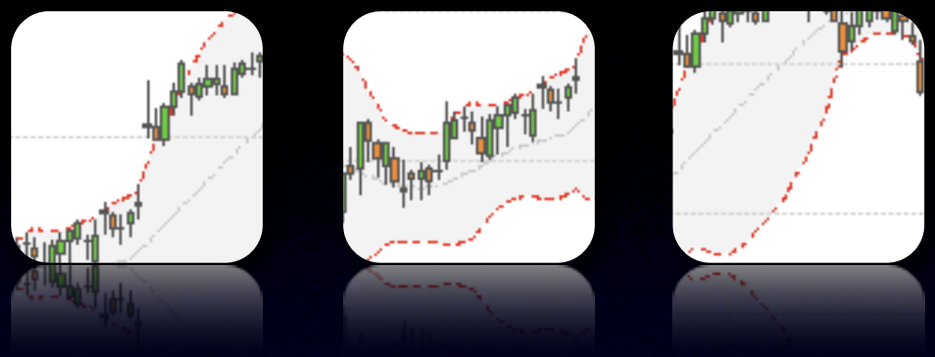


Financial Time-Series Tools

xts and chartSeries

Jeffrey A. Ryan
jeffrey.ryan @ insightalgo.com

Joshua M. Ulrich
joshua.m.ulrich @ gmail.com

Presented by Jeffrey Ryan at:
Rmetrics: Computational Finance and Financial Engineering Workshop
June 29 - July 3, 2008, Meielisalp, Lake Thune, Switzerland

www.quantmod.com/Rmetrics2008

Thanks

Diethelm Wuertz
The Rmetrics Foundation
Peter Carl
Brian Peterson
Joshua Ulrich
Gabor Grothendieck
Dirk Eddelbuettel

Part I - xts

- The Idea Behind xts
- User Benefits
- Developer Benefits
- Summary

Part II - quantmod

- Why quantmod
- getSymbols
- chartSeries
- Summary

Extensible Time-Series

Jeffrey A. Ryan and Joshua M. Ulrich

http://xts.r-forge.r-project.org

The Idea

Provide for uniform handling of R's different time-based data classes, maximizing native format information preservation and allowing for user level customization and extension, while simplifying cross-class interoperability.

(from the xts DESCRIPTION file)

The Idea

~~*Provide for uniform handling of R's different time-based data classes, maximizing native format information preservation and allowing for user level customization and extension, while simplifying cross-class interoperability.*~~

~~(from the xts DESCRIPTION file)~~

Simplify time-series data!

What is xts?

What is xts?

1. An S3 class extending zoo

What is xts?

1. An S3 class extending zoo
2. Requires indexing based on a recognized time-based class

Any of POSIXct, Date, chron, timeDate, yearmon, or yearqtr

What is xts?

1. An S3 class extending zoo
2. Requires indexing based on a recognized time-based class

   Any of POSIXct, Date, chron, timeDate, yearmon, or yearqtr
3. Allows arbitrary hidden attributes with xtsAttributes

What is xts?

1. An S3 class extending zoo
2. Requires indexing based on a recognized time-based class

   Any of POSIXct, Date, chron, timeDate, yearmon, or yearqtr

3. Allows arbitrary hidden attributes with xtsAttributes
4. Tools for lossless conversion among classes

   as.xts, try.xts, reclass, and Reclass

Why use xts?

Why use xts?

1. A true time-based version of the popular and robust zoo class

Why use xts?

1. A true time-based version of the popular and robust zoo class
2. New behavior that accounts for time - subsetting, etc.

Why use xts?

1. A true time-based version of the popular and robust zoo class
2. New behavior that accounts for time - subsetting, etc.
3. Flexibility to augment with custom hidden attributes - metadata

Why use xts?

1. A true time-based version of the popular and robust zoo class
2. New behavior that accounts for time - subsetting, etc.
3. Flexibility to augment with custom hidden attributes - metadata
4. Smart conversion tools - use xts functionality with other classes

Why use xts?

1. A true time-based version of the popular and robust zoo class
2. New behavior that accounts for time - subsetting, etc.
3. Flexibility to augment with custom hidden attributes - metadata
4. Smart conversion tools - use xts functionality with other classes
5. Time-based tools like fast aggregation, periodic functions, etc.

Why develop with xts?

Why develop with xts?

1. A true time-based version of the popular and robust zoo class
2. New behavior that accounts for time - subsetting, etc.
3. Flexibility to augment with custom hidden attributes - metadata
4. Smart conversion tools - use xts functionality with other classes
5. Time-based tools like fast aggregation, periodic functions, etc.

Why develop with xts?

1. A true time-based version of the popular and robust zoo class
2. New behavior that accounts for time - subsetting, etc.
3. Flexibility to augment with custom hidden attributes - metadata
4. Smart conversion tools - use xts functionality with other classes
5. Time-based tools like fast aggregation, periodic functions, etc.
6. Accept any time-based class in all your functions - without methods

Why develop with xts?

1. A true time-based version of the popular and robust zoo class
2. New behavior that accounts for time - subsetting, etc.
3. Flexibility to augment with custom hidden attributes - metadata
4. Smart conversion tools - use xts functionality with other classes
5. Time-based tools like fast aggregation, periodic functions, etc.
6. Accept any time-based class in all your functions - without methods
7. Provide a seamless user experience - *his* choice of class instead of yours

Why develop with xts?

1. A true time-based version of the popular and robust zoo class
2. New behavior that accounts for time - subsetting, etc.
3. Flexibility to augment with custom hidden attributes - metadata
4. Smart conversion tools - use xts functionality with other classes
5. Time-based tools like fast aggregation, periodic functions, etc.
6. Accept any time-based class in all your functions - without methods
7. Provide a seamless user experience - *his* choice of class instead of yours
8. Less data-specific coding and testing means more time to develop

Why develop with xts?

1. A true time-based version of the popular and robust zoo class
2. New behavior that accounts for time - subsetting, etc.
3. Flexibility to augment with custom hidden attributes - metadata
4. Smart conversion tools - use xts functionality with other classes
5. Time-based tools like fast aggregation, periodic functions, etc.
6. Accept any time-based class in all your functions - without methods
7. Provide a seamless user experience - *his* choice of class instead of yours
8. Less data-specific coding and testing means more time to develop

Let's start by looking at using xts...

Using xts

- as.xts()
- Time-based tools

  first(), last(), [.xts, to.period(), and period.apply()

- reclass()

Create an object

- Convert automatically with as.xts():

  ts, data.frame, matrix, zoo, timeSeries, its, irts, and xts.

- Add additional meta-data with ... arg at construction, or with xtsAttributes
- xts() constructor also available

Create an object

- Convert automatically with as.xts():

  ts, data.frame, matrix, zoo, timeSeries, its, irts, and xts.

- Add additional meta-data with ... arg at construction, or with xtsAttributes
- xts() constructor also available

You now have a time-based object!

Time-based object means ...

Time-based object means ...

... time-aware tools!

first and last

- Provide a time-based equivalent of head() and tail()
- Allow for natural-language subsetting

  find the first or last '3 weeks' or '5 months' of a dataset, regardless of underlying periodicity. e.g. last (MSFT, '3 weeks') works as well on minute data as it does on daily.

- Positive and negative indexing

  '-3 weeks' returns all data except the first or last 3 weeks.

```
> first(MSFT,'2 weeks')
           MSFT.Open MSFT.High MSFT.Low MSFT.Close MSFT.Volume MSFT.Adjusted
2007-01-03     29.91     30.25    29.40      29.86    76935100          29.35
2007-01-04     29.70     29.97    29.44      29.81    45774500          29.30
2007-01-05     29.63     29.75    29.45      29.64    44607200          29.13
2007-01-08     29.65     30.10    29.53      29.93    50220200          29.41
2007-01-09     30.00     30.18    29.73      29.96    44636600          29.44
2007-01-10     29.80     29.89    29.43      29.66    55017400          29.15
2007-01-11     29.76     30.75    29.65      30.70    99464300          30.17
2007-01-12     30.65     31.39    30.64      31.21   103972500          30.67
```

```
> last(MSFT,'2 months')
           MSFT.Open MSFT.High MSFT.Low MSFT.Close MSFT.Volume MSFT.Adjusted
2008-03-03     27.24     27.39    26.87      26.99    76544300          26.99
2008-03-04     27.02     27.63    26.96      27.59    86904000          27.59
2008-03-05     27.75     28.41    27.70      28.12   106433600          28.12
2008-03-06     28.06     28.17    27.50      27.57    91127700          27.57
2008-03-07     27.34     28.07    27.32      27.87    77597600          27.87
2008-03-10     27.83     28.26    27.75      28.05    72175100          28.05
2008-03-11     28.40     29.34    28.38      29.28    98740700          29.28
2008-03-12     29.43     29.49    28.54      28.63    75885400          28.63
2008-03-13     28.54     28.99    28.16      28.62    84552200          28.62
2008-03-14     28.72     29.01    27.64      27.96   105201700          27.96
2008-03-17     27.30     28.73    27.28      28.30    84490100          28.30
2008-03-18     28.67     29.48    28.67      29.42    83323800          29.42
2008-03-19     29.38     29.59    28.62      28.62    61442100          28.62
2008-03-20     28.74     29.22    28.59      29.18    60170200          29.18
2008-03-24     29.33     29.40    29.06      29.17    48294700          29.17
2008-03-25     29.33     29.37    28.94      29.14    49149000          29.14
2008-03-26     29.03     29.07    28.38      28.56    45855300          28.56
2008-03-27     28.48     28.49    28.00      28.05    47688900          28.05
2008-03-28     28.23     28.43    27.83      27.91    49244000          27.91
2008-03-31     27.88     28.59    27.84      28.38    46762400          28.38
2008-04-01     28.83     29.54    28.63      29.50    65774200          29.50
```

[.xts

Extends the ISO 8601-style of range specification to the standard R-style single-bracket subsetting mechanism using [from/to] or [from::to]

```
> MSFT['2007-03-01/2007-03-22']
           MSFT.Open MSFT.High MSFT.Low MSFT.Close MSFT.Volume MSFT.Adjusted
2007-03-01     27.82     28.33    27.73      28.09    80175700         27.70
2007-03-02     28.02     28.16    27.76      27.76    63254700         27.38
2007-03-05     27.49     27.91    27.41      27.55    56454300         27.17
2007-03-06     27.80     27.94    27.65      27.83    49361800         27.45
2007-03-07     27.76     27.90    27.55      27.61    52044700         27.23
2007-03-08     27.72     27.85    26.60      27.32    72175200         26.94
2007-03-09     27.42     27.48    27.03      27.29    80125000         26.91
2007-03-12     27.18     27.48    27.13      27.44    36516400         27.06
2007-03-13     27.25     27.40    26.71      26.72    75169500         26.35
2007-03-14     26.82     27.40    26.73      27.40    75730300         27.02
2007-03-15     27.32     27.47    27.20      27.28    51757100         26.90
2007-03-16     27.35     27.48    27.20      27.33    65055300         26.95
2007-03-19     27.34     27.83    27.20      27.83    49412000         27.45
2007-03-20     27.93     28.16    27.76      27.84    47902400         27.46
2007-03-21     27.90     28.52    27.56      28.52    72808200         28.13
2007-03-22     28.52     28.55    28.01      28.27    47934900         27.88
> 
```

[.xts

Extends the ISO 8601-style of range specification to the standard R-style single-bracket subsetting mechanism using [from/to] or [from::to]

```
> MSFT['2007-03-01/2007-03-22']
           MSFT.Open MSFT.High MSFT.Low MSFT.Close MSFT.Volume MSFT.Adjusted
2007-03-01     27.82     28.33    27.73      28.09    80175700         27.70
2007-03-02     28.02     28.16    27.76      27.76    63254700         27.38
2007-03-05     27.49     27.91    27.41      27.55    56454300         27.17
2007-03-06     27.80     27.94    27.65      27.83    49361800         27.45
2007-03-07     27.76     27.90    27.55      27.61    52044700         27.23
2007-03-08     27.72     27.85    26.60      27.32    72175200         26.94
2007-03-09     27.42     27.48    27.03      27.29    80125000         26.91
2007-03-12     27.18     27.48    27.13      27.44    36516400         27.06
2007-03-13     27.25     27.40    26.71      26.72    75169500         26.35
2007-03-14     26.82     27.40    26.73      27.40    75730300         27.02
2007-03-15     27.32     27.47    27.20      27.28    51757100         26.90
2007-03-16     27.35     27.48    27.20      27.33    65055300         26.95
2007-03-19     27.34     27.83    27.20      27.83    49412000         27.45
2007-03-20     27.93     28.16    27.76      27.84    47902400         27.46
2007-03-21     27.90     28.52    27.56      28.52    72808200         28.13
2007-03-22     28.52     28.55    28.01      28.27    47934900         27.88
> 
```

