Managing chronological objects with timeDate and timeSeries

Yohan Chalabi and Diethelm Wuertz

ITP ETH, Zurich
Rmetrics Association, Zurich
Finance Online, Zurich

useR! 2009
Outline

1. timeDate Class
   - timeDate Definition
   - Financial Center and Holiday Management

2. timeSeries Class
   - timeSeries Definition
   - Manipulating a timeSeries
   - Adding New Methods
   - @recordIDs Concept

3. Summary
Outline

1. timeDate Class
   - timeDate Definition
   - Financial Center and Holiday Management

2. timeSeries Class
   - timeSeries Definition
   - Manipulating a timeSeries
   - Adding New Methods
   - @recordIDsConcept

3. Summary
timeDate class is for

- mixing data collected in different time zones
- calendar manipulations for business days, weekends, public and ecclesiastical holidays.
- and is almost compatible with the same class in S-Plus.
The `timeDate` class represents calendar dates and times as

```r
> library(timeDate)
> showClass("timeDate")

Class "timeDate" [package "timeDate"]

Slots:

Name: Data format FinCenter
Class: POSIXct character character

where @Data are the timestamps in POSIXct, @format is the format typically applied to @Data and @FinCenter is the financial center.
Create a `timeDate` object

```r
> ZH <- timeDate("2009-01-01 16:00:00", zone = "GMT", FinCenter = "Zurich")
> NY <- timeDate("2009-01-01 18:00:00", zone = "GMT", FinCenter = "NewYork")
> c(ZH, NY)

Zurich
[1] [2009-01-01 17:00:00] [2009-01-01 19:00:00]

> c(NY, ZH)

NewYork
[1] [2009-01-01 13:00:00] [2009-01-01 11:00:00]
```
Many operations can be performed on `timeDate` objects.

- Math Operations
- Lagging
- Rounding and Truncating
- Subsetting
- Logical Test
- Coercions and Transformation
- Concatentation and Reorderings
Each financial center has an associated function which returns its daylight saving time rule (DST). These functions are named as their financial center, e.g. `Zurich()`, and return a `data.frame` with 4 columns,

```r
> listFinCenter("Europe/[AB].*")

[1] "Europe/Amsterdam" "Europe/Andorra"
[3] "Europe/Athens"   "Europe/Belgrade"
[5] "Europe/Berlin"   "Europe/Bratislava"
[7] "Europe/Brussels" "Europe/Bucharest"
[9] "Europe/Budapest"

> head(Zurich(), 8)

  Zurich offSet isdst   TimeZone      numeric
   1 1901-12-14 20:45:52 3600     0     CET -2147397248
   2 1941-05-05 00:00:00 7200     1   CEST  -904435200
   3 1941-10-06 00:00:00 3600     0     CET  -891129600
   4 1942-05-04 00:00:00 7200     1   CEST  -872985600
   5 1942-10-05 00:00:00 3600     0     CET  -859680000
   6 1981-03-29 01:00:00 7200     1   CEST   354675600
   7 1981-09-27 01:00:00 3600     0     CET   370400400
   8 1982-03-28 01:00:00 7200     1   CEST   386125200
```
Holidays

There are different functions to compute:

- the last day in a given month and year,
- the n-days before or after a given date,
- the n-th occurrences of the n-days for a specified year/month,
- or the last n-days for a specified year/month.
Holidays

```r
> tH <- listHolidays()
> # number of holiday days available in timeDate
> length(tH)
[1] 115

> # the first 10
> head(tH, 10)

[1] "Advent1st"      "Advent2nd"
[5] "AllSaints"      "AllSouls"
[7] "Annunciation"  "Ascension"
[9] "AshWednesday"  "AssumptionOfMary"

> # The date of Easter for the next 3 years:
> Easter(2009:(2009+3))

