

# Rapid Trade Development

Leveraging *R* with xts and quantmod for quantitative trading

Jeffrey A. Ryan

[jeffrey.ryan@insightalgo.com](mailto:jeffrey.ryan@insightalgo.com)

Presented on March 27, 2009

Department of Statistics and Mathematics

Vienna University of Economics and Business Administration

**What is quantitative trading?**

# **What is quantitative trading?**

**A programmatic/automated process to execute trades based on mathematical or statistical logic.**

# Quant Trading Components

Ideas

Strategy Dev

Market Data

Back-testing

Risk Analysis

Forward-  
testing

Execution

Performance  
Analysis

Reports

# Quant Trading Components

Ideas

Strategy Dev

Market Data

**Can this be done in one language?**

Analysis

Forward testing

Performance Analysis

Reports

**It can in R!**

# Advantages to R

Powerful language out of the box

Huge financial community use and contributions

Vector-based data and processing (time-series)

Unequaled statistical toolkit (base + CRAN)

Strong academic research usage

Open source - customization and validation

Easy to integrate with external data stores

Extensibility with C, Fortran, C++, etc

Flexible graphics on-screen and off

# Advantages to R

Powerful language out of the box

Huge financial community use and contributions

Vector-based data and processing (time-series)

Unequaled statistical toolkit (base + CRAN)

Strong academic research usage

Open source - customization and validation

Easy to integrate with external data stores

Extensibility with C, Fortran, C++, etc

Flexible graphics on-screen and off



# Advantages to R

Powerful language out of the box

Huge financial community use and contributions

Vector-based data and processing (time-series)

Unequaled statistical toolkit (base + CRAN)

Strong academic research usage

Open source - customization and validation

Easy to integrate with external data stores

Extensibility with C, Fortran, C++, etc

Flexible graphics on-screen and off

# Advantages to R

Powerful language out of the box

Huge financial community use and contributions

Vector-based data and processing (time-series)

Unequaled statistical toolkit (base + CRAN)

Strong academic research usage

Open source - customization and validation

Easy to integrate with external data stores

Extensibility with C, Fortran, C++, etc

Flexible graphics on-screen and off

# Advantages to R

Powerful language out of the box

Huge financial community use and contributions

Vector-based data and processing (time-series)

Unequaled statistical toolkit (base + CRAN)

Strong academic research usage

Open source - customization and validation

Easy to integrate with external data stores

Extensibility with C, Fortran, C++, etc

Flexible graphics on-screen and off

# Advantages to R

Powerful language out of the box

Huge financial community use and contributions

Vector-based data and processing (time-series)

Unequaled statistical toolkit (base + CRAN)

Strong academic research usage

Open source - customization and validation

Easy to integrate with external data stores

Extensibility with C, Fortran, C++, etc

Flexible graphics on-screen and off

# Advantages to R

Powerful language out of the box

Huge financial community use and contributions

Vector-based data and processing (time-series)

Unequaled statistical toolkit (base + CRAN)

Strong academic research usage

Open source - customization and validation

Easy to integrate with external data stores

Extensibility with C, Fortran, C++, etc

Flexible graphics on-screen and off

# Advantages to R

Powerful language out of the box

Huge financial community use and contributions

Vector-based data and processing (time-series)

Unequaled statistical toolkit (base + CRAN)

Strong academic research usage

Open source - customization and validation

Easy to integrate with external data stores

Extensibility with C, Fortran, C++, etc

Flexible graphics on-screen and off

# Advantages to R

Powerful language out of the box

Huge financial community use and contributions

Vector-based data and processing (time-series)

Unequaled statistical toolkit (base + CRAN)

Strong academic research usage

Open source - customization and validation

Easy to integrate with external data stores

Extensibility with C, Fortran, C++, etc

Flexible graphics on-screen and off

# Advantages to R

Powerful language out of the box

Huge financial community use and contributions

Vector-based data and processing (time-series)

Unequaled statistical toolkit (base + CRAN)

Strong academic research usage

Open source - customization and validation

Easy to integrate with external data stores

Extensibility with C, Fortran, C++, etc

Flexible graphics on-screen and off



# What was missing for trading?

A trader-oriented workflow

Standard financial visualization tools

Native tools for managing [large] time series quickly

Connections to real-time data feeds (Bloomberg, IB, DTN)

Connections to industry data stores (Vhayu, OneTick, Kdb)

Wide-spread adoption & visibility

# What was missing for trading?

A trader-oriented workflow

Standard financial visualization tools

Native tools for managing [large] time series quickly

Connections to real-time data feeds (Bloomberg, IB, DTN)

Connections to industry data stores (Vhayu, OneTick, Kdb)

Wide-spread adoption & visibility

quantmod

# What was missing for trading?

A trader-oriented workflow

Standard financial visualization tools

Native tools for managing [large] time series quickly

Connections to real-time data feeds (Bloomberg, IB, DTN)

Connections to industry data stores (Vhayu, OneTick, Kdb)

Wide-spread adoption & visibility

quantmod

xts

# What was missing for trading?

A trader-oriented workflow focus

Standard financial visualization tools

Native tools for managing [large] time series quickly

Connections to real-time data feeds (Bloomberg, IB, DTN)

Connections to industry data stores (Vhayu, OneTick, Kdb)

Wide-spread adoption & visibility

quantmod

xts

## xts: extensible time series

---

Most everything in trading involves a time series

*Regular* data (positions data, P&L)

*Irregular* data (market data, book data, trades)

R has many ways to manage this...

# xts: extensible time series

---

Data Classes

# xts: extensible time series

---

**fts**

**Data Classes**

**matrix**

**mts**

**tframe**

**data.table**

**data.frame**

**ZOO**

**its**

**ts**

**irts**

**timeSeries**

**vectors**

**zooreg**

# xts: extensible time series

---

**fts**

Data Classes

**matrix**

mts

**tframe**

**data.table**

**data.frame**

**ZOO**

its

ts

irts

**timeSeries**

vectors

zooreg

Time Classes



# xts: extensible time series

---

fts

Data Classes

matrix

mts

tframe

data.table

data.frame

ZOO

its

ts

irts

timeSeries

zooreg

vectors

Time Classes

chron

POSIXct

character

Date

POSIXlt

numeric

yearmon

yearqtr

timeDate

# xts: extensible time series

fts

Data Classes

matrix

mts

tframe

data.table

data.frame

ZOO

its

ti

Sometimes, choice is **bad** for package developers and interoperability

irts

vectors

character

Time Classes

Date

POSIXlt

numeric

yearmon

yearqtr

timeDate

# xts: extensible time series

---

The “solution” ?

# xts: extensible time series

---

add one more class *of course*...

## Motivation (c. 2007)

Avid user of zoo

- Natural R-like interface
- Flexible and complete methods
- S3!

I still wanted a few features for trading...

- Additional series metadata
- Require time-based indexing
- Conversion/reconversion tools

### Significant design requirements for xts:

- Preserve zoo behavior
- Utilize time-based indexing
- Allow for arbitrary attributes to be cleanly attached
- ISO 8601 subsetting by time strings
- Lossless conversion utilities to hide messy details

# xts: extensible time series

---

## What's inside an xts object?

Index

+

Matrix

+

Attr



Internally the storage is always a numeric vector!

Can be of any class supported by matrix

Hidden attributes AND user attributes

.indexCLASS, .indexFORMAT,  
.indexTZ, .CLASS, index

# xts: extensible time series

---

## What's inside an xts object?

Index

+

Matrix

+

Attr

Internally the storage is always a numeric vector!

Can be of any class supported by matrix

Hidden attributes AND user attributes

.indexCLASS, .indexFORMAT,  
.indexTZ, .CLASS, index



# xts: extensible time series

---

## What's inside an xts object?

Index

+

Matrix

+

Attr



Internally the storage is always a numeric vector!

Can be of any class supported by matrix

Hidden attributes AND user attributes

`.indexCLASS`, `.indexFORMAT`,  
`.indexTZ`, `.CLASS`, `index`

# xts: extensible time series

---

## What's inside an xts object?

Index

+

Matrix

+

Attr

Internally the storage is  
always a numeric vector!

Can be of any class  
supported by matrix

Hidden attributes AND user  
attributes

.indexCLASS, .indexFORMAT,  
.indexTZ, .CLASS, index

# xts: extensible time series

## What's inside an xts object?

Index

+

Matrix

+

Attr

Internally the storage is  
always a numeric vector!

Can be of any class  
supported by matrix

Hidden attributes AND user  
attributes

.indexCLASS, .indexFORMAT,  
.indexTZ, .CLASS, index

**Important! index must be a time-based  
class**

## xts: extensible time series

---

Index as numeric? That isn't "time-based"!!

