Draft guide to package "lagged"

Georgi N. Boshnakov

Abstract

Package **lagged** provides classes and methods for objects, whose indexing naturally starts from zero. Subsetting, indexing and mathematical operations are defined naturally between lagged objects and lagged and base R objects. Recycling is not used, except for singletons. The single bracket operator doesn't drop dimensions by default.

This vignette is part of package **lagged**, version 0.3-1.

Keywords: lag, autocorrelation, indexing.

1. Univariate lagged objects

Create a univariate lagged object[1]:

```
> a1 <-  drop(acf(ldeaths)$acf)
> la1 <- Lagged(a1)
> la1
```

```
An object of class "Lagged1d"
Slot *data*:
       Lag_0        Lag_1        Lag_2        Lag_3        Lag_4        Lag_5
 1.000000000  0.755051141  0.396956836  0.019395714 -0.355897989 -0.608566374
       Lag_6        Lag_7        Lag_8        Lag_9       Lag_10       Lag_11
-0.681383469 -0.607909875 -0.378212377 -0.012975866  0.383252644  0.650206704
      Lag_12       Lag_13       Lag_14       Lag_15       Lag_16       Lag_17
 0.723167071  0.638001465  0.371577811  0.009467461 -0.293699737 -0.496742216
      Lag_18
-0.585558984
```

`maxLag()` returns the maximal lag in the object. `Length()` returns the number of lags in the object, i.e. `length(la1) == maxLag(la1) + 1`. This relation is a definition and holds also for multivariate lagged objects. In particular, the length is not necessarily the length of the data slot.

```
> maxLag(la1)
```

```
[1] 18
```

```
> length(la1)
```

```
[1] 19
```

[1] The datasets `ldeaths`, `fdeaths` and `mdeaths` are in base R. The examples involving them are adapted from the help page of `acf()`.

2. Indexing

Indexing drops the "laggedness" to allow easy access to the underlying data[2]:

```
> la1[0]

[1] 1

> la1[0:4]

[1]  1.00000000  0.75505114  0.39695684  0.01939571 -0.35589799

> la1[c(1,3,5)]

[1]  0.75505114  0.01939571 -0.60856637

> la1[]

 [1]  1.000000000  0.755051141  0.396956836  0.019395714 -0.355897989
 [6] -0.608566374 -0.681383469 -0.607909875 -0.378212377 -0.012975866
[11]  0.383252644  0.650206704  0.723167071  0.638001465  0.371577811
[16]  0.009467461 -0.293699737 -0.496742216 -0.585558984

> la1a <- la1
> la1a[] <- round(la1, 2)
> la1a

An object of class "Lagged1d"
Slot *data*:
 Lag_0  Lag_1  Lag_2  Lag_3  Lag_4  Lag_5  Lag_6  Lag_7  Lag_8  Lag_9 Lag_10
  1.00   0.76   0.40   0.02  -0.36  -0.61  -0.68  -0.61  -0.38  -0.01   0.38
Lag_11 Lag_12 Lag_13 Lag_14 Lag_15 Lag_16 Lag_17 Lag_18
  0.65   0.72   0.64   0.37   0.01  -0.29  -0.50  -0.59

> la1b <- round(la1, 2)
> all(la1a == la1b)

[1] TRUE
```

3. Unary arithmetic and mathematical functions

Unary arithmetic operations and mathematical functions replace the data part of the object and keep its class.

```
> -la1a
```