This works on any periodicity or index class!

fast periodicity conversion

```
> to.monthly(MSFT)
         MSFT.Open MSFT.High MSFT.Low MSFT.Close MSFT.Volume MSFT.Adjusted
Jan 2007     29.91     31.48    29.40      30.86  1324518200          30.33
Feb 2007     30.84     30.94    27.79      28.17  1290850900          27.78
Mar 2007     27.82     28.55    26.60      27.87  1269506500          27.48
Apr 2007     27.89     30.74    27.56      29.94   958964900          29.53
May 2007     29.94     31.16    29.90      30.69  1327154700          30.36
Jun 2007     30.79     30.90    29.04      29.47  1181412800          29.16
Jul 2007     29.67     31.84    28.95      28.99  1295548000          28.68
Aug 2007     28.95     30.10    27.51      28.73  1228579500          28.52
Sep 2007     28.50     29.85    28.27      29.46  1117419500          29.25
Oct 2007     29.46     37.00    29.29      36.81  1771226400          36.55
Nov 2007     36.53     37.50    32.68      33.60  1829448700          33.47
Dec 2007     33.50     36.72    32.63      35.60  1064817100          35.46
Jan 2008     35.79     35.96    31.04      32.60  1950301600          32.47
Feb 2008     31.06     33.25    27.02      27.20  2323373900          27.20
Mar 2008     27.24     29.59    26.87      28.38  1451582800          28.38
Apr 2008     28.83     29.54    28.63      29.50    65774200          29.50
```

to.period allows for fast changes to the periodicity of univariate and OHLC data.

Convert from one periodicity to another - e.g. daily to monthly, or hourly to daily with one function.

fast periodicity conversion

```
> to.monthly(MSFT)
         MSFT.Open MSFT.High MSFT.Low MSFT.Close MSFT.Volume MSFT.Adjusted
Jan 2007     29.91     31.48    29.40      30.86  1324518200          30.33
Feb 2007     30.84     30.94    27.79      28.17  1290850900          27.78
Mar 2007     27.82     28.55    26.60      27.87  1269506500          27.48
Apr 2007     27.89     30.74    27.56      29.94   958964900          29.53
May 2007     29.94     31.16    29.90      30.69  1327154700          30.36
Jun 2007     30.79     30.90    29.04      29.47  1181412800          29.16
Jul 2007     29.67     31.84    28.95      28.99  1295548000          28.68
Aug 2007     28.95     30.10    27.51      28.73  1228579500          28.52
Sep 2007     28.50     29.85    28.27      29.46  1117419500          29.25
Oct 2007     29.46     37.00    29.29      36.81  1771226400          36.55
Nov 2007     36.53     37.50    32.68      33.60  1829448700          33.47
Dec 2007     33.50     36.72    32.63      35.60  1064817100          35.46
Jan 2008     35.79     35.96    31.04      32.60  1950301600          32.47
Feb 2008     31.06     33.25    27.02      27.20  2323373900          27.20
Mar 2008     27.24     29.59    26.87      28.38  1451582800          28.38
Apr 2008     28.83     29.54    28.63      29.50    65774200          29.50
```

indexAt allows control of resultant index class and style. Possible option include startof, firstof, lastof, endof, yearmon, and yearqtr

to.period allows for fast changes to the periodicity of univariate and OHLC data.

Convert from one periodicity to another - e.g. daily to monthly, or hourly to daily with one function.

fast periodicity conversion

```
> to.monthly(MSFT)
         MSFT.Open MSFT.High MSFT.Low MSFT.Close MSFT.Volume MSFT.Adjusted
Jan 2007     29.91     31.48    29.40      30.86  1324518200          30.33
Feb 2007     30.84     30.94    27.79      28.17  1290850900          27.78
Mar 2007     27.82     28.55    26.60      27.87  1269506500          27.48
Apr 2007     27.89     30.74    27.56      29.94   958964900          29.53
May 2007     29.94     31.16    29.90      30.69  1327154700          30.36
Jun 2007     30.79     30.90    29.04      29.47  1181412800          29.16
Jul 2007     29.67     31.84    28.95      28.99  1295548000          28.68
Aug 2007     28.95     30.10    27.51      28.73  1228579500          28.52
Sep 2007     28.50     29.85    28.27      29.46  1117419500          29.25
Oct 2007     29.46     37.00    29.29      36.81  1771226400          36.55
Nov 2007     36.53     37.50    32.68      33.60  1829448700          33.47
Dec 2007     33.50     36.72    32.63      35.60  1064817100          35.46
Jan 2008     35.79     35.96    31.04      32.60  1950301600          32.47
Feb 2008     31.06     33.25    27.02      27.20  2323373900          27.20
Mar 2008     27.24     29.59    26.87      28.38  1451582800          28.38
Apr 2008     28.83     29.54    28.63      29.50    65774200          29.50
```

indexAt allows control of resultant index class and style. Possible option include startof, firstof, lastof, endof, yearmon, and yearqtr

to.period allows for fast changes to the periodicity of univariate and OHLC data.

Convert from one periodicity to another - e.g. daily to monthly, or hourly to daily with one function.

Additional Wrappers:

to.minutes
to.hourly
to.daily
to.weekly
to.monthly
to.quarterly
to.yearly

Aggregate by period
period.apply

Apply *any* function on specified periods.

```
> apply.monthly(MSFT, function(x) { mean(Hi(x)) })
2007-01-31 2007-02-28 2007-03-30 2007-04-30 2007-05-31 2007-06-29 2007-07-31
  30.81300   29.49474   27.92318   28.91900   30.91636   30.35095   30.43667
2007-08-31 2007-09-28 2007-10-31 2007-11-30 2007-12-31 2008-01-31 2008-02-29
  29.00435   29.13105   31.50435   35.00619   35.31400   34.02238   29.07050
2008-03-31 2008-04-01
  28.75650   29.54000
```

apply.monthly(MSFT, function(x) mean(Hi(x)))
returns the monthly average Hi for MSFT

Aggregate by period
period.apply

Apply *any* function on specified periods.

```
> apply.monthly(MSFT, function(x) { mean(Hi(x)) })
2007-01-31 2007-02-28 2007-03-30 2007-04-30 2007-05-31 2007-06-29 2007-07-31
  30.81300   29.49474   27.92318   28.91900   30.91636   30.35095   30.43667
2007-08-31 2007-09-28 2007-10-31 2007-11-30 2007-12-31 2008-01-31 2008-02-29
  29.00435   29.13105   31.50435   35.00619   35.31400   34.02238   29.07050
2008-03-31 2008-04-01
  28.75650   29.54000
```

apply.monthly(MSFT, function(x) mean(Hi(x)))
returns the monthly average Hi for MSFT

Additional Wrappers: apply.daily, apply.weekly, apply.monthly, apply.quarterly, apply.yearly

reclass and Reclass

Return data converted to xts with as.xts back to its original class.

reclass and Reclass

```
> str(timeSeries(MSFT, rownames(MSFT), title='Meielisalp2008'))
Formal class 'timeSeries' [package "fSeries"] with 8 slots
  ..@ Data        : num [1:313, 1:6] 29.9 29.7 29.6 29.6 30.0 ...
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : atomic [1:313] 2007-01-03 2007-01-04 2007-01-05 2007-01-08 ...
  .. .. .. ..- attr(*, "control")= Named chr "GMT"
  .. .. .. .. ..- attr(*, "names")= chr "FinCenter"
  .. .. ..$ : chr [1:6] "MSFT.Open" "MSFT.High" "MSFT.Low" "MSFT.Close" ...
  ..@ positions   : atomic [1:313] 2007-01-03 2007-01-04 2007-01-05 2007-01-08 ...
  .. ..- attr(*, "control")= Named chr "GMT"
  .. .. ..- attr(*, "names")= chr "FinCenter"
  ..@ format      : chr "%Y-%m-%d"
  ..@ FinCenter   : chr "GMT"
  ..@ units       : chr [1:6] "MSFT.Open" "MSFT.High" "MSFT.Low" "MSFT.Close" ...
  ..@ recordIDs   :'data.frame':      0 obs. of  0 variables
  ..@ title       : chr "Meielisalp2008"
  ..@ documentation: chr "Mon Jun 23 19:29:22 2008"
> 
```

Start with a **timeSeries**

Return data converted to xts with as.xts back to its original class.

reclass and Reclass

```
> str(timeSeries(MSFT, rownames(MSFT), title='Meielisalp2008'))
Formal class 'timeSeries' [package "fSeries"] with 8 slots
  ..@ Data        : num [1:313, 1:6] 29.9 29.7 29.6 29.6 30.0 ...
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : atomic [1:313] 2007-01-03 2007-01-04 2007-01-05 2007-01-08 ...
  .. .. .. ..- attr(*, "control")= Named chr "GMT"
  .. .. .. .. ..- attr(*, "names")= chr "FinCenter"
  .. .. ..$ : chr [1:6] "MSFT.Open" "MSFT.High" "MSFT.Low" "MSFT.Close" ...
  ..@ positions   : atomic [1:313] 2007-01-03 2007-01-04 2007-01-05 2007-01-08 ...
  .. ..- attr(*, "control")= Named chr "GMT"
  .. .. ..- attr(*, "names")= chr "FinCenter"
  ..@ format      : chr "%Y-%m-%d"
  ..@ FinCenter   : chr "GMT"
  ..@ units       : chr [1:6] "MSFT.Open" "MSFT.High" "MSFT.Low" "MSFT.Close" ...
  ..@ recordIDs   :'data.frame':      0 obs. of  0 variables
  ..@ title       : chr "Meielisalp2008"
  ..@ documentation: chr "Mon Jun 23 19:29:22 2008"
>
```

Start with a **timeSeries**

Return data converted to xts with as.xts back to its original class.

```
> str(x <- as.xts(timeSeries(MSFT, rownames(MSFT), title='Meielisalp2008'), by="JA Ryan"))
An 'xts' object from 2007-01-03 to 2008-04-01 containing:
  Data: num [1:313, 1:6] 29.9 29.7 29.6 29.6 30.0 ...
 - attr(*, "dimnames")=List of 2
  ..$ : chr [1:313] "2007-01-03" "2007-01-04" "2007-01-05" "2007-01-08" ...
  ..$ : chr [1:6] "MSFT.Open" "MSFT.High" "MSFT.Low" "MSFT.Close" ...
  Indexed by:  POSIXct[1:313], format: "2007-01-03" "2007-01-04" "2007-01-05" "2007-01-08" ...
  Original class: 'timeSeries'
  xts Attributes:
List of 6
 $ format       : chr "%Y-%m-%d"
 $ FinCenter    : chr "GMT"
 $ recordIDs    :'data.frame':  0 obs. of  0 variables
 $ title        : chr "Meielisalp2008"
 $ documentation: chr "Mon Jun 23 19:42:59 2008"
 $ by           : chr "JA Ryan"
>
```

Convert to **xts**

reclass and Reclass

```
> str(timeSeries(MSFT, rownames(MSFT), title='Meielisalp2008'))
Formal class 'timeSeries' [package "fSeries"] with 8 slots
  ..@ Data         : num [1:313, 1:6] 29.9 29.7 29.6 29.6 30.0 ...
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : atomic [1:313] 2007-01-03 2007-01-04 2007-01-05 2007-01-08 ...
  .. .. .. ..- attr(*, "control")= Named chr "GMT"
  .. .. .. .. ..- attr(*, "names")= chr "FinCenter"
  .. .. ..$ : chr [1:6] "MSFT.Open" "MSFT.High" "MSFT.Low" "MSFT.Close" ...
  ..@ positions    : atomic [1:313] 2007-01-03 2007-01-04 2007-01-05 2007-01-08 ...
  .. ..- attr(*, "control")= Named chr "GMT"
  .. .. ..- attr(*, "names")= chr "FinCenter"
  ..@ format       : chr "%Y-%m-%d"
  ..@ FinCenter    : chr "GMT"
  ..@ units        : chr [1:6] "MSFT.Open" "MSFT.High" "MSFT.Low" "MSFT.Close" ...
  ..@ recordIDs    :'data.frame':       0 obs. of  0 variables
  ..@ title        : chr "Meielisalp2008"
  ..@ documentation: chr "Mon Jun 23 19:29:22 2008"
>
```