GMT
[1] [2009-04-12] [2010-04-04] [2011-04-24] [2012-04-08]
```
The following three functions can be used as model to build new holiday calendars.

- `holidayZURICH()`: the Zurich holiday calendar,
- `holidayNYSE()`: the NYSE stock exchange holiday calendar
- and `holidayTSX()`: the TSX holiday calendar.

Weekdays, weekends, business days and holidays can be tested with the functions:

- `isWeekday()`
- `isWeekend()`
- `isBizday()`
- `isHoliday()`
Outline

1. timeDate Class
   - timeDate Definition
   - Financial Center and Holiday Management

2. timeSeries Class
   - timeSeries Definition
   - Manipulating a timeSeries
   - Adding New Methods
   - @recordIDsConcept

3. Summary
The `timeSeries` class represents time series as

```r
> library(timeSeries)
> showClass("timeSeries")
```

Class "timeSeries" [package "timeSeries"]

Slots:

Name: .Data units positions
Class: matrix character numeric

Name: format FinCenter recordIDs
Class: character character data.frame

Name: title documentation
Class: character character

Extends:
Class "structure", from data part
Class "vector", by class "structure", distance 2, with explicit coerce

**Note**: `timeSeries` extends the virtual class structure
timeSeries class

> data <- matrix(round(rnorm(6), 3), ncol = 2)
> td <- timeCalendar()[1:3]
> ts <- timeSeries(data, td)
> ts

GMT

<table>
<thead>
<tr>
<th></th>
<th>TS.1</th>
<th>TS.2</th>
</tr>
</thead>
<tbody>
<tr>
<td>2009-01-01</td>
<td>0.084</td>
<td>0.858</td>
</tr>
<tr>
<td>2009-02-01</td>
<td>-0.238</td>
<td>-1.151</td>
</tr>
<tr>
<td>2009-03-01</td>
<td>-0.158</td>
<td>-0.768</td>
</tr>
</tbody>
</table>
Manipulating a timeSeries

Additional timeSeries operations which might be different from other time series packages.

- Sorting and reverting
- Aggregation
- Lagging
- Rolling windows
- Binding and merging
The time stamps of `timeSeries` objects can be sampled, sorted, and reverted.

```r
> ts <- dummySeries()
> ts

GMT

<table>
<thead>
<tr>
<th>Date</th>
<th>TS.1</th>
<th>TS.2</th>
</tr>
</thead>
<tbody>
<tr>
<td>2009-01-01</td>
<td>0.050420</td>
<td>0.9502815</td>
</tr>
<tr>
<td>2009-02-01</td>
<td>0.119620</td>
<td>0.4814418</td>
</tr>
<tr>
<td>2009-03-01</td>
<td>0.099209</td>
<td>0.9890132</td>
</tr>
<tr>
<td>2009-04-01</td>
<td>0.051417</td>
<td>0.4020588</td>
</tr>
<tr>
<td>2009-05-01</td>
<td>0.889680</td>
<td>0.1110520</td>
</tr>
<tr>
<td>2009-06-01</td>
<td>0.225331</td>
<td>0.7122814</td>
</tr>
<tr>
<td>2009-07-01</td>
<td>0.361068</td>
<td>0.3452739</td>
</tr>
<tr>
<td>2009-08-01</td>
<td>0.026264</td>
<td>0.3224443</td>
</tr>
<tr>
<td>2009-09-01</td>
<td>0.778356</td>
<td>0.4797025</td>
</tr>
<tr>
<td>2009-10-01</td>
<td>0.810493</td>
<td>0.0053789</td>
</tr>
<tr>
<td>2009-11-01</td>
<td>0.277139</td>
<td>0.6304754</td>
</tr>
<tr>
<td>2009-12-01</td>
<td>0.239023</td>
<td>0.1460500</td>
</tr>
</tbody>
</table>
```
### Sorting and Ordering