[2] For some indices, such as `0:4`, it is possible to keep a Lagged class but it would be confusing if the indexing operation was returning Lagged or non-Lagged objects depending on the values of the index.

```
An object of class "Lagged1d"
Slot *data*:
 Lag_0  Lag_1  Lag_2  Lag_3  Lag_4  Lag_5  Lag_6  Lag_7  Lag_8  Lag_9 Lag_10
 -1.00  -0.76  -0.40  -0.02   0.36   0.61   0.68   0.61   0.38   0.01  -0.38
Lag_11 Lag_12 Lag_13 Lag_14 Lag_15 Lag_16 Lag_17 Lag_18
 -0.65  -0.72  -0.64  -0.37  -0.01   0.29   0.50   0.59

> +la1a

An object of class "Lagged1d"
Slot *data*:
 Lag_0  Lag_1  Lag_2  Lag_3  Lag_4  Lag_5  Lag_6  Lag_7  Lag_8  Lag_9 Lag_10
  1.00   0.76   0.40   0.02  -0.36  -0.61  -0.68  -0.61  -0.38  -0.01   0.38
Lag_11 Lag_12 Lag_13 Lag_14 Lag_15 Lag_16 Lag_17 Lag_18
  0.65   0.72   0.64   0.37   0.01  -0.29  -0.50  -0.59

> ## Math group
> abs(la1a)

An object of class "Lagged1d"
Slot *data*:
 Lag_0  Lag_1  Lag_2  Lag_3  Lag_4  Lag_5  Lag_6  Lag_7  Lag_8  Lag_9 Lag_10
  1.00   0.76   0.40   0.02   0.36   0.61   0.68   0.61   0.38   0.01   0.38
Lag_11 Lag_12 Lag_13 Lag_14 Lag_15 Lag_16 Lag_17 Lag_18
  0.65   0.72   0.64   0.37   0.01   0.29   0.50   0.59

> sinpi(la1a)

An object of class "Lagged1d"
Slot *data*:
      Lag_0       Lag_1       Lag_2       Lag_3       Lag_4       Lag_5
 0.00000000  0.68454711  0.95105652  0.06279052 -0.90482705 -0.94088077
      Lag_6       Lag_7       Lag_8       Lag_9      Lag_10      Lag_11
-0.84432793 -0.94088077 -0.92977649 -0.03141076  0.92977649  0.89100652
     Lag_12      Lag_13      Lag_14      Lag_15      Lag_16      Lag_17
 0.77051324  0.90482705  0.91775463  0.03141076 -0.79015501 -1.00000000
     Lag_18
-0.96029369

> sqrt(abs(la1a))

An object of class "Lagged1d"
Slot *data*:
    Lag_0     Lag_1     Lag_2     Lag_3     Lag_4     Lag_5     Lag_6     Lag_7
1.0000000 0.8717798 0.6324555 0.1414214 0.6000000 0.7810250 0.8246211 0.7810250
    Lag_8     Lag_9    Lag_10    Lag_11    Lag_12    Lag_13    Lag_14    Lag_15
0.6164414 0.1000000 0.6164414 0.8062258 0.8485281 0.8000000 0.6082763 0.1000000
   Lag_16    Lag_17    Lag_18
0.5385165 0.7071068 0.7681146

> ## Math2 group
> round(la1a)
```

```
An object of class "Lagged1d"
Slot *data*:
 Lag_0  Lag_1  Lag_2  Lag_3  Lag_4  Lag_5  Lag_6  Lag_7  Lag_8  Lag_9 Lag_10
     1      1      0      0      0     -1     -1     -1      0      0      0
Lag_11 Lag_12 Lag_13 Lag_14 Lag_15 Lag_16 Lag_17 Lag_18
     1      1      1      0      0      0      0     -1
```

```
> round(la1a, 2)
```

```
An object of class "Lagged1d"
Slot *data*:
 Lag_0  Lag_1  Lag_2  Lag_3  Lag_4  Lag_5  Lag_6  Lag_7  Lag_8  Lag_9 Lag_10
  1.00   0.76   0.40   0.02  -0.36  -0.61  -0.68  -0.61  -0.38  -0.01   0.38
Lag_11 Lag_12 Lag_13 Lag_14 Lag_15 Lag_16 Lag_17 Lag_18
  0.65   0.72   0.64   0.37   0.01  -0.29  -0.50  -0.59
```

```
> signif(la1a)
```

```
An object of class "Lagged1d"
Slot *data*:
 Lag_0  Lag_1  Lag_2  Lag_3  Lag_4  Lag_5  Lag_6  Lag_7  Lag_8  Lag_9 Lag_10
  1.00   0.76   0.40   0.02  -0.36  -0.61  -0.68  -0.61  -0.38  -0.01   0.38
Lag_11 Lag_12 Lag_13 Lag_14 Lag_15 Lag_16 Lag_17 Lag_18
  0.65   0.72   0.64   0.37   0.01  -0.29  -0.50  -0.59
```

```
> signif(la1a, 4)
```

```
An object of class "Lagged1d"
Slot *data*:
 Lag_0  Lag_1  Lag_2  Lag_3  Lag_4  Lag_5  Lag_6  Lag_7  Lag_8  Lag_9 Lag_10
  1.00   0.76   0.40   0.02  -0.36  -0.61  -0.68  -0.61  -0.38  -0.01   0.38
Lag_11 Lag_12 Lag_13 Lag_14 Lag_15 Lag_16 Lag_17 Lag_18
  0.65   0.72   0.64   0.37   0.01  -0.29  -0.50  -0.59
```