Start with a **timeSeries**

Return data converted to xts with as.xts back to its original class.

```
> str(x <- as.xts(timeSeries(MSFT, rownames(MSFT), title='Meielisalp2008'), by="JA Ryan"))
An 'xts' object from 2007-01-03 to 2008-04-01 containing:
  Data: num [1:313, 1:6] 29.9 29.7 29.6 29.6 30.0 ...
 - attr(*, "dimnames")=List of 2
  ..$ : chr [1:313] "2007-01-03" "2007-01-04" "2007-01-05" "2007-01-08" ...
  ..$ : chr [1:6] "MSFT.Open" "MSFT.High" "MSFT.Low" "MSFT.Close" ...
  Indexed by:  POSIXct[1:313], format: "2007-01-03" "2007-01-04" "2007-01-05" "2007-01-08" ...
  Original class: 'timeSeries'
  xts Attributes:
List of 6
 $ format       : chr "%Y-%m-%d"
 $ FinCenter    : chr "GMT"
 $ recordIDs    :'data.frame':  0 obs. of  0 variables
 $ title        : chr "Meielisalp2008"
 $ documentation: chr "Mon Jun 23 19:42:59 2008"
 $ by           : chr "JA Ryan"
>
```

Convert to **xts**

xtsAttributes store meta-data

Convert back without losing information using reclass

```
> str(reclass(x))
Formal class 'timeSeries' [package "fSeries"] with 8 slots
  ..@ Data         : num [1:313, 1:6] 29.9 29.7 29.6 29.6 30.0 ...
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : atomic [1:313] 2007-01-03 2007-01-04 2007-01-05 2007-01-08 ...
  .. .. .. ..- attr(*, "control")= Named chr "GMT"
  .. .. .. .. ..- attr(*, "names")= chr "FinCenter"
  .. .. ..$ : chr [1:6] "MSFT.Open" "MSFT.High" "MSFT.Low" "MSFT.Close" ...
  ..@ positions    : atomic [1:313] 2007-01-03 2007-01-04 2007-01-05 2007-01-08 ...
  .. ..- attr(*, "control")= Named chr "GMT"
  .. .. ..- attr(*, "names")= chr "FinCenter"
  ..@ format       : chr "%Y-%m-%d"
  ..@ FinCenter    : chr "GMT"
  ..@ units        : chr [1:6] "MSFT.Open" "MSFT.High" "MSFT.Low" "MSFT.Close" ...
  ..@ recordIDs    :'data.frame':      0 obs. of  0 variables
  ..@ title        : chr "Meielisalp2008"
  ..@ documentation: chr "Mon Jun 23 19:42:59 2008"
>
```

converting back to *timeSeries* from *xts* maintains the original attributes, e.g. @title and @documentation

as.xts and reclass example

```
> getSymbols("SBUX",return='timeSeries')
[1] "SBUX"
>
```

```
> is.timeSeries(SBUX)
[1] TRUE
>
```

download a timeSeries of SBUX

as.xts and reclass example

```
> SBUX['2008-06']
Error in `[.timeSeries`(SBUX, "2008-06") :
subscript out of bounds
```

attempting to subset xts-style results in an error

as.xts and reclass example

the solution?

```
> x <- as.xts(SBUX)['2008-06']
> x
           SBUX.Open SBUX.High SBUX.Low SBUX.Close SBUX.Volume SBUX.Adjusted
2008-06-02     18.17     18.17    17.63       17.93    10877200          17.93
2008-06-03     17.93     18.13    17.40       17.75    13076200          17.75
2008-06-04     17.72     18.34    17.72       18.12    15915400          18.12
2008-06-05     18.15     18.52    18.01       18.52    13662400          18.52
2008-06-06     18.35     18.35    17.64       17.67    12236500          17.67
2008-06-09     17.71     18.20    17.37       17.52    16788400          17.52
2008-06-10     17.33     17.93    17.25       17.84     9761300          17.84
2008-06-11     17.94     17.94    17.55       17.57    11351100          17.57
2008-06-12     17.76     18.06    17.66       17.82     8571900          17.82
2008-06-13     17.95     18.20    17.83       18.17    10910400          18.17
2008-06-16     18.01     18.56    17.89       18.35     8295000          18.35
2008-06-17     18.37     18.47    18.07       18.12     6369500          18.12
2008-06-18     18.00     18.09    17.66       17.77     8452600          17.77
2008-06-19     17.73     18.06    17.38       17.99     9215800          17.99
2008-06-20     17.77     17.82    17.18       17.23    13066200          17.23
2008-06-23     17.26     17.49    16.27       16.30    19602100          16.30
2008-06-24     16.36     16.92    16.25       16.57    12763800          16.57
```

first convert with as.xts

as.xts and reclass example

then reclass back to timeSeries

```
> is.timeSeries(x)
[1] FALSE
> x <- reclass(x)
> is.timeSeries(x)
[1] TRUE
> x
           SBUX.Open SBUX.High SBUX.Low SBUX.Close SBUX.Volume SBUX.Adjusted
2008-06-02     18.17     18.17    17.63       17.93    10877200          17.93
2008-06-03     17.93     18.13    17.40       17.75    13076200          17.75
2008-06-04     17.72     18.34    17.72       18.12    15915400          18.12
2008-06-05     18.15     18.52    18.01       18.52    13662400          18.52
2008-06-06     18.35     18.35    17.64       17.67    12236500          17.67
2008-06-09     17.71     18.20    17.37       17.52    16788400          17.52
2008-06-10     17.33     17.93    17.25       17.84     9761300          17.84
2008-06-11     17.94     17.94    17.55       17.57    11351100          17.57
2008-06-12     17.76     18.06    17.66       17.82     8571900          17.82
2008-06-13     17.95     18.20    17.83       18.17    10910400          18.17
2008-06-16     18.01     18.56    17.89       18.35     8295000          18.35
2008-06-17     18.37     18.47    18.07       18.12     6369500          18.12
2008-06-18     18.00     18.09    17.66       17.77     8452600          17.77
2008-06-19     17.73     18.06    17.38       17.99     9215800          17.99
2008-06-20     17.77     17.82    17.18       17.23    13066200          17.23
2008-06-23     17.26     17.49    16.27       16.30    19602100          16.30
2008-06-24     16.36     16.92    16.25       16.57    12763800          16.57
```

as.xts and reclass example

then reclass back to timeSeries

```
> is.timeSeries(x)
[1] FALSE
> x <- reclass(x)
> is.timeSeries(x)
[1] TRUE
> x
           SBUX.Open SBUX                                X.Volume SBUX.Adjusted
2008-06-02     18.17                                  10877200          17.93
2008-06-03     17.93                                  13076200          17.75
2008-06-04     17.72                                  15915400          18.12
2008-06-05     18.15                                  13662400          18.52
2008-06-06     18.35                                  12236500          17.67
2008-06-09     17.71                                  16788400          17.52
2008-06-10     17.33                                   9761300          17.84
2008-06-11     17.94                                  11351100          17.57
2008-06-12     17.76                                   8571900          17.82
2008-06-13     17.95                                  10910400          18.17
2008-06-16     18.01    18.56    17.89    18.35        8295000          18.35
2008-06-17     18.37    18.47    18.07    18.12        6369500          18.12
2008-06-18     18.00    18.09    17.66    17.77        8452600          17.77
2008-06-19     17.73    18.06    17.38    17.99        9215800          17.99
2008-06-20     17.77    17.82    17.18    17.23       13066200          17.23
2008-06-23     17.26    17.49    16.27    16.30       19602100          16.30
2008-06-24     16.36    16.92    16.25    16.57       12763800          16.57
```

xts-style subsetting on a timeSeries object!

What is Reclass for?

What is Reclass for?

Reclass attempts to take *any* function call and force it to return an object that matches the class of the object you passed in.

What is Reclass for?

EMA from TTR returns a vector

```
> EMA(Cl(MSFT))[1:20]
 [1]       NA       NA       NA       NA       NA       NA       NA       NA
 [9]       NA 30.30300 30.42973 30.55341 30.58370 30.61212 30.69901 30.65373
[17] 30.64396 30.62324 30.59720 30.64498
> 
```

What is Reclass for?

EMA from TTR returns a vector

```
> EMA(Cl(MSFT))[1:20]
 [1]       NA       NA       NA       NA       NA       NA       NA       NA
 [9]       NA 30.30300 30.42973 30.55341 30.58370 30.61212 30.69901 30.65373
[17] 30.64396 30.62324 30.59720 30.64498
>
```

Wrapped in Reclass, and the result is the original xts class

```
> Reclass(EMA(Cl(MSFT)))[1:20]

2007-01-03       NA
2007-01-04       NA
2007-01-05       NA
2007-01-08       NA
2007-01-09       NA
2007-01-10       NA
2007-01-11       NA
2007-01-12       NA
2007-01-16       NA
2007-01-17 30.30300
2007-01-18 30.42973
2007-01-19 30.55341
2007-01-22 30.58370
2007-01-23 30.61212
2007-01-24 30.69901
2007-01-25 30.65373
2007-01-26 30.64396
2007-01-29 30.62324
2007-01-30 30.59720
2007-01-31 30.64498
```

Developing with xts

Using xts internally to manage time-based data in your own functions and packages.

Developing with xts

Four (4) options for handling R's time-based classes in functions:

1) Write methods for all possible inputs

Developing with xts

Four (4) options for handling R's time-based classes in functions:

1) Write methods for all possible inputs (9 classes!)

Data Classes:
matrix,
data.frame, ts,
zoo, its, irts,
timeSeries, xts,
vectors

Time Classes:
POSIXct,
POSIXlt, chron,
Date, timeDate,
yearmon,
yearqtr

Developing with xts

Four (4) options for handling R's time-based classes in functions:

1) Write methods for all possible inputs
2) Choose one class to accept

Data Classes:
matrix, data.frame, ts, zoo, its, irts, timeSeries, xts, vectors

Time Classes:
POSIXct, POSIXlt, chron, Date, timeDate, yearmon, yearqtr

Developing with xts

Four (4) options for handling R's time-based classes in functions:

1) Write methods for all possible inputs
2) Choose one class to accept
3) Convert internally to ... (matrix?)

Data Classes:
matrix, data.frame, ts, zoo, its, irts, timeSeries, xts, vectors

Time Classes:
POSIXct, POSIXlt, chron, Date, timeDate, yearmon, yearqtr

Developing with xts

Four (4) options for handling R's time-based classes in functions:

1) Write methods for all possible inputs
2) Choose one class to accept
3) Convert internally to ... (matrix?)
4) Use xts functions to manage the process

Data Classes:
matrix, data.frame, ts, zoo, its, irts, timeSeries, xts, vectors

Time Classes:
POSIXct, POSIXlt, chron, Date, timeDate, yearmon, yearqtr

try.xts and reclass

Two simple functions let you...

- Accept all common time-series in R
- Manage one type of object internally
- Return original class to the user

try.xts and reclass

Two simple functions let you...

- Accept all common time-series in R
- Manage one type of object internally
- Return original class to the user

which translates to...

- Less code to write and maintain
- Increased reliability and flexibility
- Freedom to focus on core development tasks

try.xts

Convert, if possible, an incoming object to xts

Use is.xts to test and branch if desired

try.xts

Convert, if possible, an incoming object to xts

Use is.xts to test and branch if desired

reclass

Convert, if possible, back to the original class

... an example?

period.apply using try.xts & reclass

```
period.apply <-
function (x, INDEX, FUN, ...)
{
    x <- try.xts(x, error = FALSE)
    FUN <- match.fun(FUN)
    xx <- sapply(1:(length(INDEX) - 1), function(y) {
        FUN(x[(INDEX[y] + 1):INDEX[y + 1]], ...)
    })
    reclass(xx, x[INDEX])
}
```

period.apply using try.xts & reclass

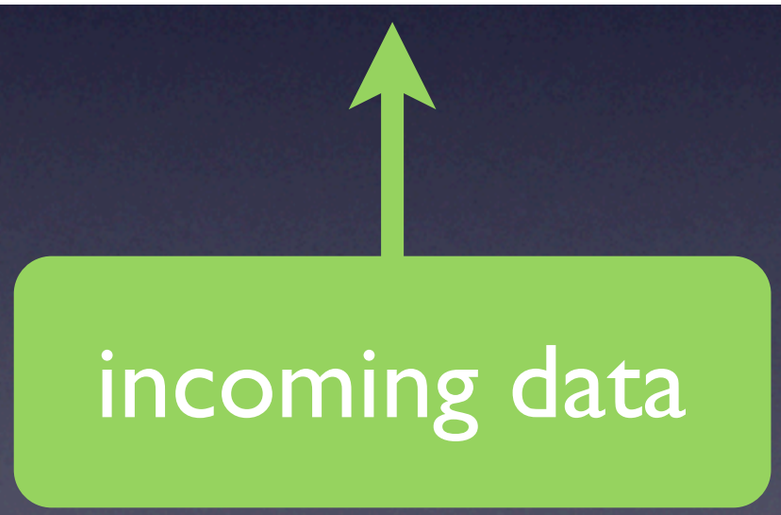
```
period.apply <-
function (x, INDEX, FUN, ...)
{
    x <- try.xts(x, error = FALSE)
    FUN <- match.fun(FUN)
    xx <- sapply(1:(length(INDEX) - 1), function(y) {
        FUN(x[(INDEX[y] + 1):INDEX[y + 1]], ...)
    })
    reclass(xx, x[INDEX])
}
```

Two simple additions allows the period.apply function to accept, and ultimately return, any class of time-series object. By setting error=FALSE, it is even possible to accept non-xts coercible args. Truly universal data acceptance.