```r
> sa <- sample(ts)
> sa

GMT

<table>
<thead>
<tr>
<th>GMT</th>
<th>TS.1</th>
<th>TS.2</th>
</tr>
</thead>
<tbody>
<tr>
<td>2009-12-01</td>
<td>0.239023</td>
<td>0.146050</td>
</tr>
<tr>
<td>2009-10-01</td>
<td>0.810493</td>
<td>0.005379</td>
</tr>
<tr>
<td>2009-03-01</td>
<td>0.099209</td>
<td>0.989013</td>
</tr>
<tr>
<td>2009-01-01</td>
<td>0.050420</td>
<td>0.950281</td>
</tr>
<tr>
<td>2009-02-01</td>
<td>0.119620</td>
<td>0.481441</td>
</tr>
<tr>
<td>2009-06-01</td>
<td>0.225331</td>
<td>0.712281</td>
</tr>
<tr>
<td>2009-09-01</td>
<td>0.778356</td>
<td>0.479702</td>
</tr>
<tr>
<td>2009-04-01</td>
<td>0.051417</td>
<td>0.402058</td>
</tr>
<tr>
<td>2009-08-01</td>
<td>0.026264</td>
<td>0.322444</td>
</tr>
<tr>
<td>2009-05-01</td>
<td>0.889680</td>
<td>0.111052</td>
</tr>
<tr>
<td>2009-07-01</td>
<td>0.361068</td>
<td>0.345273</td>
</tr>
<tr>
<td>2009-11-01</td>
<td>0.277139</td>
<td>0.630475</td>
</tr>
</tbody>
</table>
### Sorting and Ordering

```r
> so <- sort(sa)
> so
```

<table>
<thead>
<tr>
<th>GMT</th>
<th>TS.1</th>
<th>TS.2</th>
</tr>
</thead>
<tbody>
<tr>
<td>2009-01-01</td>
<td>0.050420</td>
<td>0.9502815</td>
</tr>
<tr>
<td>2009-02-01</td>
<td>0.119620</td>
<td>0.4814418</td>
</tr>
<tr>
<td>2009-03-01</td>
<td>0.099209</td>
<td>0.9890132</td>
</tr>
<tr>
<td>2009-04-01</td>
<td>0.051417</td>
<td>0.4020588</td>
</tr>
<tr>
<td>2009-05-01</td>
<td>0.889680</td>
<td>0.1110520</td>
</tr>
<tr>
<td>2009-06-01</td>
<td>0.225331</td>
<td>0.7122814</td>
</tr>
<tr>
<td>2009-07-01</td>
<td>0.361068</td>
<td>0.3452739</td>
</tr>
<tr>
<td>2009-08-01</td>
<td>0.026264</td>
<td>0.3224443</td>
</tr>
<tr>
<td>2009-09-01</td>
<td>0.778356</td>
<td>0.4797025</td>
</tr>
<tr>
<td>2009-10-01</td>
<td>0.810493</td>
<td>0.0053789</td>
</tr>
<tr>
<td>2009-11-01</td>
<td>0.277139</td>
<td>0.6304754</td>
</tr>
<tr>
<td>2009-12-01</td>
<td>0.239023</td>
<td>0.1460500</td>
</tr>
</tbody>
</table>
### Sorting and Ordering

```r
> re <- rev(so)
> re

<table>
<thead>
<tr>
<th>GMT</th>
<th>TS.1</th>
<th>TS.2</th>
</tr>
</thead>
<tbody>
<tr>
<td>2009-12-01</td>
<td>0.239023</td>
<td>0.1460500</td>
</tr>
<tr>
<td>2009-11-01</td>
<td>0.277139</td>
<td>0.6304754</td>
</tr>
<tr>
<td>2009-10-01</td>
<td>0.810493</td>
<td>0.0053789</td>
</tr>
<tr>
<td>2009-09-01</td>
<td>0.778356</td>
<td>0.4797025</td>
</tr>
<tr>
<td>2009-08-01</td>
<td>0.026264</td>
<td>0.3224443</td>
</tr>
<tr>
<td>2009-07-01</td>
<td>0.361068</td>
<td>0.3452739</td>
</tr>
<tr>
<td>2009-06-01</td>
<td>0.225331</td>
<td>0.7122814</td>
</tr>
<tr>
<td>2009-05-01</td>
<td>0.889680</td>
<td>0.1110520</td>
</tr>
<tr>
<td>2009-04-01</td>
<td>0.051417</td>
<td>0.4020588</td>
</tr>
<tr>
<td>2009-03-01</td>
<td>0.099209</td>
<td>0.9890132</td>
</tr>
<tr>
<td>2009-02-01</td>
<td>0.119620</td>
<td>0.4814418</td>
</tr>
<tr>
<td>2009-01-01</td>
<td>0.050420</td>
<td>0.9502815</td>
</tr>
</tbody>
</table>
```
Aggregation