- Internally all index values are represented in POSIX time (seconds since the epoch)
- Coercion happens at object creation or upon index replacement
- `index()` converts back to user-level class
- `.index()` access the raw seconds in the index
- `.indexCLASS`, `.indexFORMAT` and `.indexTZ` attributes
- Rationale? Simplicity for C level code, removal of multiple conversion in most instances, more consistent behavior
- All details are hidden from the user

### Time-based indexing in xts (ISO 8601)

- Date and time organized from *most significant to least significant*:  
CCYY-MM-DD HH:MM:SS[.s]
- Fixed number of digits
- Separators can be omitted e.g. CCYYMMDDHHMMSS
- Reduced accuracy forms are valid: e.g. CCYY-MM
- Fractional decimal time is supported
- Intervals can be expressed e.g. 2000-05/2001-04

## Create an xts object

Load xts package

```
> library(xts)
Loading required package: zoo
xts now requires a valid TZ variable to be set
your current TZ:America/Chicago
```

```
> x <- xts(rnorm(10), Sys.Date()+1:10)
```

```
> x
```

```
          [,1]
2009-03-24 0.3554788
2009-03-25 1.2812633
2009-03-26 0.1268833
2009-03-27 -0.6945146
2009-03-28 -0.3936148
2009-03-29 -0.1938840
2009-03-30 0.2368576
2009-03-31 -1.2152293
2009-04-01 0.8100493
2009-04-02 1.4152439
```

## Create an xts object

Create an xts object

```
> library(xts)
Loading required package: zoo
xts now requires a valid TZ variable to be set
your current TZ:America/Chicago
```

```
> x <- xts(rnorm(10), Sys.Date()+1:10)
> x
```

```
      [,1]
2009-03-24 0.3554788
2009-03-25 1.2812633
2009-03-26 0.1268833
2009-03-27 -0.6945146
2009-03-28 -0.3936148
2009-03-29 -0.1938840
2009-03-30 0.2368576
2009-03-31 -1.2152293
2009-04-01 0.8100493
2009-04-02 1.4152439
```

## Indexing by time

Index using standard tools (still works)

```
> x[ index(x) >= as.Date("2009-03-28") & index(x) <=
+ as.Date("2009-04-01") ]
```

```
      [,1]
2009-03-28 -0.3936148
2009-03-29 -0.1938840
2009-03-30  0.2368576
2009-03-31 -1.2152293
2009-04-01  0.8100493
```

```
> x["20090328/20090401"]
```

```
      [,1]
2009-03-28 -0.3936148
2009-03-29 -0.1938840
2009-03-30  0.2368576
2009-03-31 -1.2152293
2009-04-01  0.8100493
```



## Indexing by time

Index via ISO-style with xts

```
> x[ index(x) >= as.Date("2009-03-28") & index(x) <=
+ as.Date("2009-04-01") ]
```

```
      [,1]
2009-03-28 -0.3936148
2009-03-29 -0.1938840
2009-03-30  0.2368576
2009-03-31 -1.2152293
2009-04-01  0.8100493
```

```
> x["20090328/20090401"]
```

```
      [,1]
2009-03-28 -0.3936148
2009-03-29 -0.1938840
2009-03-30  0.2368576
2009-03-31 -1.2152293
2009-04-01  0.8100493
```

# xts: extensible time series

## Indexing by time

```
> x["2009"]  
      [,1]  
2009-03-24 0.3554788  
2009-03-25 1.2812633  
2009-03-26 0.1268833  
2009-03-27 -0.6945146  
2009-03-28 -0.3936148  
2009-03-29 -0.1938840  
2009-03-30 0.2368576  
2009-03-31 -1.2152293  
2009-04-01 0.8100493  
2009-04-02 1.4152439
```

```
> x["200904"]  
      [,1]  
2009-04-01 0.8100493  
2009-04-02 1.4152439
```

```
> x["20090301/200903"]  
      [,1]  
2009-03-24 0.3554788  
2009-03-25 1.2812633  
2009-03-26 0.1268833  
2009-03-27 -0.6945146  
2009-03-28 -0.3936148  
2009-03-29 -0.1938840  
2009-03-30 0.2368576  
2009-03-31 -1.2152293
```



All of 2009

## Indexing by time

```
> x["2009"]  
      [,1]  
2009-03-24 0.3554788  
2009-03-25 1.2812633  
2009-03-26 0.1268833  
2009-03-27 -0.6945146  
2009-03-28 -0.3936148  
2009-03-29 -0.1938840  
2009-03-30 0.2368576  
2009-03-31 -1.2152293  
2009-04-01 0.8100493  
2009-04-02 1.4152439
```

```
> x["200904"]  
      [,1]  
2009-04-01 0.8100493  
2009-04-02 1.4152439
```

```
> x["20090301/200903"]  
      [,1]  
2009-03-24 0.3554788  
2009-03-25 1.2812633  
2009-03-26 0.1268833  
2009-03-27 -0.6945146  
2009-03-28 -0.3936148  
2009-03-29 -0.1938840  
2009-03-30 0.2368576  
2009-03-31 -1.2152293
```



All of April 2009

## Indexing by time

```
> x["2009"]  
      [,1]  
2009-03-24 0.3554788  
2009-03-25 1.2812633  
2009-03-26 0.1268833  
2009-03-27 -0.6945146  
2009-03-28 -0.3936148  
2009-03-29 -0.1938840  
2009-03-30 0.2368576  
2009-03-31 -1.2152293  
2009-04-01 0.8100493  
2009-04-02 1.4152439
```

```
> x["200904"]  
      [,1]  
2009-04-01 0.8100493  
2009-04-02 1.4152439
```

```
> x["20090301/200903"]  
      [,1]  
2009-03-24 0.3554788  
2009-03-25 1.2812633  
2009-03-26 0.1268833  
2009-03-27 -0.6945146  
2009-03-28 -0.3936148  
2009-03-29 -0.1938840  
2009-03-30 0.2368576  
2009-03-31 -1.2152293
```



From the first March to the end of March

# xts: extensible time series

---

All subsetting is via a binary search algorithm. F-A-S-T!

```
> str(x10m) # 10 million observations
```

```
An 'xts' object from 2009-03-23 16:19:00 to 2009-07-17 10:05:39 containing:
```

```
Data: int [1:1000000, 1] 1 2 3 4 5 6 7 8 9 10 ...
```

```
Indexed by objects of class: [POSIXt,POSIXct] TZ:America/Chicago
```

```
xts Attributes:
```

```
NULL
```

```
> str(x100k) # 100 thousand observations
```

```
An 'xts' object from 2009-03-23 16:19:00 to 2009-03-24 20:05:39 containing:
```

```
Data: int [1:100000, 1] 1 2 3 4 5 6 7 8 9 10 ...
```

```
Indexed by objects of class: [POSIXt,POSIXct] TZ:America/Chicago
```

```
xts Attributes:
```

```
NULL
```

```
> system.time(x10m['20090323'])
```

```
user system elapsed
```

```
0.006 0.001 0.006
```

```
> system.time(x100k['20090323'])
```

```
user system elapsed
```

```
0.006 0.001 0.006
```

```
> system.time(x10m[index(x10m) >= as.POSIXct('2009-03-23 16:19:00') & index(x10m) <=
```

```
as.POSIXct('2009-03-23 23:59:58')])
```

```
user system elapsed
```

```
1.457 1.372 2.832
```

# xts: extensible time series

---

All subsetting is via a binary search algorithm. F-A-S-T!

```
> str(x10m) # 10 million observations
An 'xts' object from 2009-03-23 16:19:00 to 2009-07-17 10:05:39 containing:
  Data: int [1:10000000, 1] 1 2 3 4 5 6 7 8 9 10 ...
  Indexed by objects of class: [POSIXt,POSIXct] TZ:America/Chicago
  xts Attributes:
  NULL

> str(x100k) # 100 thousand observations
An 'xts' object from 2009-03-23 16:19:00 to 2009-03-24 20:05:39 containing:
  Data: int [1:100000, 1] 1 2 3 4 5 6 7 8 9 10 ...
  Indexed by objects of class: [POSIXt,POSIXct] TZ:America/Chicago
  xts Attributes:
  NULL

> system.time(x10m['20090323'])
  user system elapsed
 0.006  0.001  0.006
> system.time(x100k['20090323'])
  user system elapsed
 0.006  0.001  0.006

> system.time(x10m[index(x10m) >= as.POSIXct('2009-03-23 16:19:00') & index(x10m) <=
  as.POSIXct('2009-03-23 23:59:58')])
  user system elapsed
 1.457  1.372  2.832
```

# xts: extensible time series

---

All subsetting is via a binary search algorithm. F-A-S-T!

```
> str(x10m) # 10 million observations
An 'xts' object from 2009-03-23 16:19:00 to 2009-07-17 10:05:39 containing:
  Data: int [1:10000000, 1] 1 2 3 4 5 6 7 8 9 10 ...
  Indexed by objects of class: [POSIXt,POSIXct] TZ:America/Chicago
  xts Attributes:
  NULL

> str(x100k) # 100 thousand observations
An 'xts' object from 2009-03-23 16:19:00 to 2009-03-24 20:05:39 containing:
  Data: int [1:100000, 1] 1 2 3 4 5 6 7 8 9 10 ...
  Indexed by objects of class: [POSIXt,POSIXct] TZ:America/Chicago
  xts Attributes:
  NULL

> system.time(x10m['20090323'])
  user system elapsed
 0.006 0.001 0.006
> system.time(x100k['20090323'])
  user system elapsed
 0.006 0.001 0.006

> system.time(x10m[index(x10m) >= as.POSIXct('2009-03-23 16:19:00') & index(x10m) <=
  as.POSIXct('2009-03-23 23:59:58')])
  user system elapsed
 1.457 1.372 2.832
```

## xts: extensible time series

---

xts + C

- Moving [.xts to C dramatically decreased subsetting costs
- Highest cost basic operation in R was merge. Prime C candidate
- Implemented optimized sort-merge-join in C with custom algorithm
- Additional C based routines followed...

xts now has 3000+ lines of C



xts: extensible time series

---

**...in development**

## Binary `.xd` files

Representation of xts objects on disk

Seekable for disk-based subsetting

Future time-series database structure

XTS  
(disk)

xts  
(memory)

xtsDB

## Parallel processing

period.apply  
runSum, runCov, runSD, etc.