Using try.xts

using the incoming data, attempt to convert

```
try.xts(x, ..., error = FALSE)
```



Using try.xts

add any additional args to xts constructor

```
try.xts(x, ..., error = FALSE)
```

passed to xts()

Using try.xts

FALSE means success isn't required

```
try.xts(x, ..., error = FALSE)
```

failure OK?

Using reclass

first argument - result of internal calculations

```
reclass(xx, match.to=x)
```

the result before converting

Using reclass

second argument - the original object (more or less)

```
reclass(xx, match.to=x)
```

the original result from try.xts

Using reclass

second argument - the original object (more or less)

```
reclass(xx, match.to=x)
```

the original result from try.xts

the match.to argument is the template for re-indexing as an xts object.

Put it all together

```
period.apply <-
function (x, INDEX, FUN, ...)
{
    x <- try.xts(x, error = FALSE)
    FUN <- match.fun(FUN)
    xx <- sapply(1:(length(INDEX) - 1), function(y) {
        FUN(x[(INDEX[y] + 1):INDEX[y + 1]], ...)
    })
    reclass(xx, x[INDEX])
}
```

Using try.xts and reclass

Put it all together

```
period.apply <-
function (x, INDEX, FUN, ...)
{
    x <- try.xts(x, error = FALSE)
    FUN <- match.fun(FUN)
    xx <- sapply(1:(length(INDEX) - 1), function(y) {
        FUN(x[(INDEX[y] + 1):INDEX[y + 1]], ...)
    })
    reclass(xx, x[INDEX])
}
```

try.xts

Attempt to convert to xts, if not possible - continue on.
In this case, it isn't necessary that we have an xts object,
it is only to provide the user with a seemless experience

Put it all together

```
period.apply <-
function (x, INDEX, FUN, ...)
{
    x <- try.xts(x, error = FALSE)
    FUN <- match.fun(FUN)
    xx <- sapply(1:(length(INDEX) - 1), function(y) {
        FUN(x[(INDEX[y] + 1):INDEX[y + 1]], ...)
    })
    reclass(xx, x[INDEX])
}
```

main calculations

Proceed with function work. It is important to keep the original x variable untouched, otherwise the data may be lost or corrupted by non-xts aware functions

Put it all together

```
period.apply <-
function (x, INDEX, FUN, ...)
{
    x <- try.xts(x, error = FALSE)
    FUN <- match.fun(FUN)
    xx <- sapply(1:(length(INDEX) - 1), function(y) {
        FUN(x[(INDEX[y] + 1):INDEX[y + 1]], ...)
    })
    reclass(xx, x[INDEX])
}
```

reclass

xx is the result of the function call, before attempting the reclass. In this case the data is shorter than the original, and must be modified for reclass to work correctly.

xts Summary

xts Summary

- zoo modified for time

xts Summary

- zoo modified for time
- new time-aware tools

xts Summary

- zoo modified for time
- new time-aware tools
- increased developer productivity

xts Summary

- zoo modified for time
- new time-aware tools
- increased developer productivity

Now on to quantmod and charts!

quantmod

Jeffrey A. Ryan

www.quantmod.com

quantmod.r-forge.r-project.org

Purpose

Provide a single, unified R-based workflow.

Purpose

Provide a single, unified R-based workflow.

Data (getSymbols)

Purpose

Provide a single, unified R-based workflow.

Data (getSymbols)

Visuals (chartSeries)

Purpose

Provide a single, unified R-based workflow.

Data (getSymbols)

Visuals (chartSeries)

Model (buildModel)

Purpose

Provide a single, unified R-based workflow.

Data (getSymbols)

Visuals (chartSeries)

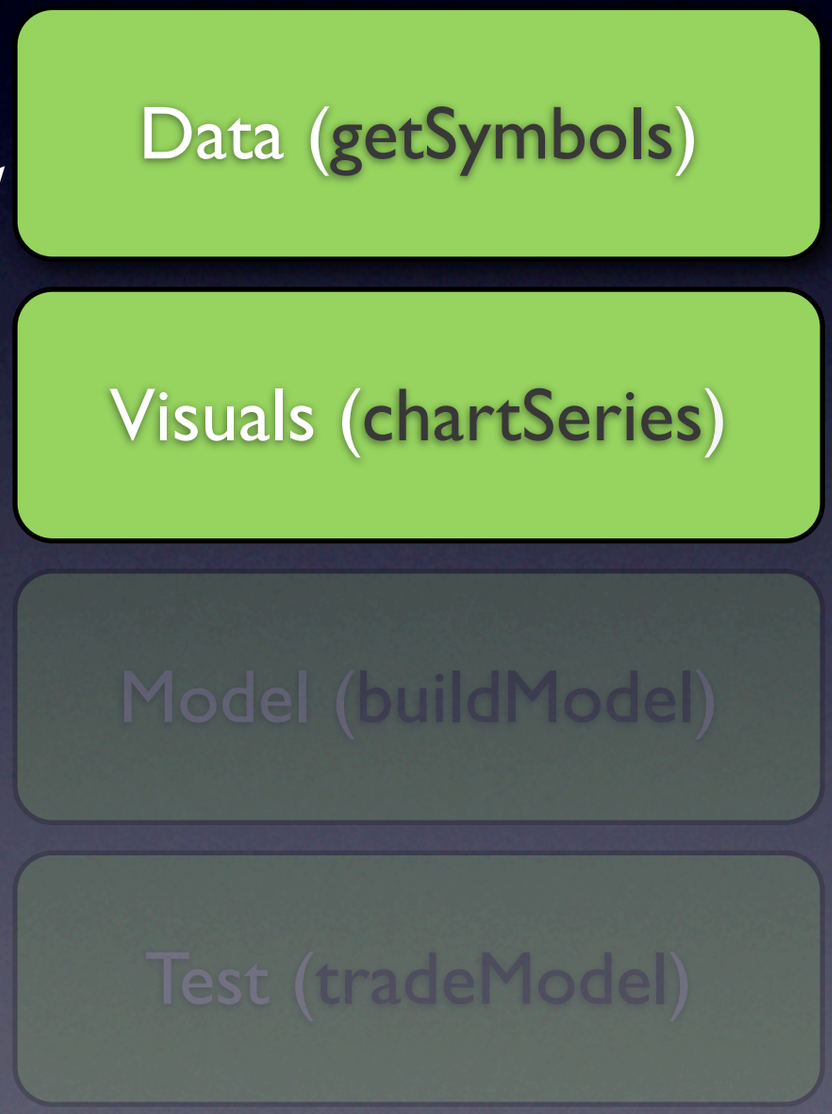
Model (buildModel)

Test (tradeModel)

Purpose

Provide a single, unified R-based workflow.

Today's presentation



Single Data Interface

getSymbols

- One wrapper function for all data sources
- Extensible by simple naming convention
- auto.assign into specified environment
- Settable return class

getSymbols

oanda	FRED (Federal Reserve Bank of St. Louis)	RData/rda
Yahoo! Finance	Google Finance	MySQL
SQLite	csv	IBrokers

getSymbols

getSymbols("SBUX", src = 'yahoo')

getSymbols.yahoo

getSymbols

getSymbols("SBUX", src = 'yahoo')

getSymbols.yahoo

Simple to extend to other data sources

Example 1

Download data from Yahoo!

Specify src = “yahoo” to getSymbols call

```
> getSymbols("YHOO", src="yahoo")
[1] "YHOO"
> head(YHOO)
           YHOO.Open YHOO.High YHOO.Low YHOO.Close YHOO.Volume YHOO.Adjusted
2007-01-03     25.85     26.26    25.26      25.61    26352700         25.61
2007-01-04     25.64     26.92    25.52      26.85    32512200         26.85
2007-01-05     26.70     27.87    26.66      27.74    64264600         27.74
2007-01-08     27.70     28.04    27.43      27.92    25713700         27.92
2007-01-09     28.00     28.05    27.41      27.58    25621500         27.58
2007-01-10     27.48     28.92    27.44      28.70    40240000         28.70
>
```

Example 1

Download data from Yahoo!

Specify src = "yahoo" to getSymbols call

```
> getSymbols("YHOO", src="yahoo")
[1] "YHOO"
> head(YHOO)
           YHOO.Open YHOO.High YHOO.Low YHOO.Close YHOO.Volume YHOO.Adjusted
2007-01-03     25.85     26.26    25.26      25.61    26352700         25.61
2007-01-04     25.64     26.92    25.52      26.85    32512200         26.85
2007-01-05     26.70     27.87    26.66      27.74    64264600         27.74
2007-01-08     27.70     28.04    27.43      27.92    25713700         27.92
2007-01-09     28.00     28.05    27.41      27.58    25621500         27.58
2007-01-10     27.48     28.92    27.44      28.70    40240000         28.70
> 
```

showSymbols returns information on what has been loaded

```
> showSymbols()
   YHOO
"yahoo"
> 
```

Example 11

Download from FRED and Google in one call

Use setSymbolLookup to change default source for certain symbols

```
> setSymbolLookup(DEXJPUS=list(src='FRED'),YHOO=list(src='google'))
> getSymbols("DEXJPUS;YHOO")
[1] "DEXJPUS" "YHOO"
>
```

Example 11

Download from FRED and Google in one call

Use setSymbolLookup to change default source for certain symbols

```
> setSymbolLookup(DEXJPUS=list(src='FRED'),YHOO=list(src='google'))
> getSymbols("DEXJPUS;YHOO")
[1] "DEXJPUS" "YHOO"
>
```

loads...

JPY/USD

```
> head(DEXJPUS)
           DEXJPUS
1971-01-04  357.73
1971-01-05  357.81
1971-01-06  357.86
1971-01-07  357.87
1971-01-08  357.82
1971-01-11  357.95
>
```

from Federal Reserve Bank of St. Louis FRED

YHOO

```
> head(YHOO)
           YHOO.Open YHOO.High YHOO.Low YHOO.Close YHOO.Volume
2007-01-03     25.85     26.26    25.26      25.61    26352700
2007-01-04     25.64     26.92    25.52      26.85    32512200
2007-01-05     26.70     27.87    26.66      27.74    64264600
2007-01-08     27.70     28.04    27.43      27.92    25713700
2007-01-09     28.00     28.05    27.41      27.58    25621500
2007-01-10     27.48     28.92    27.44      28.70    40240000
>
```

from Google Finance

Additional Data Tools

Additional Data Tools

- getQuote

  get highly configurable quotes from Yahoo! and others

Additional Data Tools

- getQuote

  get highly configurable quotes from Yahoo! and others

- getFinancials

  retrieve fundamental data from Google/Reuters

Additional Data Tools

- getQuote

  get highly configurable quotes from Yahoo! and others

- getFinancials

  retrieve fundamental data from Google/Reuters

- getFX & getMetals

  Get currency and metal prices from Oanda.com

Additional Data Tools

- getQuote

  get highly configurable quotes from Yahoo! and others

- getFinancials

  retrieve fundamental data from Google/Reuters

- getFX & getMetals

  Get currency and metal prices from Oanda.com

- getDividends

  Get dividend data from Yahoo!