> library(fEcofin)
> LPP <- as.timeSeries(data(SWXLP))[,4:6]
> (by <- timeSequence(from = "2003-01-01", to = "2005-01-01", by = "quarter"))

GMT

> aggregate(LPP, by, mean)

GMT
                     LP25  LP40  LP60
2003-01-01  100.37  97.073 92.658
2003-04-01  97.46   86.600 73.363
2003-07-01 100.43  90.155 77.372
2003-10-01 103.42  94.390 82.812
2004-01-01 104.86  96.218 84.984
2004-04-01 108.08  99.842 88.920
2004-07-01 107.71  99.763 89.154
2004-10-01 107.71  99.238 88.076
2005-01-01 109.85 101.101 89.602
Rolling windows can be performed with `applySeries()`.

```r
> by <- periods(time(LPP), period = "24m", by = "6m")
> applySeries(LPP, from = by$from, to = by$to, FUN = "colMeans")

GMT

<table>
<thead>
<tr>
<th></th>
<th>LP25</th>
<th>LP40</th>
<th>LP60</th>
</tr>
</thead>
<tbody>
<tr>
<td>2001-12-31</td>
<td>100.911</td>
<td>99.438</td>
<td>97.310</td>
</tr>
<tr>
<td>2002-06-30</td>
<td>101.209</td>
<td>98.476</td>
<td>94.639</td>
</tr>
<tr>
<td>2002-12-31</td>
<td>100.252</td>
<td>95.199</td>
<td>88.487</td>
</tr>
<tr>
<td>2003-06-30</td>
<td>99.437</td>
<td>92.248</td>
<td>83.037</td>
</tr>
<tr>
<td>2003-12-31</td>
<td>100.420</td>
<td>92.120</td>
<td>81.543</td>
</tr>
<tr>
<td>2004-06-30</td>
<td>102.222</td>
<td>93.126</td>
<td>81.524</td>
</tr>
<tr>
<td>2004-12-31</td>
<td>104.943</td>
<td>95.920</td>
<td>84.295</td>
</tr>
<tr>
<td>2005-06-30</td>
<td>108.678</td>
<td>100.223</td>
<td>89.070</td>
</tr>
<tr>
<td>2005-12-31</td>
<td>112.648</td>
<td>104.784</td>
<td>94.170</td>
</tr>
<tr>
<td>2006-12-31</td>
<td>120.190</td>
<td>114.424</td>
<td>105.964</td>
</tr>
</tbody>
</table>
```
There are four functions to bind time series together. These are, with increasing complexity, `c()`, `cbind()`, `rbind()` and `merge()`.

```r
> (ts1 <- timeSeries(matrix(rnorm(4), ncol = 2), c("2009-01-01", "2009-03-01")))
GMT
     TS.1  TS.2
2009-01-01 -0.195804 1.18347
2009-03-01 -0.063472 -0.89746

> (ts2 <- timeSeries(matrix(rnorm(4), ncol = 2), c("2009-02-01", "2009-04-01")))
GMT
     TS.1  TS.2
2009-02-01 -0.11698 -1.2321
2009-04-01 -1.39368 -1.6083
```
c()  