The functions from the summary group work on the data part, as if they were called on it.

```
> c(Max = max(la1a), Min = min(la1a), Range = range(la1a))
```

```
   Max    Min Range1 Range2
  1.00  -0.68  -0.68   1.00
```

```
> c(Prod = prod(la1a), Sum = sum(la1a))
```

```
         Prod           Sum
-7.582098e-11  9.200000e-01
```

```
> c(Any = any(la1a < 0), All = all(la1a >= 0))
```

```
  Any   All
 TRUE FALSE
```

Binary arithmetic operators are defined between two lagged objects and between a lagged object and a vector. They return a lagged object from one of the "basic" lagged classes, but not necessarilly exactly from the class of the argument(s). The class of the returned value is from a suitable lagged superclass of the argument(s). This concerns operations on objects from classes inheriting from the classes considered here, so is not visible in the examples below, since they use objects from the basic lagged classes.

```
> 2*la1a

An object of class "Lagged1d"
Slot *data*:
 Lag_0  Lag_1  Lag_2  Lag_3  Lag_4  Lag_5  Lag_6  Lag_7  Lag_8  Lag_9 Lag_10
  2.00   1.52   0.80   0.04  -0.72  -1.22  -1.36  -1.22  -0.76  -0.02   0.76
Lag_11 Lag_12 Lag_13 Lag_14 Lag_15 Lag_16 Lag_17 Lag_18
  1.30   1.44   1.28   0.74   0.02  -0.58  -1.00  -1.18

> la1a^2

An object of class "Lagged1d"
Slot *data*:
 Lag_0  Lag_1  Lag_2  Lag_3  Lag_4  Lag_5  Lag_6  Lag_7  Lag_8  Lag_9 Lag_10
1.0000 0.5776 0.1600 0.0004 0.1296 0.3721 0.4624 0.3721 0.1444 0.0001 0.1444
Lag_11 Lag_12 Lag_13 Lag_14 Lag_15 Lag_16 Lag_17 Lag_18
0.4225 0.5184 0.4096 0.1369 0.0001 0.0841 0.2500 0.3481

> la1a + la1a^2

An object of class "Lagged1d"
Slot *data*:
  Lag_0   Lag_1   Lag_2   Lag_3   Lag_4   Lag_5   Lag_6   Lag_7   Lag_8   Lag_9
 2.0000  1.3376  0.5600  0.0204 -0.2304 -0.2379 -0.2176 -0.2379 -0.2356 -0.0099
 Lag_10  Lag_11  Lag_12  Lag_13  Lag_14  Lag_15  Lag_16  Lag_17  Lag_18
 0.5244  1.0725  1.2384  1.0496  0.5069  0.0101 -0.2059 -0.2500 -0.2419

> la1a - la1a^2

An object of class "Lagged1d"
Slot *data*:
  Lag_0   Lag_1   Lag_2   Lag_3   Lag_4   Lag_5   Lag_6   Lag_7   Lag_8   Lag_9
 0.0000  0.1824  0.2400  0.0196 -0.4896 -0.9821 -1.1424 -0.9821 -0.5244 -0.0101
 Lag_10  Lag_11  Lag_12  Lag_13  Lag_14  Lag_15  Lag_16  Lag_17  Lag_18
 0.2356  0.2275  0.2016  0.2304  0.2331  0.0099 -0.3741 -0.7500 -0.9381

> la1a * la1a^2

An object of class "Lagged1d"
Slot *data*:
    Lag_0     Lag_1     Lag_2     Lag_3     Lag_4     Lag_5     Lag_6     Lag_7
 1.000000  0.438976  0.064000  0.000008 -0.046656 -0.226981 -0.314432 -0.226981
    Lag_8     Lag_9    Lag_10    Lag_11    Lag_12    Lag_13    Lag_14    Lag_15
-0.054872 -0.000001  0.054872  0.274625  0.373248  0.262144  0.050653  0.000001
   Lag_16    Lag_17    Lag_18
-0.024389 -0.125000 -0.205379
```

```
> la1a / la1a^2

An object of class "Lagged1d"
Slot *data*:
      Lag_0       Lag_1       Lag_2       Lag_3       Lag_4       Lag_5
   1.000000    1.315789    2.500000   50.000000   -2.777778   -1.639344
      Lag_6       Lag_7       Lag_8       Lag_9      Lag_10      Lag_11
  -1.470588   -1.639344   -2.631579 -100.000000    2.631579    1.538462
     Lag_12      Lag_13      Lag_14      Lag_15      Lag_16      Lag_17
   1.388889    1.562500    2.702703  100.000000   -3.448276   -2.000000
     Lag_18
  -1.694915

> la1a + 1:length(la1a)

An object of class "Lagged1d"
Slot *data*:
 Lag_0  Lag_1  Lag_2  Lag_3  Lag_4  Lag_5  Lag_6  Lag_7  Lag_8  Lag_9 Lag_10
  2.00   2.76   3.40   4.02   4.64   5.39   6.32   7.39   8.62   9.99  11.38
Lag_11 Lag_12 Lag_13 Lag_14 Lag_15 Lag_16 Lag_17 Lag_18
 12.65  13.72  14.64  15.37  16.01  16.71  17.50  18.41
```