Additional Data Tools

- getQuote

  get highly configurable quotes from Yahoo! and others

- getFinancials

  retrieve fundamental data from Google/Reuters

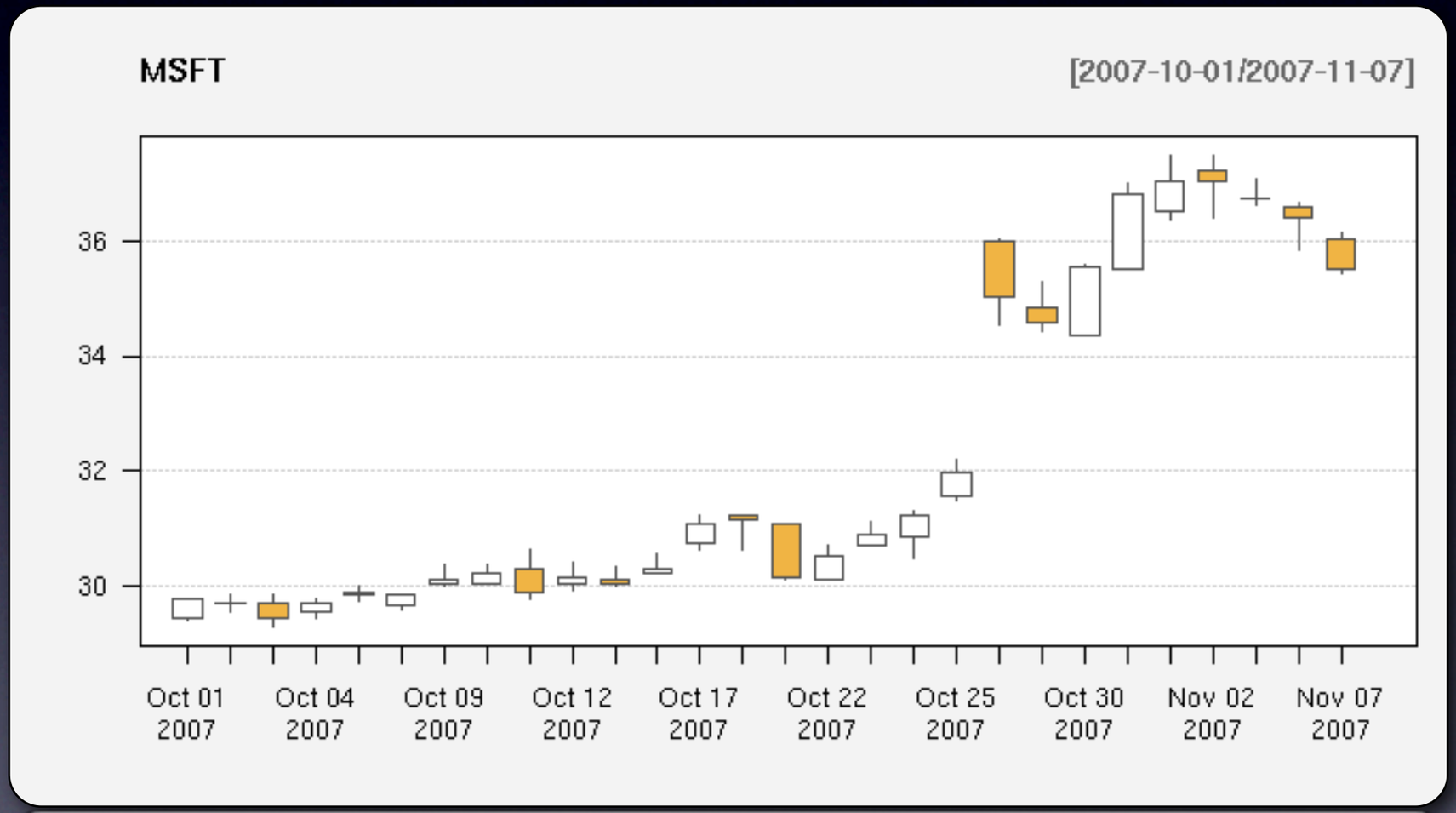
- getFX & getMetals

  Get currency and metal prices from Oanda.com

Now, on to the charts...

Visualization

chartSeries



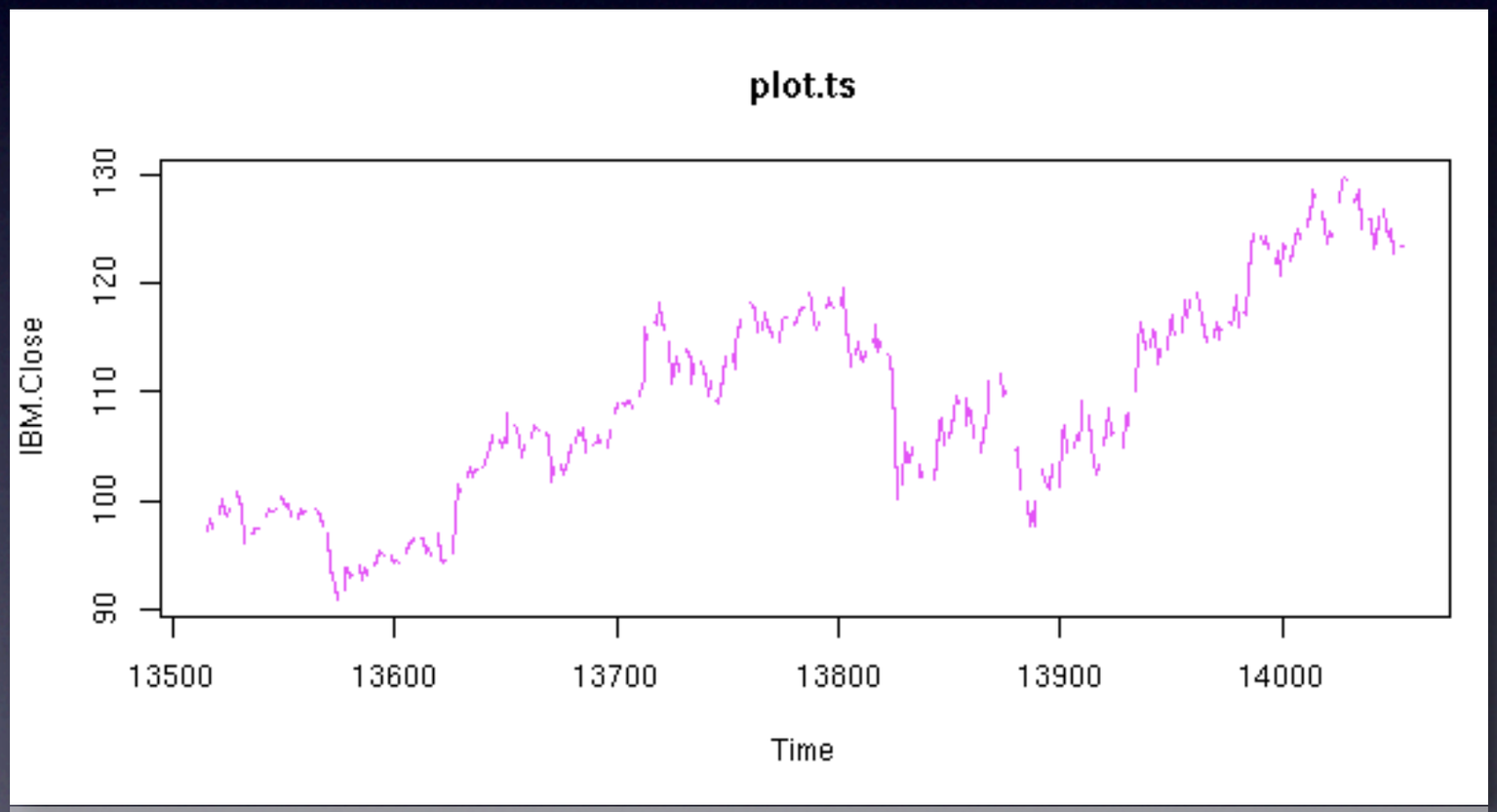
Charting in R

Charting in R

Most time-series plotting in R is derived from standard line charts

Charting in R

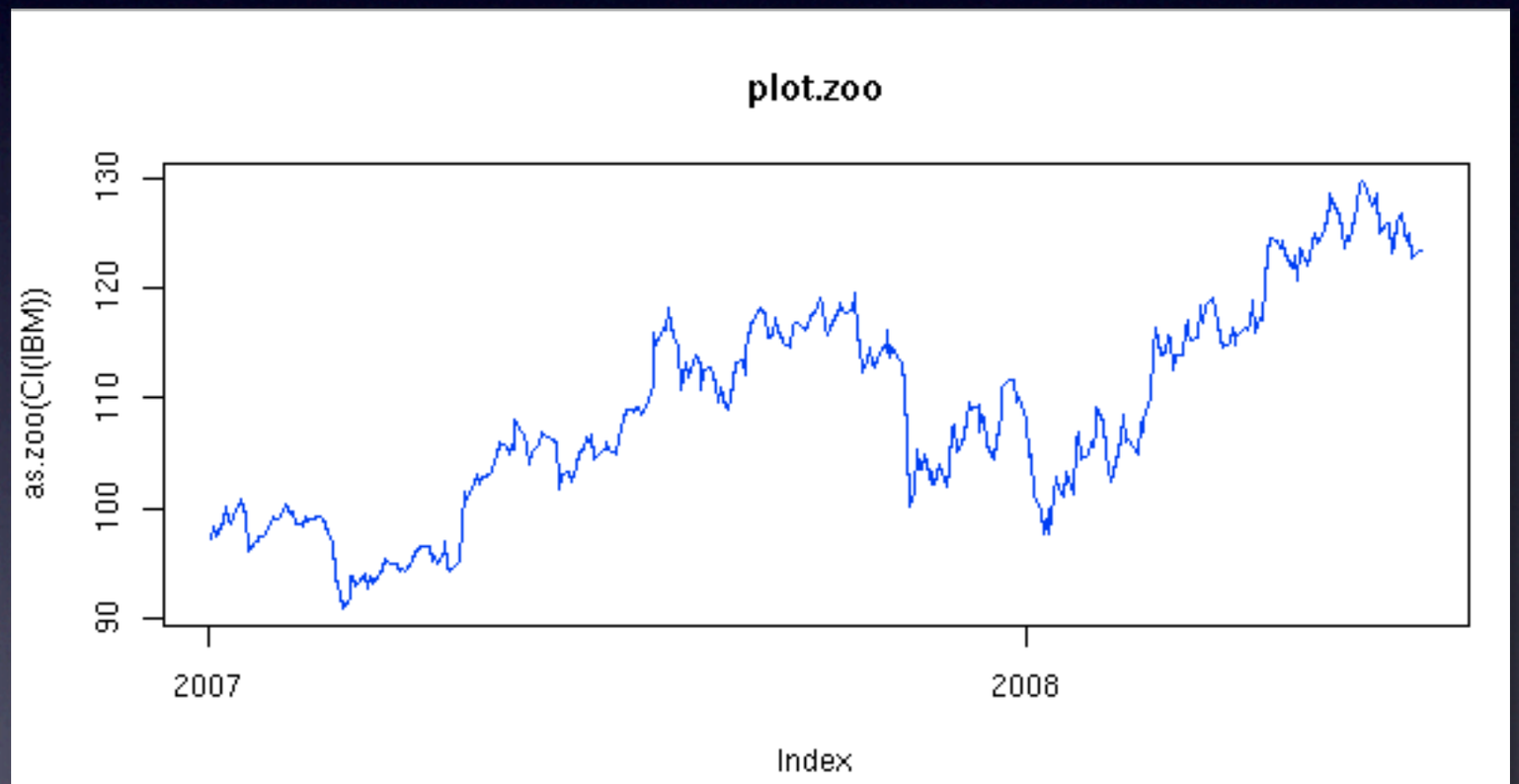
Most time-series plotting in R is derived from standard line charts



