



# 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

# 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

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.

# timeDate class

The `timeDate` class represents calendar dates and times as

```
> 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

```
> 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]
```

# Operations

Many operations can be performed on `timeDate` objects.

- Math Operations
- Lagging
- Rounding and Truncating
- Subsetting
- Logical Test
- Coercions and Transformation
- Concatentation and Reorderings

# FinCenter

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,

```
> 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

```
> tH <- listHolidays()
> # number of holiday days available in timeDate
> length(tH)

[1] 115

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

 [1] "Advent1st"          "Advent2nd"
 [3] "Advent3rd"         "Advent4th"
 [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]
```

# Calendar and Logical Test

- 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

# timeSeries class

The timeSeries class represents time series as

```
> 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

	TS.1	TS.2
2009-01-01	0.084	0.858
2009-02-01	-0.238	-1.151
2009-03-01	-0.158	-0.768

# 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

# Sorting and Ordering

The time stamps of `timeSeries` objects can be sampled, sorted, and reverted.

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

```
> ts
```

GMT

	TS.1	TS.2
2009-01-01	0.050420	0.9502815
2009-02-01	0.119620	0.4814418
2009-03-01	0.099209	0.9890132
2009-04-01	0.051417	0.4020588
2009-05-01	0.889680	0.1110520
2009-06-01	0.225331	0.7122814
2009-07-01	0.361068	0.3452739
2009-08-01	0.026264	0.3224443
2009-09-01	0.778356	0.4797025
2009-10-01	0.810493	0.0053789
2009-11-01	0.277139	0.6304754
2009-12-01	0.239023	0.1460500



# Sorting and Ordering

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

GMT

	TS.1	TS.2
2009-12-01	0.239023	0.1460500
2009-10-01	0.810493	0.0053789
2009-03-01	0.099209	0.9890132
2009-01-01	0.050420	0.9502815
2009-02-01	0.119620	0.4814418
2009-06-01	0.225331	0.7122814
2009-09-01	0.778356	0.4797025
2009-04-01	0.051417	0.4020588
2009-08-01	0.026264	0.3224443
2009-05-01	0.889680	0.1110520
2009-07-01	0.361068	0.3452739
2009-11-01	0.277139	0.6304754

# Sorting and Ordering

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

GMT

	TS.1	TS.2
2009-01-01	0.050420	0.9502815
2009-02-01	0.119620	0.4814418
2009-03-01	0.099209	0.9890132
2009-04-01	0.051417	0.4020588
2009-05-01	0.889680	0.1110520
2009-06-01	0.225331	0.7122814
2009-07-01	0.361068	0.3452739
2009-08-01	0.026264	0.3224443
2009-09-01	0.778356	0.4797025
2009-10-01	0.810493	0.0053789
2009-11-01	0.277139	0.6304754
2009-12-01	0.239023	0.1460500

# Sorting and Ordering

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

GMT

	TS.1	TS.2
2009-12-01	0.239023	0.1460500
2009-11-01	0.277139	0.6304754
2009-10-01	0.810493	0.0053789
2009-09-01	0.778356	0.4797025
2009-08-01	0.026264	0.3224443
2009-07-01	0.361068	0.3452739
2009-06-01	0.225331	0.7122814
2009-05-01	0.889680	0.1110520
2009-04-01	0.051417	0.4020588
2009-03-01	0.099209	0.9890132
2009-02-01	0.119620	0.4814418
2009-01-01	0.050420	0.9502815

# Aggregation

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

GMT

```
[1] [2003-01-01] [2003-04-01] [2003-07-01] [2003-10-01]
[5] [2004-01-01] [2004-04-01] [2004-07-01] [2004-10-01]
[9] [2005-01-01]
```

```
> 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

Rolling windows can be performed with `applySeries()`.

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

GMT

	LP25	LP40	LP60
2001-12-31	100.911	99.438	97.310
2002-06-30	101.209	98.476	94.639
2002-12-31	100.252	95.199	88.487
2003-06-30	99.437	92.248	83.037
2003-12-31	100.420	92.120	81.543
2004-06-30	102.222	93.126	81.524
2004-12-31	104.943	95.920	84.295
2005-06-30	108.678	100.223	89.070
2005-12-31	112.648	104.784	94.170
2006-06-30	116.179	109.216	99.525
2006-12-31	120.190	114.424	105.964

# Merging and Binding

There are four functions to bind time series together. These are, with increasing complexity, `c()`, `cbind()`, `rbind()` and `merge()`.

```
> (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
```

## cbind()

```
> cbind(ts1, ts2)
```

```
GMT
```

	TS.1.1	TS.2.1	TS.1.2	TS.2.2
2009-01-01	-0.195804	1.18347	NA	NA
2009-02-01	NA	NA	-0.11698	-1.2321
2009-03-01	-0.063472	-0.89746	NA	NA
2009-04-01	NA	NA	-1.39368	-1.6083



# rbind()

```
> rbind(ts1, ts2)
```

```
GMT
```

	TS.1_TS.1	TS.2_TS.2
2009-01-01	-0.195804	1.18347
2009-03-01	-0.063472	-0.89746
2009-02-01	-0.116982	-1.23208
2009-04-01	-1.393675	-1.60829

# merge()

```
> merge(ts1, ts2)
```

```
GMT
```

	TS.1	TS.2
2009-01-01	-0.195804	1.18347
2009-02-01	-0.116982	-1.23208
2009-03-01	-0.063472	-0.89746
2009-04-01	-1.393675	-1.60829

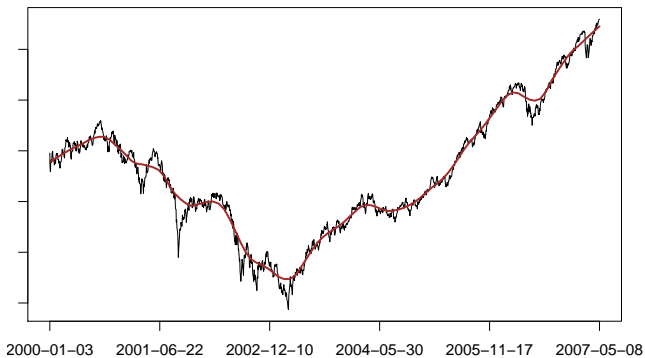
## Adding New Methods

- 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.

```
> 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 <- LPP[, "LP60"]  
> LP60low <- lowess(LP60, f = 0.08)  
> plot(LP60)  
> lines(LP60low, col = "brown", lwd = 2)
```



# @recordIDs Concept

- 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.

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

```
GMT  
                TS.1      TS.2 id*  
2009-01-01 0.050420 0.950282 id  
2009-02-01 0.119620 0.481442 id  
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
```

```
> cov(ts)  
                TS.1      TS.2  
TS.1 0.101236 -0.056448  
TS.2 -0.056448 0.097816
```

## @recordIDs Example

- 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`<sup>1</sup>.
- The function determines the number and the positions of extrema, i.e. the turning points, either peaks or pits, in a regular time series.

```
> 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"
```

---

<sup>1</sup>Ibanez, Grosjean & Etienne, 2009

# @recordIDs Example

```
> head(LP60low <- turnpoints(LP60low))
```

GMT

	LP60	peaks*	pits*
2000-01-03	97.730	FALSE	FALSE
2000-01-04	97.767	FALSE	FALSE
2000-01-05	97.805	FALSE	FALSE
2000-01-06	97.842	FALSE	FALSE
2000-01-07	97.880	FALSE	FALSE
2000-01-10	97.917	FALSE	FALSE










# 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.

# References I

 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\_NUMERI ...
- 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