There is a case to argue for keeping the class in some situations, e.g. when the other argument is a scalar but eventually I decided to keep the simple rule of not trying to preserve the class.

Note however that unary operators and mathematical functions do preserve the class.

4. Multivariate lagged objects

Compute the autocorrelations of a multivariate time series and convert it to a lagged object.

```
> acv2 <- acf(ts.union(mdeaths, fdeaths))
> la2 <- Lagged(acv2)
```

Get the value for lag 1.

```
> la2[1]

, , 1

          [,1]      [,2]
[1,] 0.7570591 0.7356685
[2,] 0.7443093 0.7295201

> acv2$acf[2, ,] # same

          [,1]      [,2]
[1,] 0.7570591 0.7356685
[2,] 0.7443093 0.7295201
```

Indexing in `acf()` is somewhat misterious. For some insight, here is a comparison with a DIY calculation of the autocorrelations.

```
> n <- length(mdeaths)
> tmpcov <- sum((mdeaths - mean(mdeaths)) * (fdeaths - mean(fdeaths)) ) / n
> msd <- sqrt(sum((mdeaths - mean(mdeaths))^2)/n)
> fsd <- sqrt(sum((fdeaths - mean(fdeaths))^2)/n)
> tmpcov1 <- sum((mdeaths - mean(mdeaths))[2:n] * (fdeaths - mean(fdeaths))[1:(n-1)] ) / n
> tmpcov1 / (msd * fsd)

[1] 0.7356685

> la2[[1]][1,2] == tmpcov1 / (msd * fsd) # FALSE, but:

[1] FALSE

> la2[[1]][1,2] - tmpcov1 / (msd * fsd)  # only numerically different

[1] 2.220446e-16
```

Some examples for the correspondence between the indices in lagged objects and those from `acf()`.

```
> la2[[1]][1,2] == acv2$acf[2, 1, 2] # TRUE

[1] TRUE

> la2[0]

, , 1

          [,1]      [,2]
[1,] 1.0000000 0.9762413
[2,] 0.9762413 1.0000000

> acv2[0]

Autocorrelations of series ‘ts.union(mdeaths, fdeaths)’, by lag

, , mdeaths

 mdeaths   fdeaths
 1.000 (0) 0.976 (0)

, , fdeaths

 mdeaths   fdeaths
 0.976 (0) 1.000 (0)

> la2[1]

, , 1

          [,1]      [,2]
[1,] 0.7570591 0.7356685
[2,] 0.7443093 0.7295201
```

```
> acv2[1]

Autocorrelations of series ‘ts.union(mdeaths, fdeaths)’, by lag

, , mdeaths

 mdeaths    fdeaths
 0.717 ( 1) 0.708 (-1)

, , fdeaths

 mdeaths    fdeaths
 0.721 ( 1) 0.716 ( 1)
```

Affiliation:

Georgi N. Boshnakov
School of Mathematics
The University of Manchester
Oxford Road, Manchester M13 9PL, UK
URL: http://www.maths.manchester.ac.uk/~gb/