## Parallel processing

`period.apply`  
`runSum`, `runCov`, `runSD`, etc.

## Multiple index support

*index lists*

## Parallel processing

`period.apply`  
`runSum`, `runCov`, `runSD`, etc.

## Multiple index support

*index lists*

## Tighter zoo integration

Backport C code into zoo



quantmod

quantmod was envisioned to be a rapid prototyping environment in R to facilitate quantitative modeling, testing, and trading

Data. Visualization. Modeling.



Data. Visualization. Modeling.

Trading requires lots of different types of data, from many different sources. quantmod aims to hide the details of the data source, to make using data a priority

Data. Visualization. Modeling.

Trading requires lots of different types of data, from many different sources. quantmod aims to hide the details of the data source, to make using data a priority

**getSymbols**

Data. Visualization. Modeling.

## getSymbols

csv	Rdata	MySQL
SQLite	google	yahoo
Interactive Brokers	FRED	oanda

Data. Visualization. Modeling.

## getSymbols

getSymbols is the top level function that dispatches to custom methods based on user direction

setSymbolLookup  
getSymbolLookup  
saveSymbolLookup  
loadSymbolLookup

Data. Visualization. Modeling.

## getSymbols

getSymbols behave like base::load by assigning objects into the user's workspace (.GlobalEnv)

Data. Visualization. Modeling.

## getSymbols

getSymbols behave like base::load by assigning objects into the user's workspace (.GlobalEnv)

Rationale: when dealing with potentially dozens of symbols interactively, it is redundant to have to manually assign each. Also facilitates multiple requests.

Data. Visualization. Modeling.

## getSymbols

```
getSymbols("AAPL")
```

```
getSymbols("AAPL;SBUX")
```

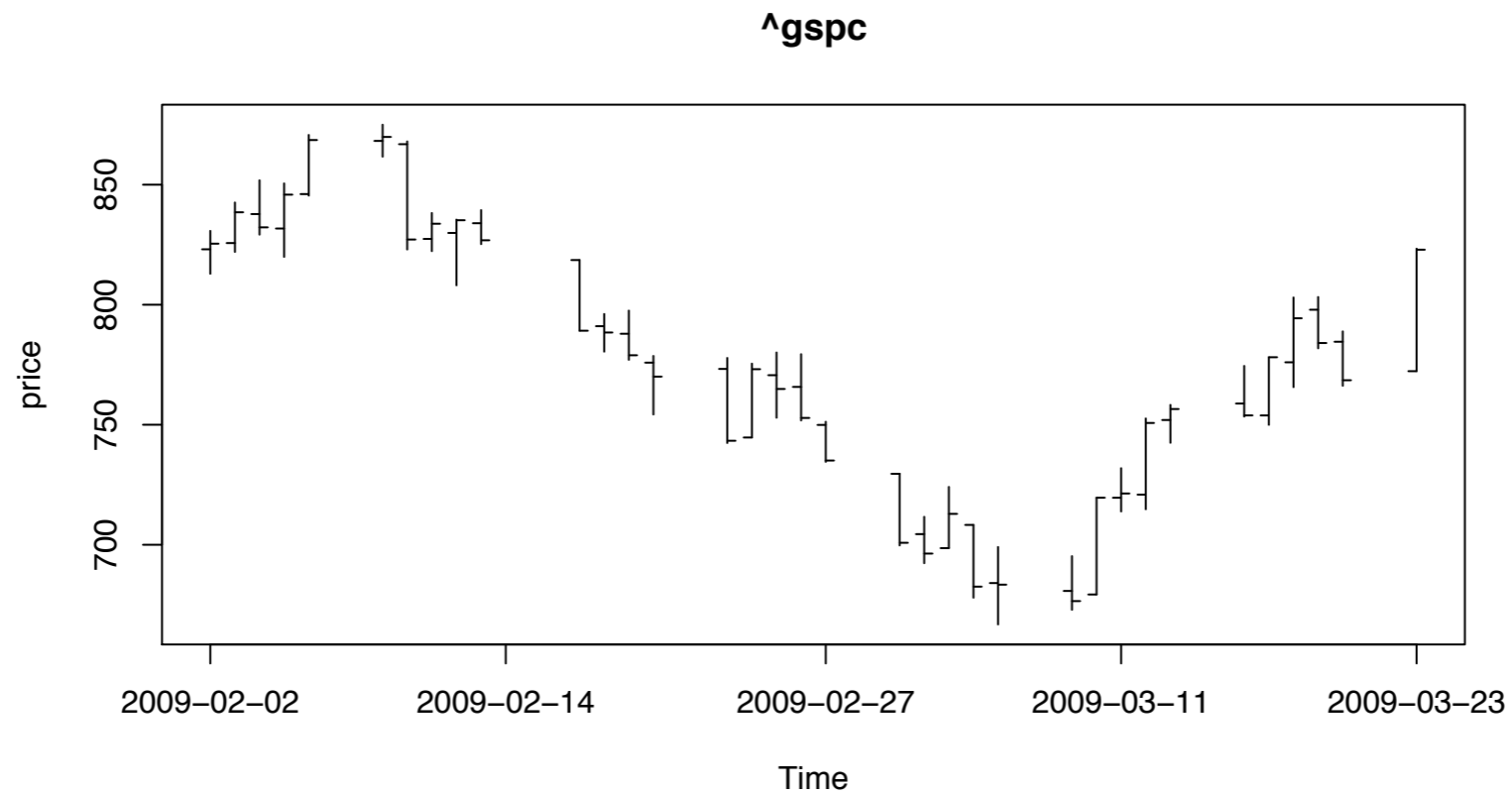
```
getSymbols("USD/EUR",src="oanda")
```

Data. Visualization. Modeling.