> c(ts1, ts2)  
[1] -0.195804 -0.063472 1.183473 -0.897456 -0.116982  
[6] -1.393675 -1.232076 -1.608285
```r
> cbind(ts1, ts2)

GMT

<table>
<thead>
<tr>
<th></th>
<th>TS.1.1</th>
<th>TS.2.1</th>
<th>TS.1.2</th>
<th>TS.2.2</th>
</tr>
</thead>
<tbody>
<tr>
<td>2009-01-01</td>
<td>-0.195804</td>
<td>1.18347</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>2009-02-01</td>
<td>NA</td>
<td>NA</td>
<td>-0.11698</td>
<td>-1.2321</td>
</tr>
<tr>
<td>2009-03-01</td>
<td>-0.063472</td>
<td>-0.89746</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>2009-04-01</td>
<td>NA</td>
<td>NA</td>
<td>-1.39368</td>
<td>-1.6083</td>
</tr>
</tbody>
</table>
```
```r
> rbind(ts1, ts2)

GMT

<table>
<thead>
<tr>
<th>GMT</th>
<th>TS.1</th>
<th>TS.2</th>
</tr>
</thead>
<tbody>
<tr>
<td>2009-01-01</td>
<td>-0.195804</td>
<td>1.18347</td>
</tr>
<tr>
<td>2009-03-01</td>
<td>-0.063472</td>
<td>-0.89746</td>
</tr>
<tr>
<td>2009-02-01</td>
<td>-0.116982</td>
<td>-1.23208</td>
</tr>
<tr>
<td>2009-04-01</td>
<td>-1.393675</td>
<td>-1.60829</td>
</tr>
</tbody>
</table>
```
merge()

> merge(ts1, ts2)

GMT

<table>
<thead>
<tr>
<th>Date</th>
<th>TS.1</th>
<th>TS.2</th>
</tr>
</thead>
<tbody>
<tr>
<td>2009-01-01</td>
<td>-0.195804</td>
<td>1.18347</td>
</tr>
<tr>
<td>2009-02-01</td>
<td>-0.116982</td>
<td>-1.23208</td>
</tr>
<tr>
<td>2009-03-01</td>
<td>-0.063472</td>
<td>-0.89746</td>
</tr>
<tr>
<td>2009-04-01</td>
<td>-1.393675</td>
<td>-1.60829</td>
</tr>
</tbody>
</table>
Adding New Methods

- Since `timeSeries` is an S4 class, we can use the function `setMethod()` to create new methods for a generic function which has can not handle by default the class.

- In this example, we write a method for the `lowess()` function from the `stats` package.

```r
> setMethod("lowess", "timeSeries",
        function(x, y = NULL, f = 2/3, iter = 3)
          {
            stopifnot(isUnivariate(x))
            series(x) <- stats::lowess(x = x, y, f, iter)$y
            x
          }
)
[1] "lowess"
```
Adding new methods

\[ LP60 \leftarrow \text{LPP[,} "LP60"\text{]} \]
\[ LP60\text{low} \leftarrow \text{lowess}(LP60, f = 0.08) \]
\[ \text{plot}(LP60) \]
\[ \text{lines}(LP60\text{low, col =} "\text{brown}\), lwd = 2) \]
The slot @recordIDs is meant for additional information that we want to keep for each time entries but which is not part of data part.

As starting from timeSeries version '2100.84' we have added a method for the operator '$' to access the @recordIDs as well as the data part.

by default show() will print the data part with the @recordIDs. Note the '*' in the column names of @recordIDs in the output.

@recordIDs can be used to give a data.frame behavior to your time series.

```r
> ts$id <- "id"
> head(ts)

GMT     TS.1     TS.2 id*       > cov(ts)
2009-01-01  0.050420  0.950282 id  TS.1  0.101236 -0.056448
2009-02-01  0.119620  0.481442 id  TS.2 -0.056448  0.097816
2009-03-01  0.099209  0.989013 id
2009-04-01  0.051417  0.402059 id
2009-05-01  0.889680  0.111052 id
2009-06-01  0.225331  0.712281 id
```
A good example is to include turnpoints of the smoothed index to the time series.