from stats

Charting in R

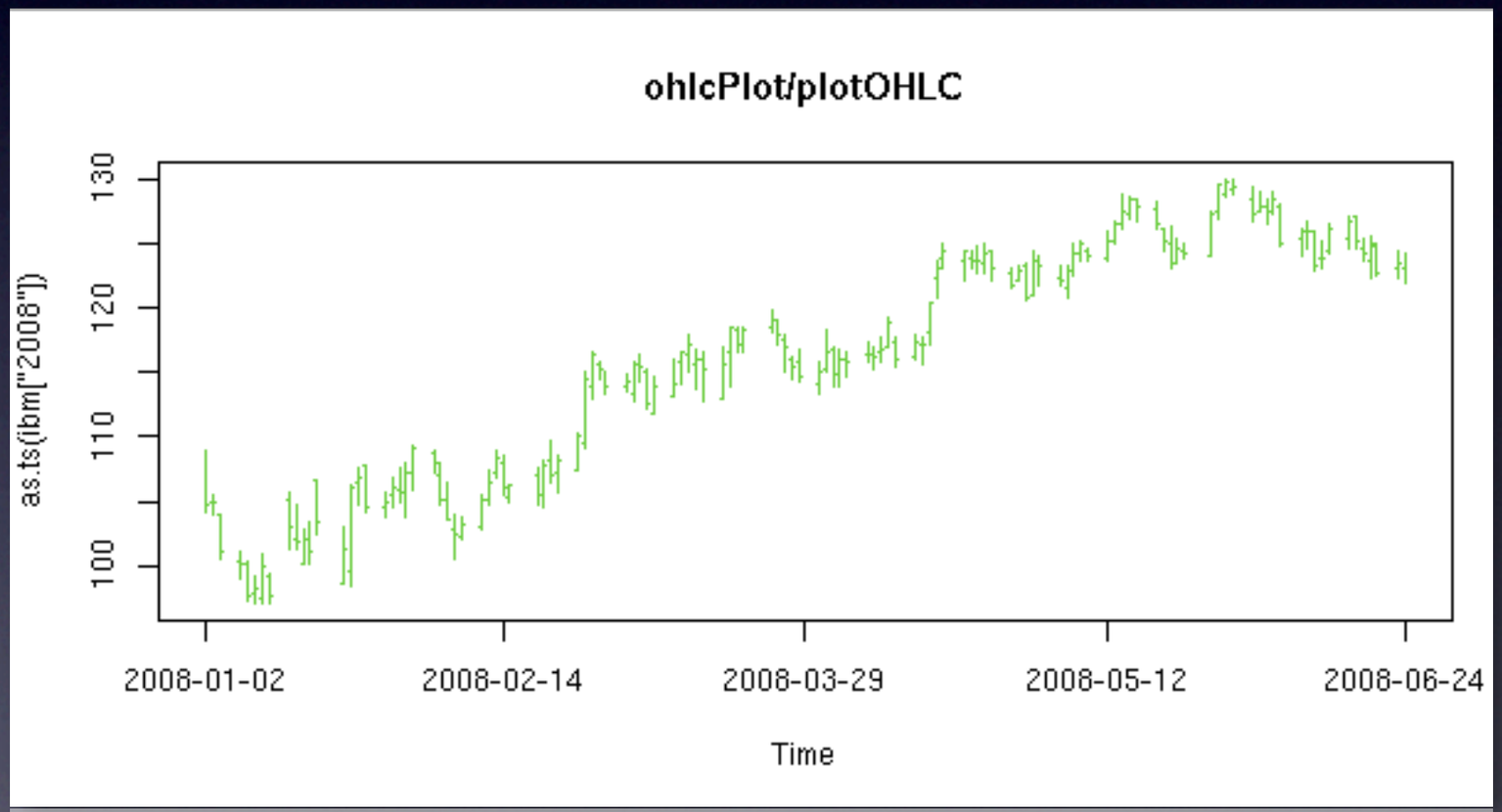
Most time-series plotting in R is derived from standard line charts



from zoo

Charting in R

Most time-series plotting in R is derived from standard line charts



from tseries and fTrading

Charting in R

Most time-series plotting in R is derived from standard line charts

from xts

Charting in R

Most time-series plotting in R is derived from standard line charts

All useful, but not really *financial* charts

from xts

Charting in R

An example posted to addictedtor.free.fr by Dirk Eddelbuettel shows what is possible with some 200+ lines of code...

Charting in R

Charting in R

chartSeries makes it easier...

chartSeries(IBM, TA='addBBands();addBBands(draw="p");addVo()',subset='2008')

1 line of code

```
chartSeries(IBM,TA='addBBands();addBBands(draw="p");addVo()',subset='2008')
```

chartSeries

- Overview

chartSeries

- Overview
- Customizing the look
  chartTheme and reChart

chartSeries

- Overview
- Customizing the look
  chartTheme and reChart
- zoomChart and zooom

chartSeries

- Overview
- Customizing the look
  chartTheme and reChart
- zoomChart and zooom
- Adding TA indicators
  Built-in
  ad-hoc additions with addTA
  custom indicators with newTA

Overview

Overview

- Works interactively or from scripts

Overview

- Works interactively or from scripts
- Built-in facility for bars, candles, and lines

Overview

- Works interactively or from scripts
- Built-in facility for bars, candles, and lines
- Manages layout dynamically

Overview

- Works interactively or from scripts
- Built-in facility for bars, candles, and lines
- Manages layout dynamically
- Add and remove elements at will

Overview

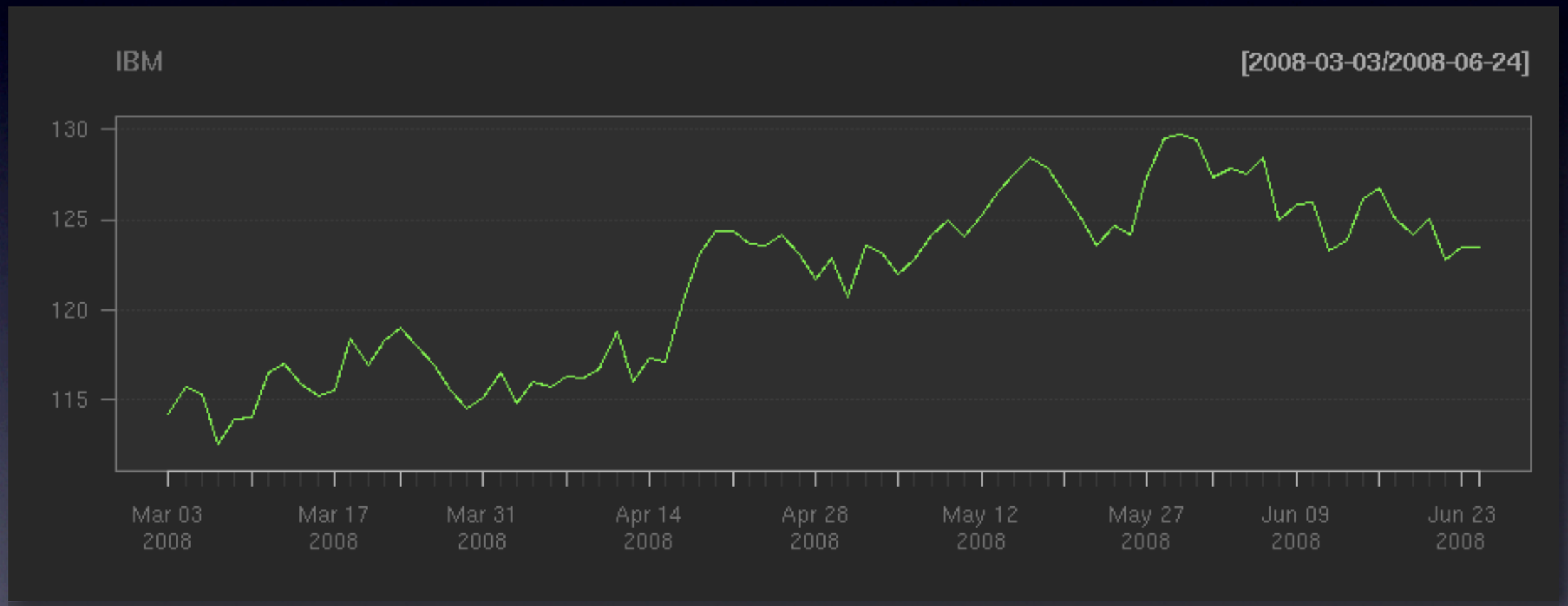
- Works interactively or from scripts
- Built-in facility for bars, candles, and lines
- Manages layout dynamically
- Add and remove elements at will
- Highly configurable

Overview

- Works interactively or from scripts
- Built-in facility for bars, candles, and lines
- Manages layout dynamically
- Add and remove elements at will
- Highly configurable
- Fully extensible

chartSeries can draw 3 styles of charts

lines



lineChart(IBM, subset='last 4 months', TA=NULL)

chartSeries can draw 3 styles of charts

ohlc & hlc bars

barChart(IBM, subset='last 4 months',TA=NULL)

chartSeries can draw 3 styles of charts

candlesticks

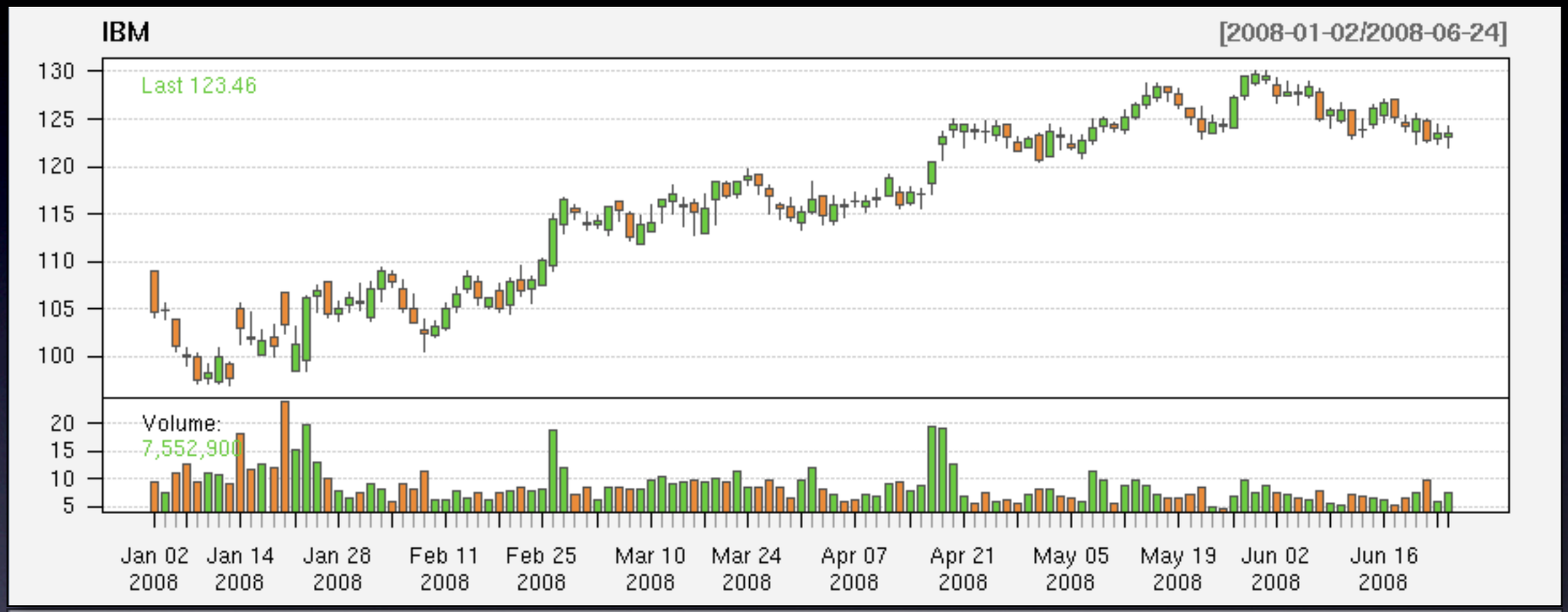
candleChart(IBM, subset='last 4 months', TA=NULL)

chartTheme

Charts use “themes” to coordinate colors

using the theme argument to chartSeries

theme = chartTheme("white")



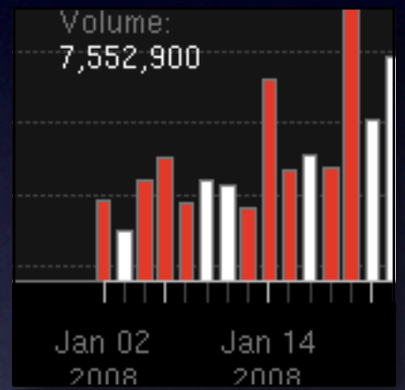
theme = chartTheme("black")

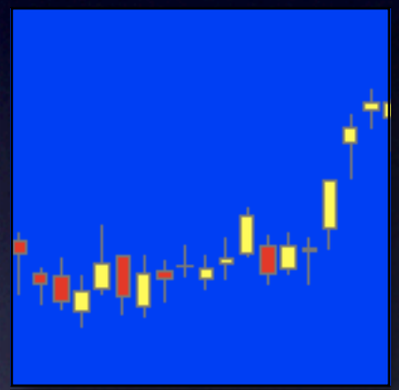
theme = chartTheme("white.mono")

theme = chartTheme("black.mono")

theme = chartTheme("beige")