Interactive, highly customizable financial charting *in R*

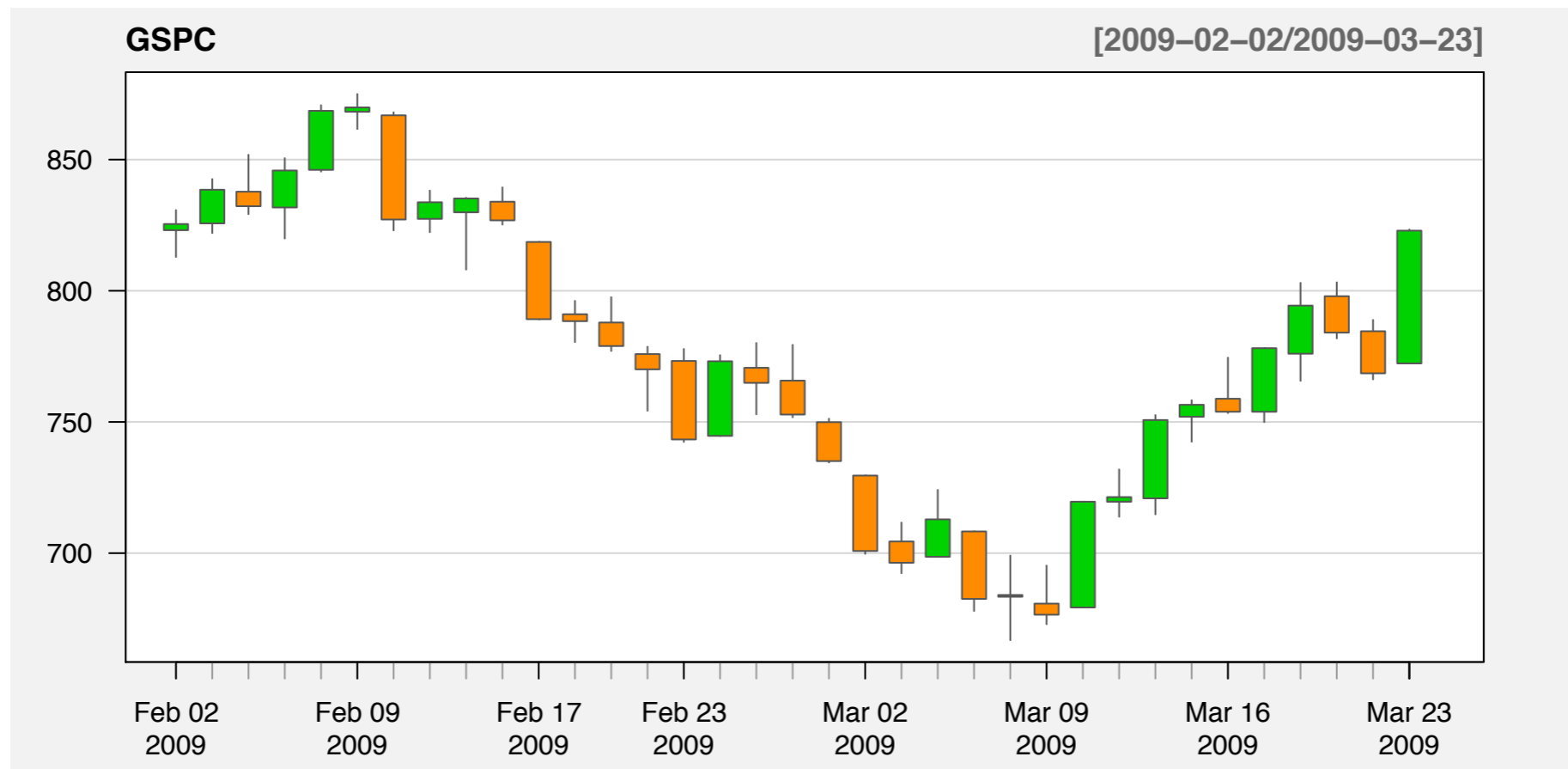


Data. Visualization. Modeling.



Basic OHLC chart from **tseries**

Data. Visualization. Modeling.



```
candleChart(GSPC, subset='200902/', theme='white', TA=NULL)
```

Data. **Visualization.** Modeling.

## Requirements

- Fast rendering (base plotting tools)
- Interactive and scriptable
- Work with all timeseries classes
- Minimal commands
- Highly customizable
- Full technical analysis support (via TTR)

Data. Visualization. Modeling.

# The Basics



> chartSeries(IBM)

Data. Visualization. Modeling.

# The Basics



> addBBands()

Data. Visualization. Modeling.

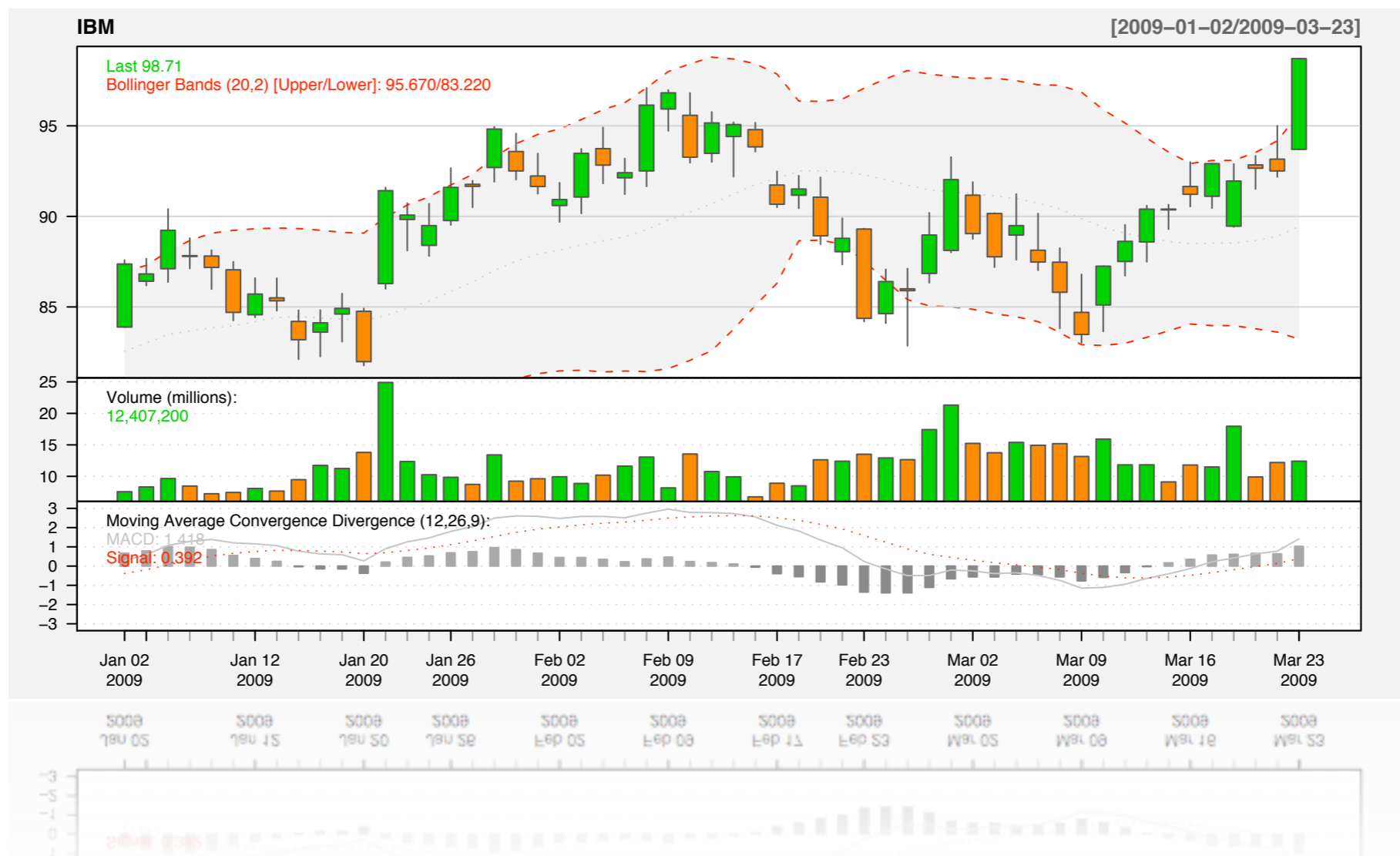
# The Basics



```
> addMACD(32,50,12)
```

Data. Visualization. Modeling.

# The Basics



> reChart(subset="2009",theme="white",type="candles")

Data. Visualization. Modeling.