We can use the `turnpoints()` function from the R package `pastecs`.

The function determines the number and the positions of extrema, i.e. the turning points, either peaks or pits, in a regular time series.

```r
> library(pastecs)
> setMethod("turnpoints", "timeSeries", function(x) {
    stopifnot(isUnivariate(x))
    tp <- turnpoints(as.ts(x))
    x$peaks <- tp$peaks #-> need timeSeries >= 2100.84
    x$pits <- tp$pits
    x
}

[1] "turnpoints"
```

---

1Ibanez, Grosjean & Etienne, 2009
@recordIDs Example

> head(LP60low <- turnpoints(LP60low))

GMT

<table>
<thead>
<tr>
<th></th>
<th>LP60</th>
<th>peaks*</th>
<th>pits*</th>
</tr>
</thead>
<tbody>
<tr>
<td>2000-01-03</td>
<td>97.730</td>
<td>FALSE</td>
<td>FALSE</td>
</tr>
<tr>
<td>2000-01-04</td>
<td>97.767</td>
<td>FALSE</td>
<td>FALSE</td>
</tr>
<tr>
<td>2000-01-05</td>
<td>97.805</td>
<td>FALSE</td>
<td>FALSE</td>
</tr>
<tr>
<td>2000-01-06</td>
<td>97.842</td>
<td>FALSE</td>
<td>FALSE</td>
</tr>
<tr>
<td>2000-01-07</td>
<td>97.880</td>
<td>FALSE</td>
<td>FALSE</td>
</tr>
<tr>
<td>2000-01-10</td>
<td>97.917</td>
<td>FALSE</td>
<td>FALSE</td>
</tr>
</tbody>
</table>
@recordIDs Example

We plot the original index series and the smoothed series and add points for the peaks and pits in green and red respectively.

```r
> plot(LP60)
> lines(LP60low, col = "brown", lwd = 2)
> points(LP60low[LP60low$peaks,], col = "green3", pch = 24)
> points(LP60low[LP60low$pits,], col = "red", pch = 25)
```

**Note:** the use of the operator '$$'.
@recordIDs Example

```
Example
```

```
2000−01−03 2001−06−22 2002−12−10 2004−05−30 2005−11−17 2007−05−08
70 80 90 100 110 120
```
Outline

1. timeDate Class
   - timeDate Definition
   - Financial Center and Holiday Management

2. timeSeries Class
   - timeSeries Definition
   - Manipulating a timeSeries
   - Adding New Methods
   - @recordIDsConcept

3. Summary
timeSeries is meant to have a matrix like behavior
With some aspects of a data.frame,
It can handle ordered/unordered data and display them in any order.
It takes care of financial centers when merging/binding.
And has facilities to manage calendars thanks to the timeDate package.
D. Wuertz, Y. Chalabi, W. Chen, A. Ellis,
*Portfolio Optimization with R/Rmetrics.*
Finance Online, 2009.

D. Wuertz, Y. Chalabi, A. Ellis,
FAQ - Time Series Objects for R in Finance
http://www.rmetrics.org
> toLatex(sessionInfo())

- R version 2.10.0 Under development (unstable) (2009-07-02 r48890), i686-pc-linux-gnu
- Locale: LC_CTYPE=en_US.UTF-8, LC_NUMERIC ...
- Base packages: base, datasets, graphics, grDevices, methods, stats, utils
- Other packages: boot 1.2-37, fEcofin 2100.77, pastecs 1.3-8, timeDate 2100.86, timeSeries 2100.84
- Loaded via a namespace (and not attached): tools 2.10.0
Managing chronological objects with timeDate and timeSeries

Yohan Chalabi and Diethelm Wuertz

ITP ETH, Zurich
Rmetrics Association, Zurich
Finance Online, Zurich

useR! 2009