Easy to modify themes





```
> # create a new theme
> blackandred <- chartTheme(up.col='white',dn.col='red',
+ area='#080808',bg.col='#000000')
>
> candleChart(IBM, theme=blackandred, subset='2008')
```

```
> # create a new theme
> blackandred <- chartTheme(up.col='white',dn.col='red',
+ area='#080808',bg.col='#000000')
>
> candleChart(IBM, theme=blackandred, subset='2008')
```

Modify drawn charts with reChart

Modify drawn charts with reChart


```
> reChart(theme='white', type='bars')
```

Modify drawn charts with reChart


```
> reChart(theme='white', type='bars')
```


Works for most chartSeries args!

zoomChart

(and zooom)

zoomChart

(and zooom)

Functions to zoom-in and zoom-out of a chart

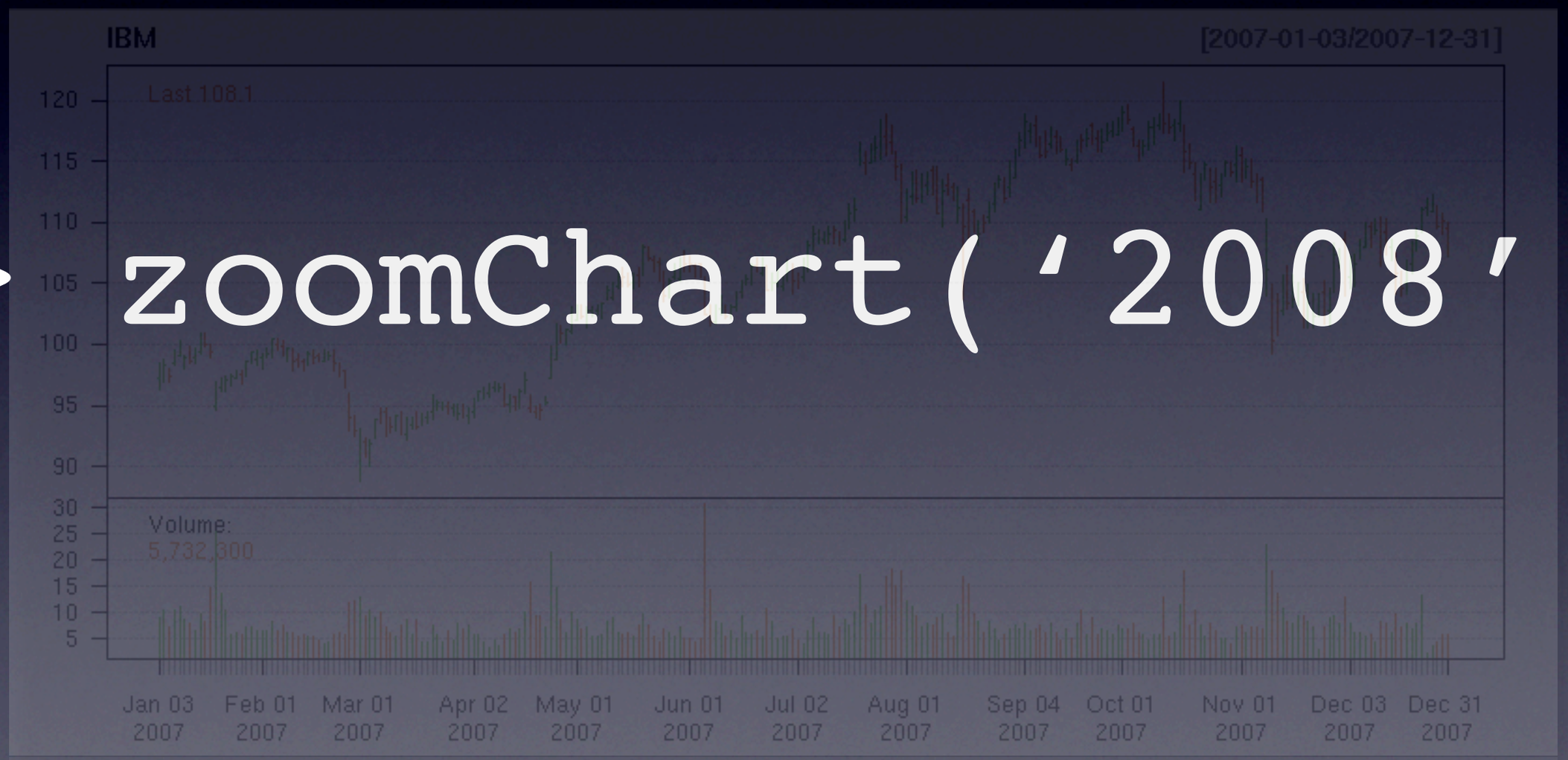
zoomChart

(and zooom)

all of 2007

zoomChart

(and zooom)



> zoomChart('2008')

zoomChart

(and zooom)

now 2008!

zoomChart

(and zooom)

now 2008!

Works just like subsetting in xts

Chart Additions

Adding data to charts is easy and fast

Chart Additions

Adding data to charts is easy and fast

Using the TA argument to chartSeries

Chart Additions

Adding data to charts is easy and fast

Using the TA argument to chartSeries

Interactively with addTA and friends

Adding with TA=

TA stands for (T)echnical (A)analysis

Built-in TA functionality from quantmod and TTR

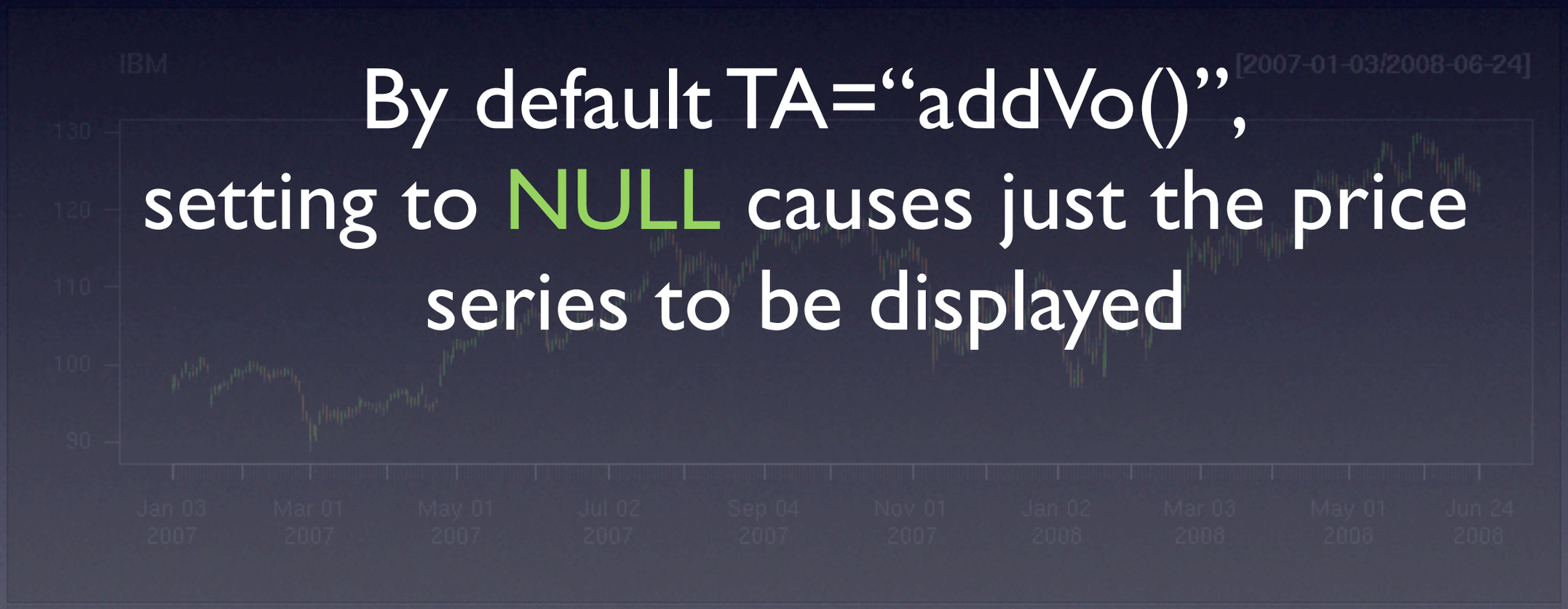
addADX	addATR	addBBands
addCCI	addCMF	addCMO
addDEMA	addDPO	addEMA
addEnvelope	addEVWMA	addExpiry
addMACD	addMomentum	addROC
addRSI	addSAR	addSMI
addTRIX	addVo	addWMA
addWPR	addZLEMA	more to come!!!!

Adding with TA=

chartSeries(IBM, TA=NULL)

Adding with TA=

chartSeries(IBM, TA=NULL)



By default TA="addVo()",
setting to NULL causes just the price series to be displayed

Adding with TA=

chartSeries(IBM, TA = “addMACD();addBBands()”)

Adding with TA=

chartSeries(IBM, TA = "addMACD();addBBands()")

Adding with TA=

chartSeries(IBM, TA = “addMACD();addBBands()”)

> zoomChart(‘2008’)

Adding with TA=

chartSeries(IBM, TA = "addMACD();addBBands()")

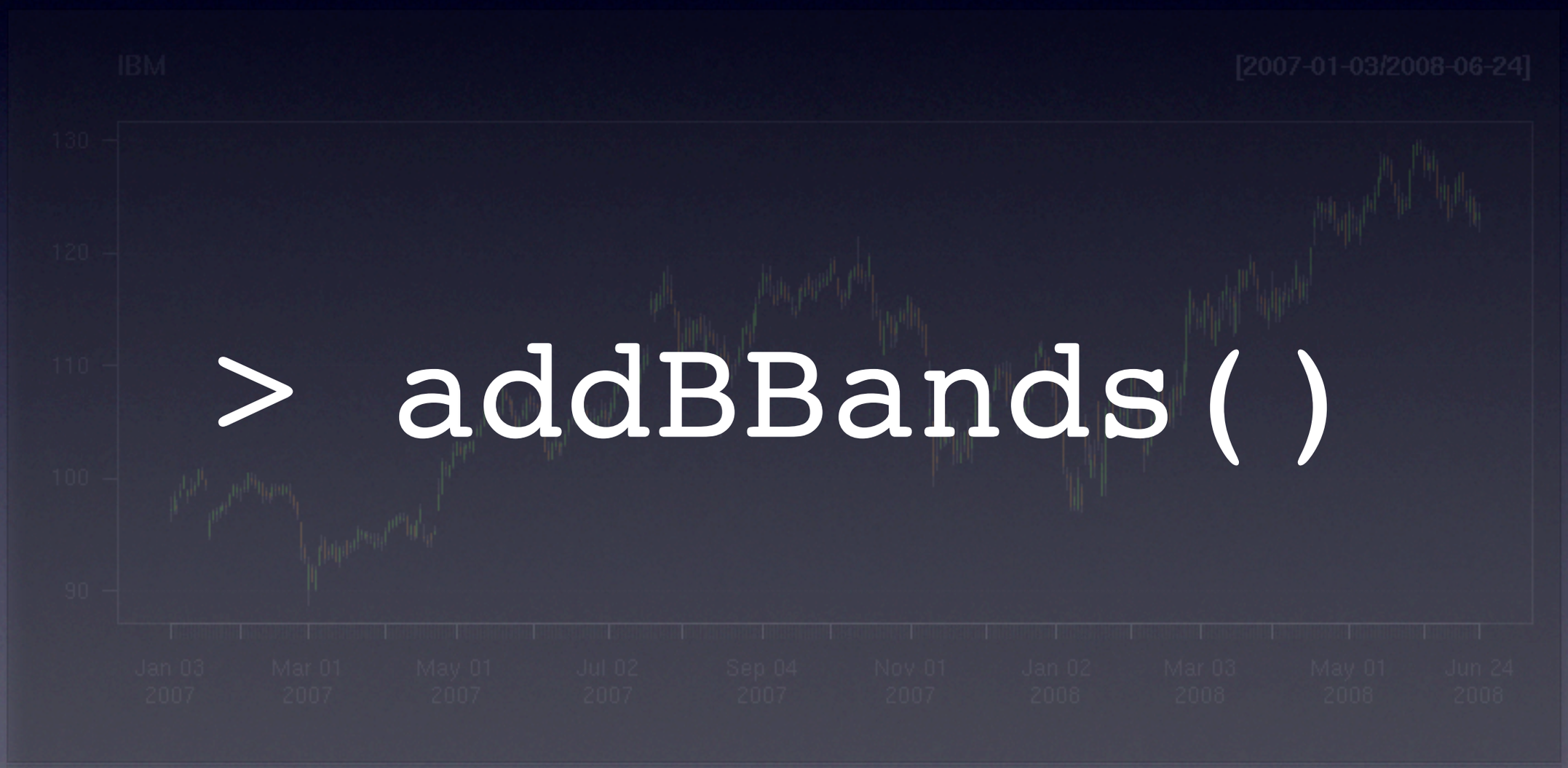
zooming works too!