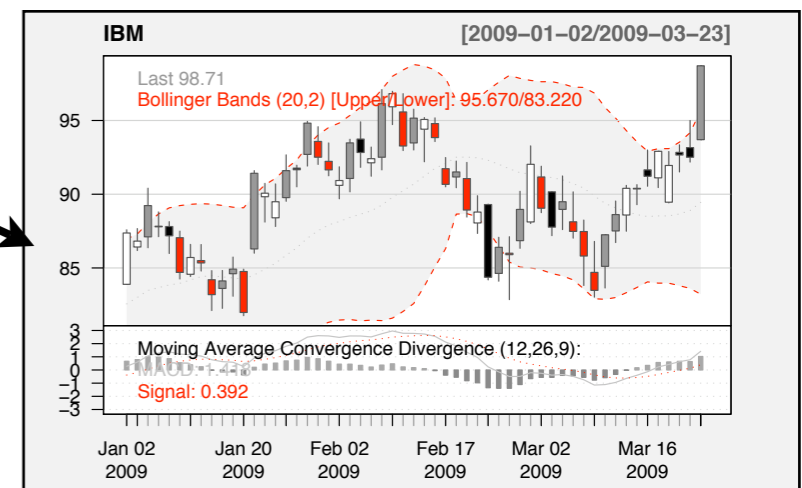
## Inside chartSeries

chartSeries

addTA

chobTA

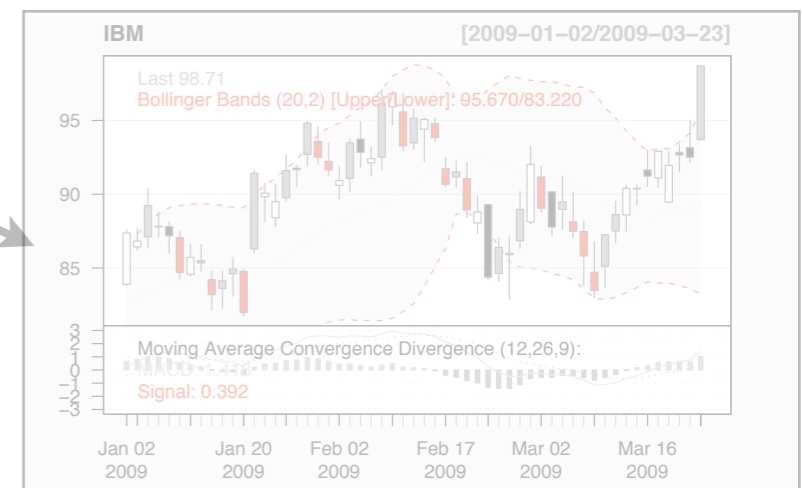
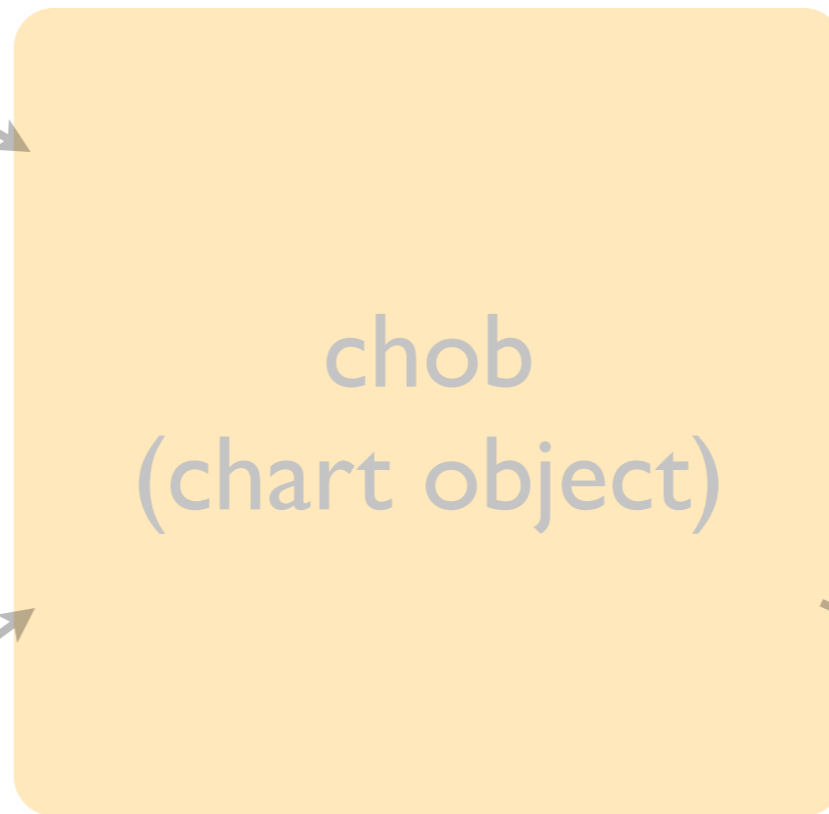
chob  
(chart object)





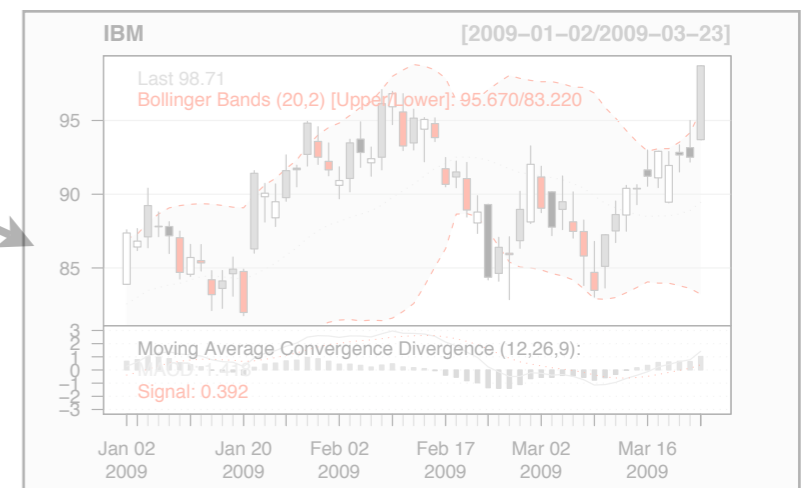
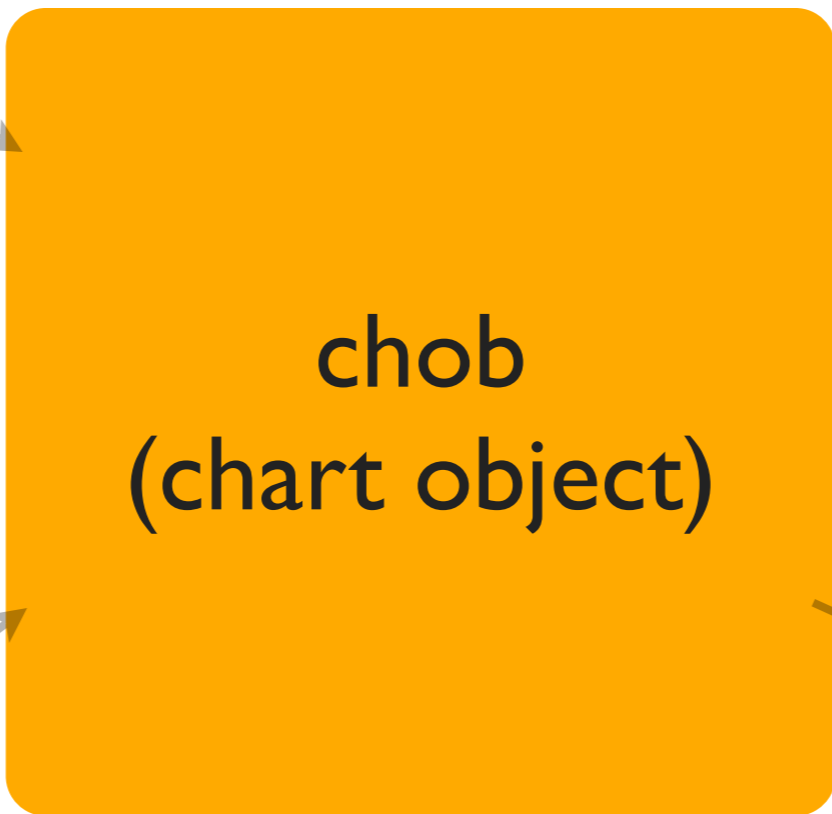
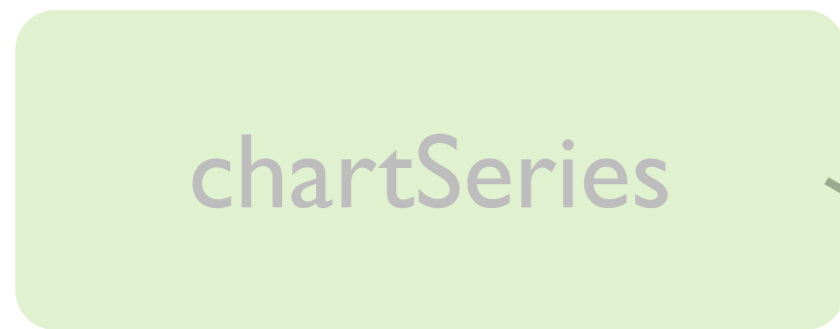
Data. Visualization. Modeling.

