (1 + x_{t-1,j}) \sum_{i=1}^{n} \prod_{j=1}^{i} (1 + x_{t,j}) \end{aligned}$$

Then X in any period t can be calculated from

$$X_t = Z_t / Z_{t-1} - 1$$

Using the expression for Z_{t-1}

$$Z_{t-1} = z_{t-2,n} \sum_{i=1}^{n} \prod_{j=1}^{i} (1 + x_{t-1,j})$$

we obtain

$$X_t = \frac{\prod_{j=2}^{n} (1 + x_{t-1,j}) \sum_{i=1}^{n} \prod_{j=1}^{i} (1 + x_{t,j})}{1 + \sum_{i=2}^{n} \prod_{j=2}^{i} (1 + x_{t-1,j})} - 1$$

`pct` method

Define x to be the percentage change in z. Thus

$$x_{t,i} = 100(z_{t,i} - z_{t,i-1}) / z_{t,i-1}$$

We can also define $z_{t,i}$ in terms of x as follows

$$\begin{aligned} z_{t,i} &= z_{t,i-1}(1 + 0.01 x_{t,i}) \\ &= z_{t-1,n} \prod_{j=1}^{i} (1 + 0.01 x_{t,j}) \end{aligned}$$

Now define Z_t as the level timeseries in time domain t corresponding to z

$$Z_t = \sum_{i=1}^{n} z_{t,i}$$

Then X in any period t can be calculated from

$$X_t = 100(Z_t / Z_{t-1} - 1)$$

Further derivations are analogous to the case for the `dif1s` method described in the previous section.