Adding interactively

chartSeries(IBM, TA=NULL)

Adding interactively

chartSeries(IBM, TA=NULL)




```
> addBBands()
```

Adding interactively

Now with Bollinger Bands under the series

Adding interactively


```
> addDPO()
```

Adding interactively

With a De-trended Price Oscillator

Custom TA

Custom TA

addTA

add data directly to a chart

Custom TA

addTA

add data directly to a chart

newTA

create new TA functions easily

addTA

Provide a mechanism to create TA additions on-demand, using only raw data

addTA

The newest TTR package includes a volatility function to calculate different measures of volatility

addTA

The newest TTR package includes a volatility function to calculate different measures of volatility

e.g. volatility(OHLC(IBM), calc = ‘garman.klass’)

addTA

The newest TTR package includes a volatility function to calculate different measures of volatility

e.g. volatility(OHLC(IBM), calc = 'garman.klass')

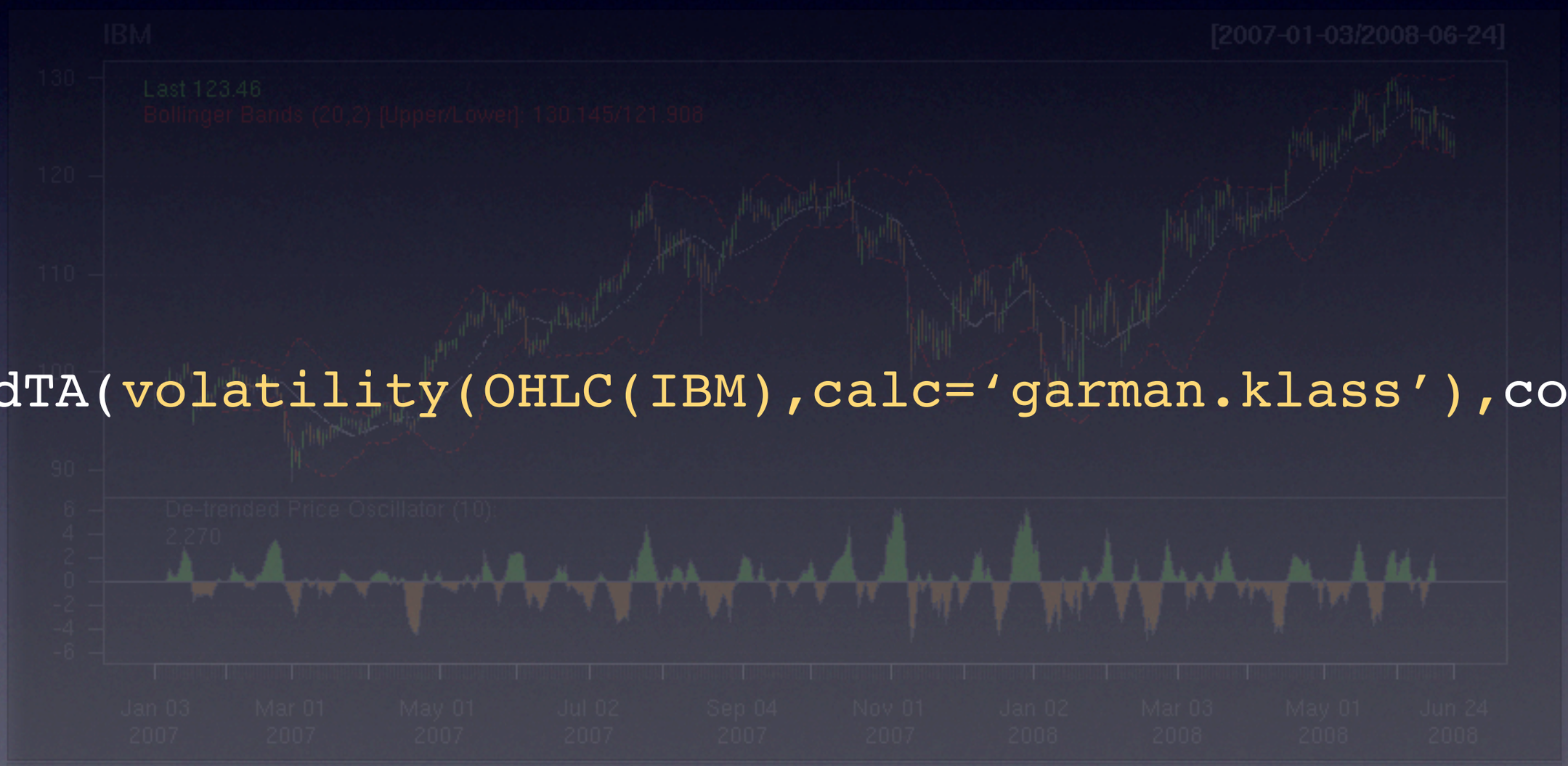
returns the Garman-Klass volatility

addTA

To add this to the IBM chart:

addTA

To add this to the IBM chart:



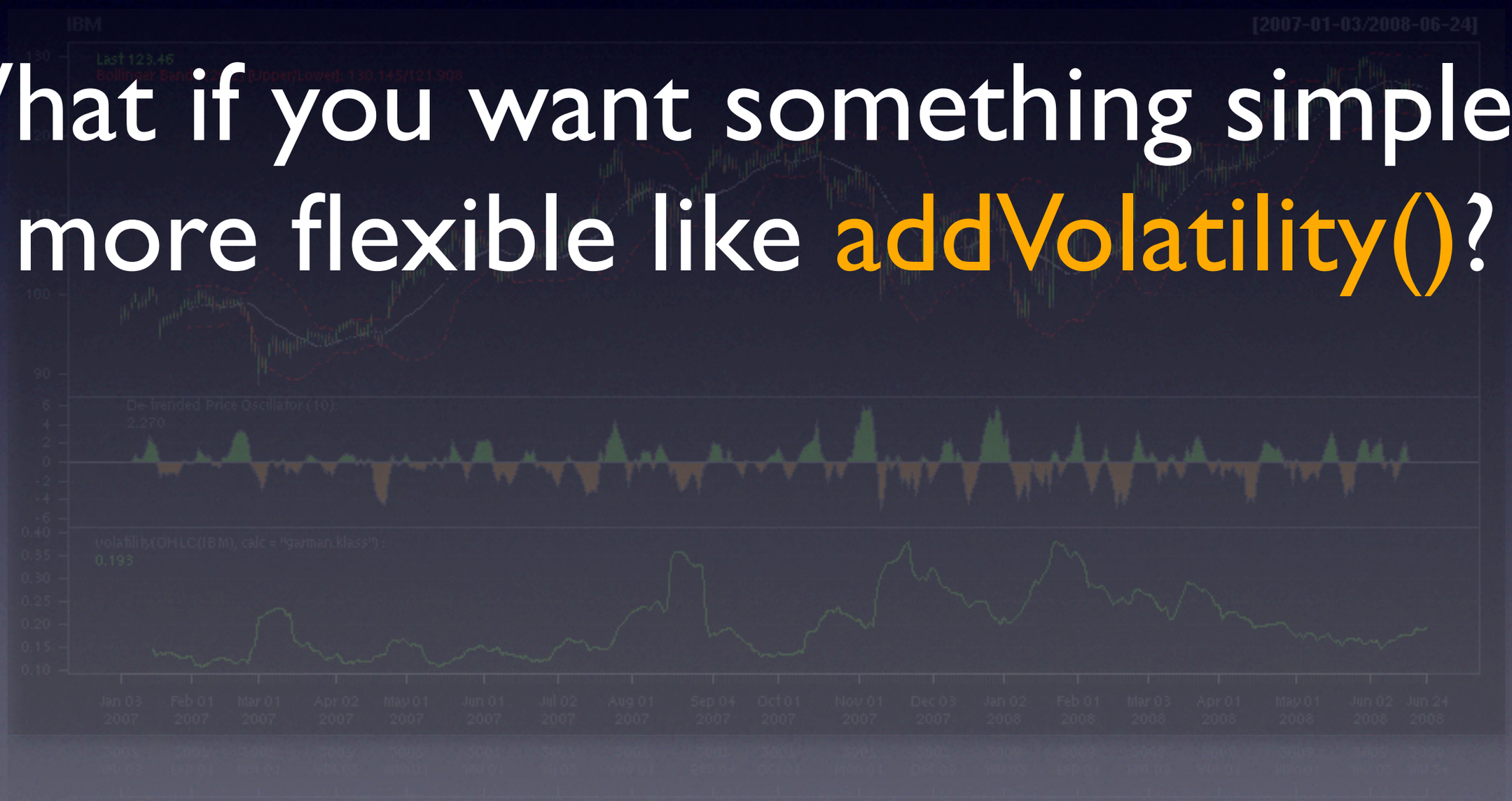

```
> addTA(volatility(OHLC(IBM),calc='garman.klass'),col=3)
```

addTA

The result is displayed just like any built-in TA

addTA

The result is displayed just like any built-in TA



What if you want something simple & more flexible like addVolatility()?

newTA

Provide a mechanism to create functional TA additions based on user functions

newTA

newTA automatically creates the code needed!

```
> addVolatility <- newTA(volatility, preFUN=OHLC, col=4,lwd=2)
>
> class(addVolatility)
[1] “function”
>
```

newTA

simply call the new function


```
> addVolatility()
```

newTA

done!


IBM
[2007-01-03/2008-06-24]
Last 123.46
Bollinger Bands (20,2) [Upper/Lower]: 130.145/121.908
De-trended Price Oscillator (10):
2.270
volatility(OHLC(IBM), calc = "garman.klass") :
0.193
Volatility(calc = "garman.klass") :
0.193
Jan 03 2007
Feb 01 2007
Mar 01 2007
Apr 02 2007
May 01 2007
Jun 01 2007
Jul 02 2007
Aug 01 2007
Sep 04 2007
Oct 01 2007
Nov 01 2007
Dec 03 2007
Jan 02 2008
Feb 01 2008
Mar 03 2008
Apr 01 2008
May 01 2008
Jun 02 2008
Jun 24 2008

quantmod Summary

quantmod Summary

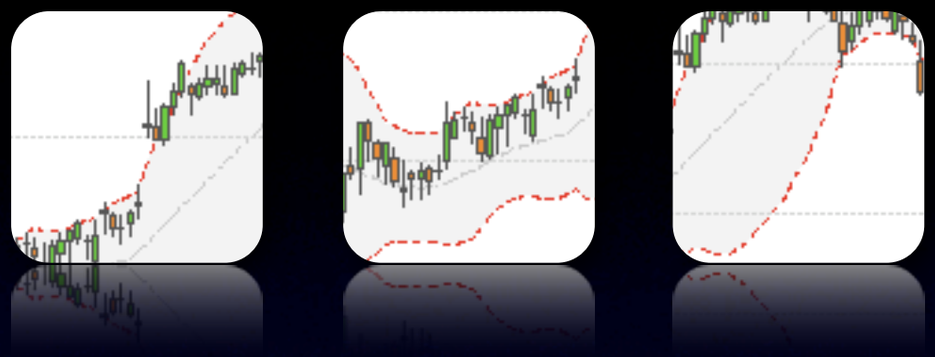
- single interface to data with getSymbols

quantmod Summary

- single interface to data with getSymbols
- fast and flexible visualization in chartSeries

quantmod Summary

- single interface to data with getSymbols
- fast and flexible visualization in chartSeries
- big plans for the future!



Financial Time-Series Tools

xts and chartSeries

Jeffrey A. Ryan
jeffrey.ryan @ insightalgo.com

Joshua M. Ulrich
joshua.m.ulrich @ gmail.com

Presented by Jeffrey Ryan at:
Rmetrics: Computational Finance and Financial Engineering Workshop
June 29 - July 3, 2008, Meielisalp, Lake Thune, Switzerland

www.quantmod.com/Rmetrics2008