## Inside chartSeries



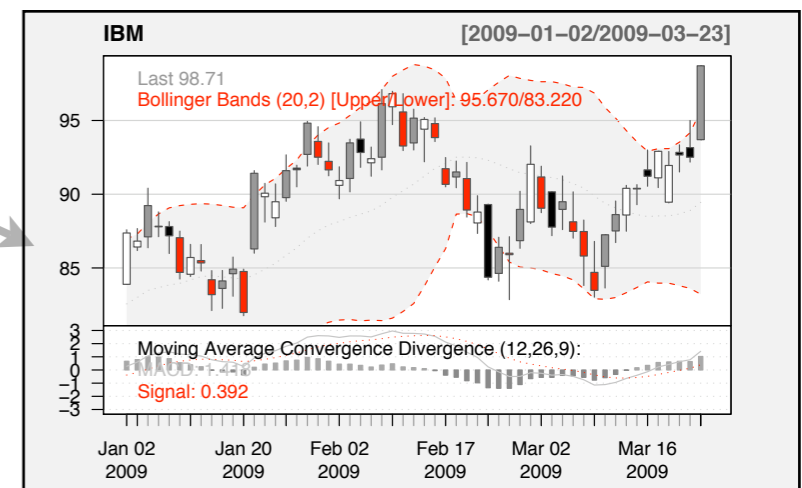
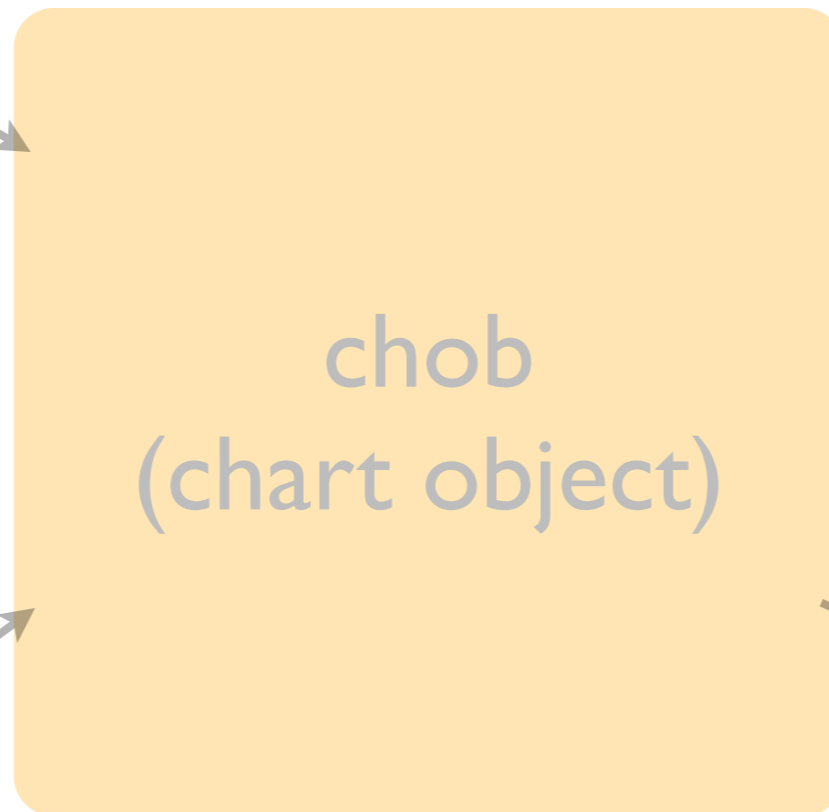
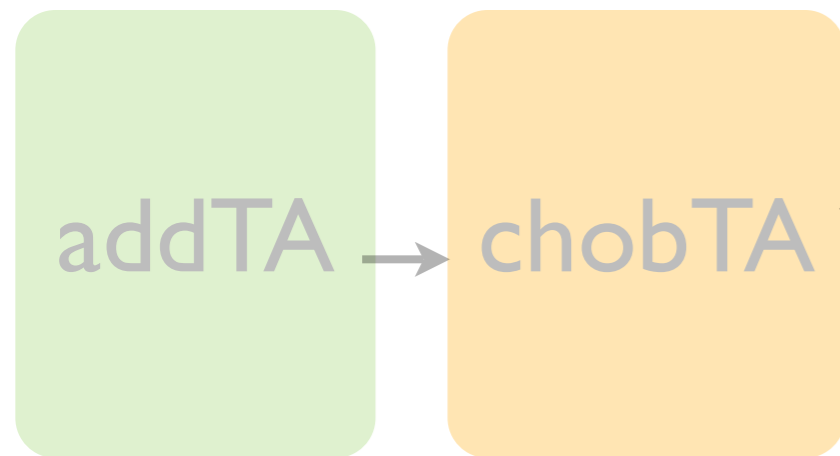
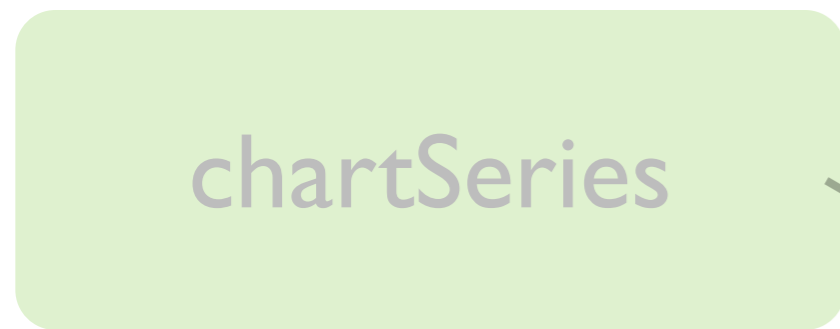
Data. Visualization. Modeling.

## Inside chartSeries



Data. Visualization. Modeling.

## Inside chartSeries



Drawn by chartSeries.chob

Data. **Visualization.** Modeling.

## Extending chartSeries

Data. Visualization. Modeling.

GMMA

Guppy Multiple Moving Average  
(with newTA)

Data. Visualization. Modeling.

```
> # create a function that returns our GMMA
> GMMA <- function(x) {
+   fastMA <- c(3,5,8,10,12,15)
+   slowMA <- c(30,35,40,45,50,60)
+   x <- sapply(c(fastMA,slowMA),
+               function(xx) EMA(x,xx))
+   return(x)
+ }
>
```

## Data. Visualization. Modeling.

```
> # create a function that returns our GMMA
> GMMA <- function(x) {
+   fastMA <- c(3,5,8,10,12,15)
+   slowMA <- c(30,35,40,45,50,60)
+   x <- sapply(c(fastMA,slowMA),
+               function(xx) EMA(x,xx))
+   return(x)
+ }
>
```

```
> # create an addGuppy function with newTA
> addGuppy <- newTA(FUN=GMMA,
+                  preFUN=CI,
+                  col=c(rep(3,6),
+                        rep("#333333",6)),
+                  legend="GMMA")
> class(addGuppy)
[1] "function"
```

Data. **Visualization.** Modeling.

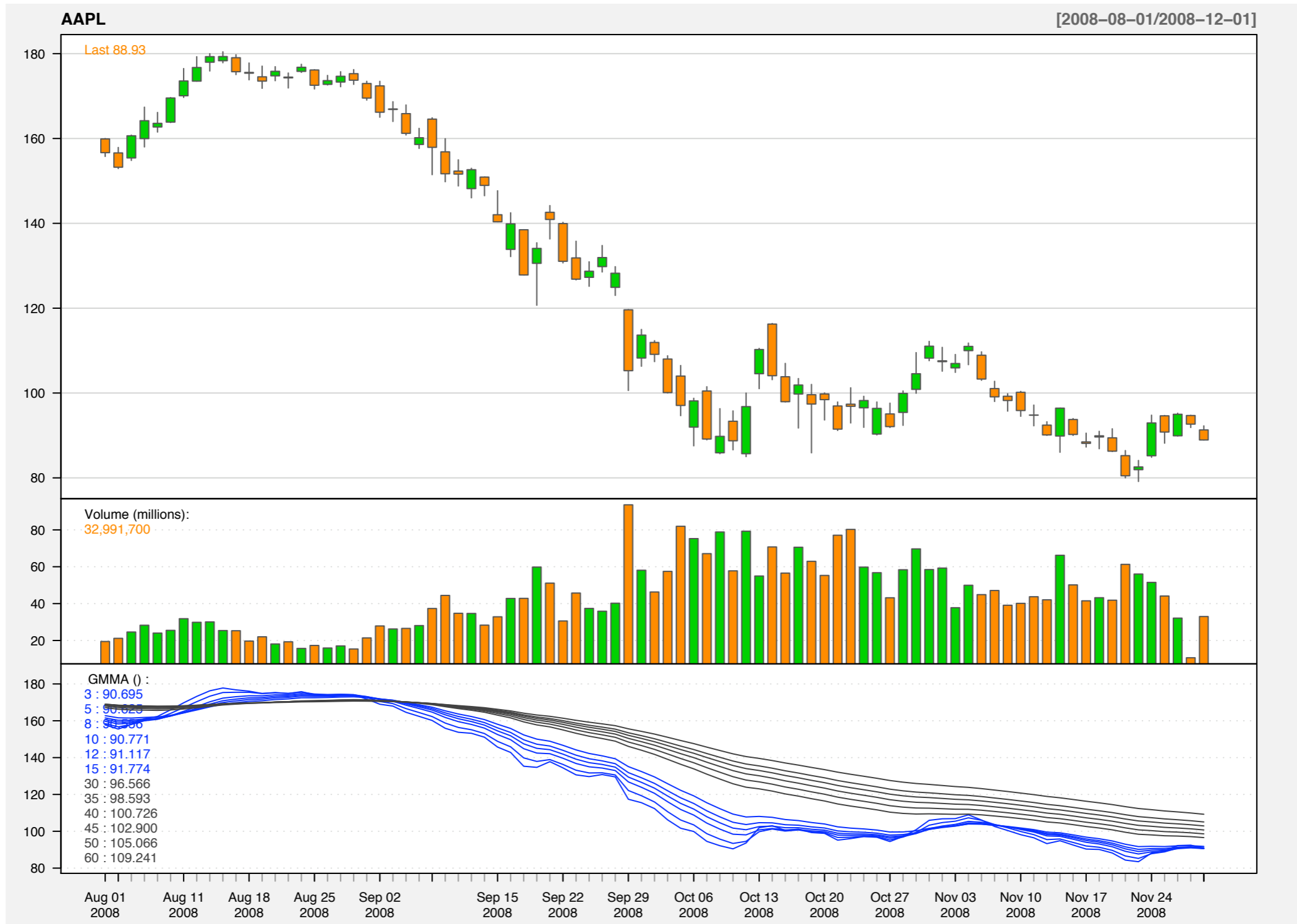
```
> # create a function that returns our GMMA
> GMMA <- function(x) {
+   fastMA <- c(3,5,8,10,12,15)
+   slowMA <- c(30,35,40,45,50,60)
+   x <- sapply(c(fastMA,slowMA),
+               function(xx) EMA(x,xx))
+   return(x)
+ }
```

## candleChart(AAPL); addGuppy()

```
> # create an addGuppy function with newTA
> addGuppy <- newTA(FUN=GMMA,
+                   preFUN=CI,
+                   col=c(rep(3,6),
+                           rep("#333333",6)),
+                   legend="GMMA")
> class(addGuppy)
[1] "function"
```



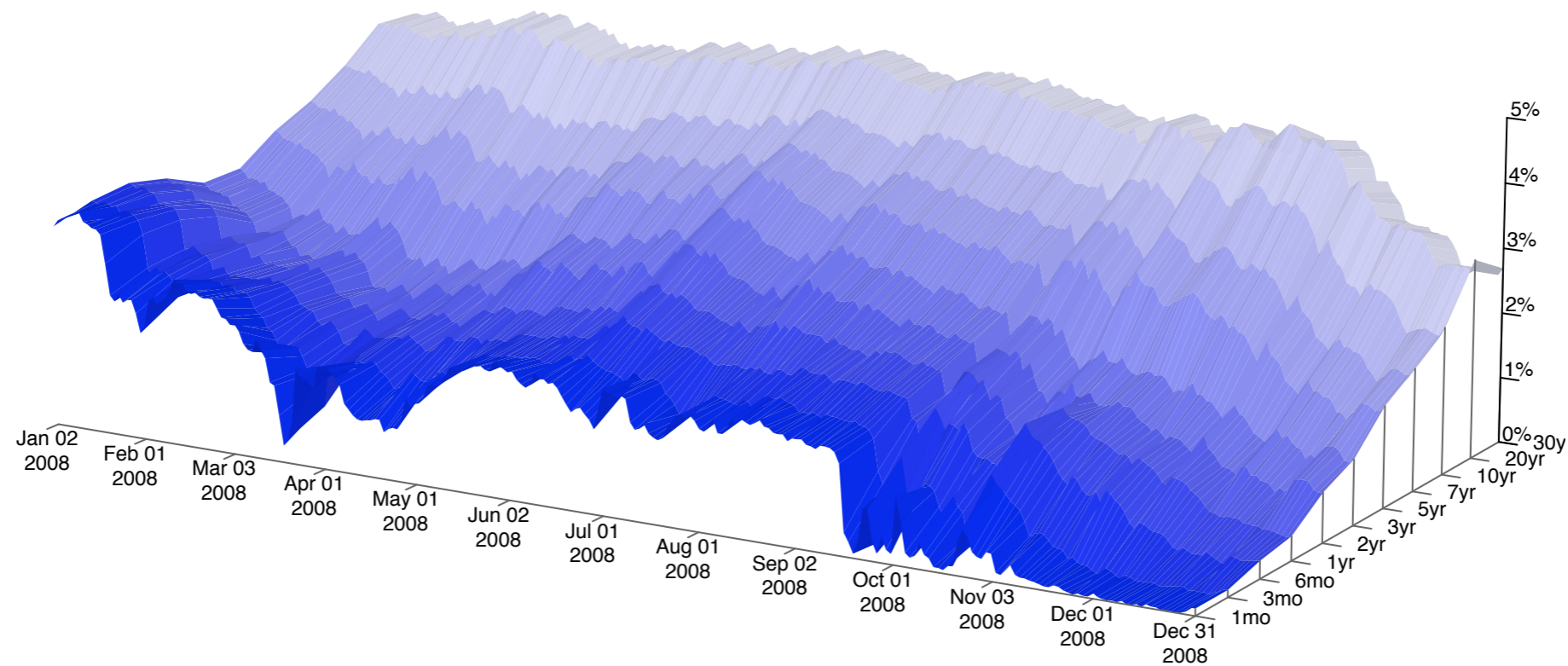
## Data. Visualization. Modeling.



Data. Visualization. Modeling.

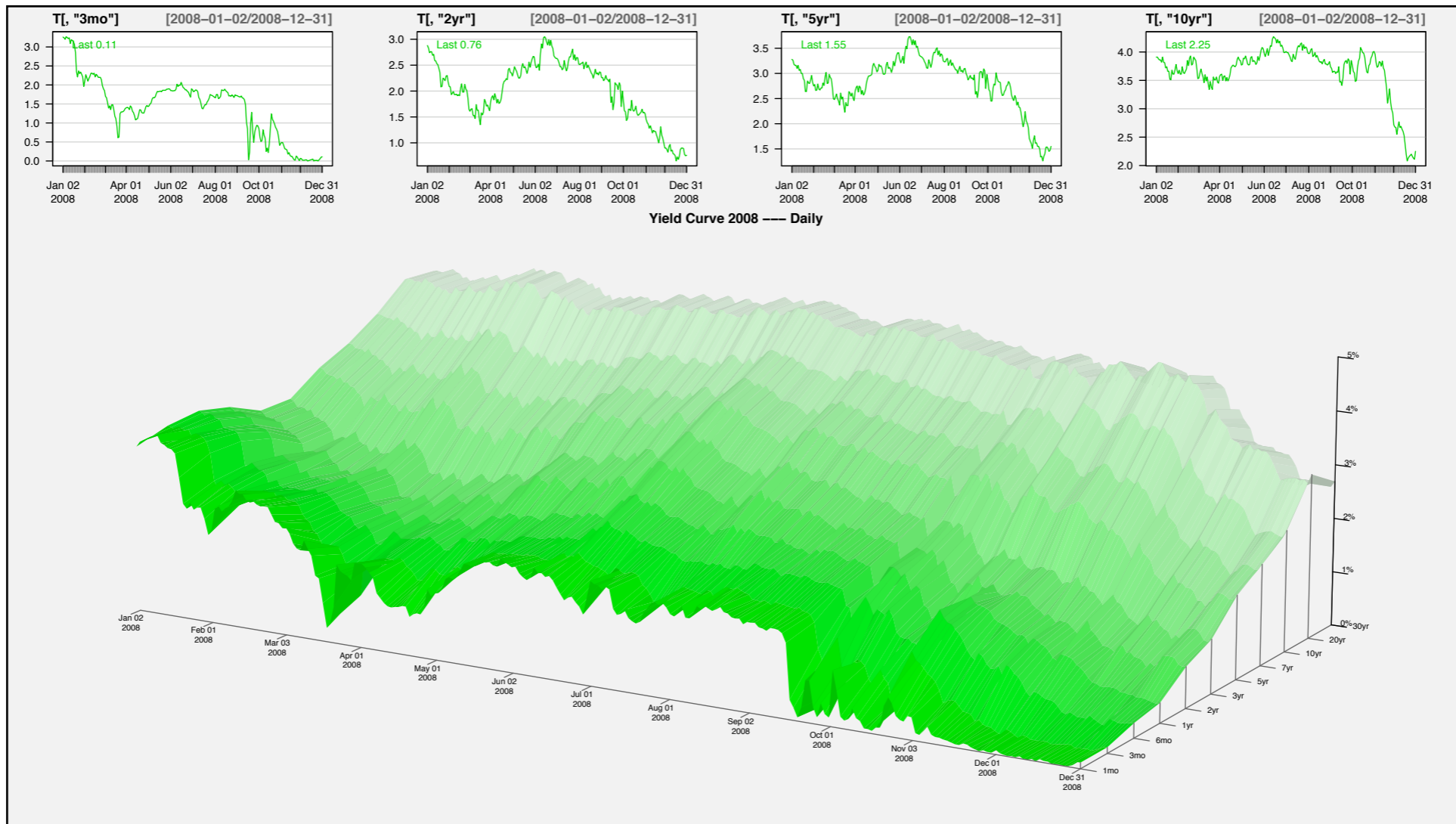
# chartSeries3d

Yield Curve 2008 --- Daily

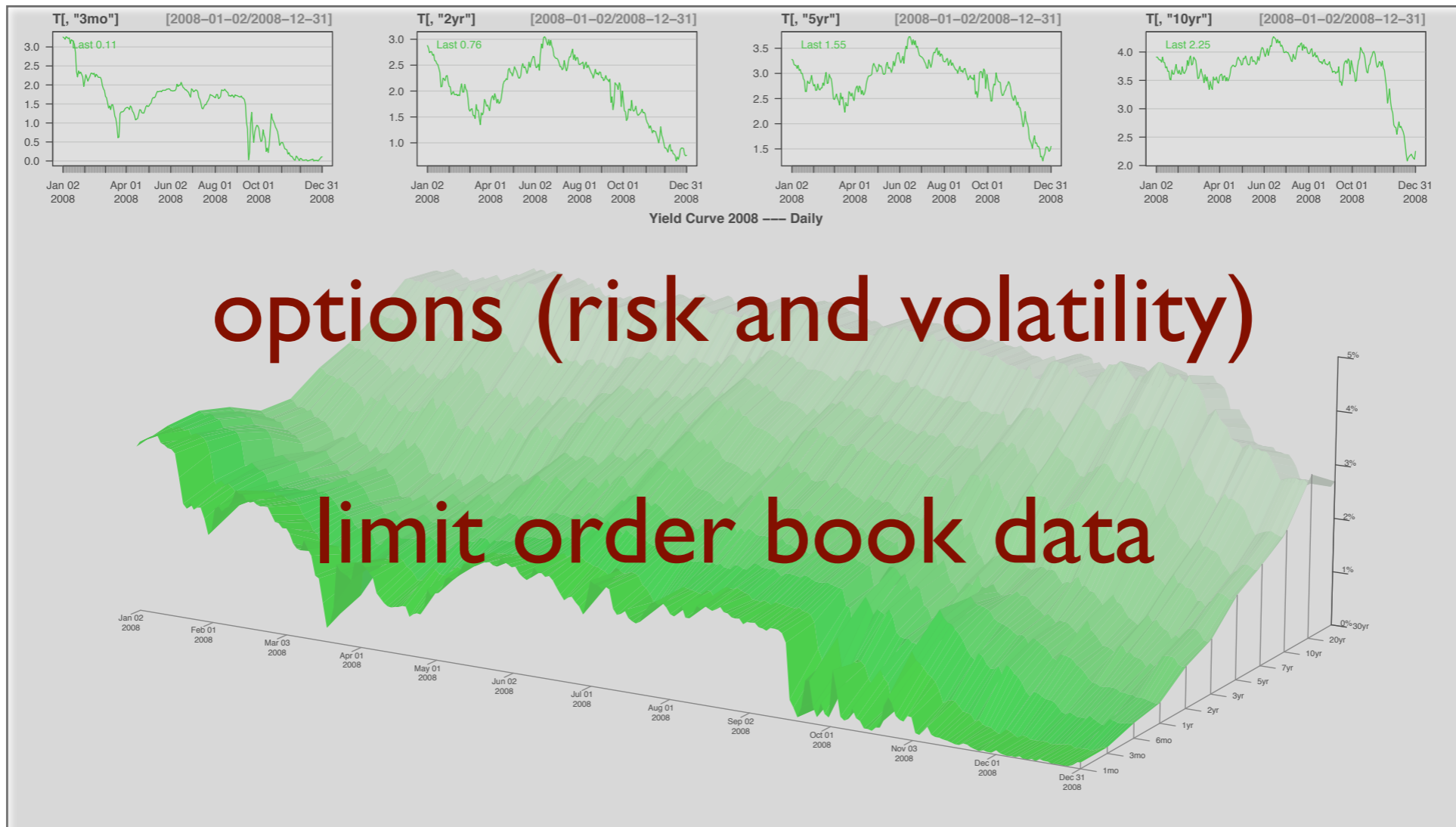


Data. Visualization. Modeling.

# chartSeries + chartSeries3d



Data. Visualization. Modeling.



options (risk and volatility)

limit order book data

## attachSymbols

New functionality to extend upon `getSymbols`

## attachSymbols

New functionality to extend upon `getSymbols`

Create a demand based database system using `getSymbols` that allows for implicit loading of an entire universe of symbols

## attachSymbols

Example: All US Equity symbols on demand.

```
> search()
[1] ".GlobalEnv"      "package:quantmod" "package:Defaults"
[4] "package:xts"     "package:zoo"      "package:stats"
[7] "package:graphics" "package:grDevices" "package:utils"
[10] "package:datasets" "package:methods" "Autoloads"
[13] "package:base"
```

## attachSymbols

Example: All US Equity symbols on demand.

```
> search()
[1] ".GlobalEnv"      "package:quantmod" "package:Defaults"
[4] "package:xts"     "package:zoo"      "package:stats"
[7] "package:graphics" "package:grDevices" "package:utils"
[10] "package:datasets" "package:methods" "Autoloads"
[13] "package:base"

> attachSymbols(DB=DDB_Yahoo(), pos=2, prefix="E.")
```

Contains symbols and method



## attachSymbols

Example: All US Equity symbols on demand.

```
> search()
[1] ".GlobalEnv"      "package:quantmod" "package:Defaults"
[4] "package:xts"     "package:zoo"      "package:stats"
[7] "package:graphics" "package:grDevices" "package:utils"
[10] "package:datasets" "package:methods" "Autoloads"
[13] "package:base"

> attachSymbols(DB=DDB_Yahoo(), pos=2, prefix="E.")

> search()
[1] ".GlobalEnv"      "DDB:Yahoo"        "package:quantmod"
[4] "package:Defaults" "package:xts"      "package:zoo"
[7] "package:stats"   "package:graphics" "package:grDevices"
[10] "package:utils"   "package:datasets" "package:methods"
[13] "Autoloads"      "package:base"
```

## attachSymbols

Example: All US Equity symbols on demand.

```
> search()
[1] ".GlobalEnv"      "package:quantmod" "package:Defaults"
[4] "package:xts"     "package:zoo"      "package:stats"
[7] "package:graphics" "package:grDevices" "package:utils"
[10] "package:datasets" "package:methods" "Autoloads"
[13] "package:base"

> attachSymbols(DB=DDB_Yahoo(), pos=2, prefix="E.")

> search()
[1] ".GlobalEnv"      "DDB:Yahoo"       "package:quantmod"
[4] "package:Defaults" "package:xts"     "package:zoo"
[7] "package:stats"   "package:graphics" "package:grDevices"
[10] "package:utils"  "package:datasets" "package:methods"
[13] "Autoloads"     "package:base"

> str(ls("DDB:Yahoo"))
chr [1:7406] "E.A" "E.AA" "E.AAC" "E.AACC" "E.AAI" "E.AAII" ...
```

## attachSymbols

Example: All US Equity symbols on demand.

```
> search()
[1] ".GlobalEnv"      "package:quantmod" "package:Defaults"
[4] "package:xts"     "package:zoo"      "package:stats"
[7] "package:graphics" "package:grDevices" "package:utils"
[10] "package:datasets" "package:methods" "Autoloads"
```

7406 symbols are available

```
[1] ".GlobalEnv"      "DDB:Yahoo"        "package:quantmod"
[4] "package:Defaults" "package:xts"      "package:zoo"
[7] "package:stats"    "package:graphics" "package:grDevices"
[10] "package:utils"   "package:datasets" "package:methods"
[13] "Autoloads"       "package:base"

> str(ls("DDB:Yahoo"))
chr [1:7406] "E.A" "E.AA" "E.AAC" "E.AACC" "E.AAI" "E.AAII" ...
```

## attachSymbols

Example: All US Equity symbols on demand.

```
> str(E.A)
An 'xts' object from 2007-01-03 to 2009-03-23 containing:
  Data: num [1:559, 1:6] 35 34.3 34.3 34 34.1 ...
- attr(*, "dimnames")=List of 2
  ..$ : NULL
  ..$ : chr [1:6] "A.Open" "A.High" "A.Low" "A.Close" ...
Indexed by objects of class: [POSIXt,POSIXct] TZ:America/
Chicago
xts Attributes:
List of 2
 $ src   : chr "yahoo"
 $ updated: POSIXct[1:1], format: "2009-03-24 10:59:14"
```

## attachSymbols

Example: All US Equity symbols on demand.

```
> str(E.A)
An 'xts' object from 2007-01-03 to 2009-03-23 containing:
  Data: num [1:559, 1:6] 35 34.3 34.3 34 34.1 ...
- attr(*, "dimnames")=List of 2
  ..$ : NULL
  ..$ : chr [1:6] "A.Open" "A.High" "A.Low" "A.Close" ...
Indexed by objects of class: [POSIXt,POSIXct] TZ:America/
Chicago
xts Attributes:
List of 2
 $ src   : chr "yahoo"
 $ updated: POSIXct[1:1], format: "2009-03-24 10:59:14"
```

First access loads data

## attachSymbols

Example: All US Equity symbols on demand.

```
> system.time(E.AKAM)
user system elapsed
0.032  0.004  0.267
```



download from Yahoo

## attachSymbols

Example: All US Equity symbols on demand.

```
> system.time(E.AKAM)
user system elapsed
0.032 0.004 0.267
```

```
> system.time(E.AKAM)
user system elapsed
0 0 0
```



subsequent calls from cache

## attachSymbols

### Two Methods to Cache

#### Disk

after first access,  
objects are  
cached to disk.  
`makeActiveBinding`

#### Memory

after first access  
objects remain in  
memory  
`delayedAssign`



## attachSymbols

### Custom DDB methods

example: DDB:Yahoo

- > DDB\_Yahoo()
- > # creates DDB of all US Equity Symbols

## attachSymbols

### Custom DDB methods

example: DDB:Yahoo

- > attachSymbols()
- > # “binds” symbols to functions to load/reload

## attachSymbols

### Custom DDB methods

example: DDB:Yahoo

All symbols are attached to new environment and accessible on demand.

## Future Work

Integration of trading/testing with **blotter** package


More data methods, easier to extend

**specifyModel** - **buildModel** - **tradeModel** work

# More Information

[www.insightalgo.com](http://www.insightalgo.com)

[www.quantmod.com](http://www.quantmod.com)



**insight algorithmics™**


**Quantitative Software and Consulting**

insight algorithmics™ specializes in custom quantitative software solutions for small to mid-sized Proprietary Trade Desks, Hedge Funds, Fund of Funds and Commodity Trading Advisors.

We currently provide services covering:

- Quantitative Trading
- Visualization
- Open-Source Integration
- Data Integration & Management
- Time Series Analysis
- Algorithm Development
- On-Site Training and Consulting for R
- Expert R, Python, C and Fortran Development

[Contact](#) us today for a confidential consultation.

```
return.class("cta", ...)  
library("quantmod")  
this.env <- environment()  
var <- rnorm(100, list(...))  
# report all named elements  
assign(var, list(...))  
if (missing(verbose)) verb  
if (missing(auto.assign))  
FRED_URL <- "http://fred.stlouisfed.org" </pre>

| Date       | Open  | High  | Low   | Close |
|------------|-------|-------|-------|-------|
| 2008-01-02 | 18.24 | 18.34 | 18.14 | 18.24 |
| 2008-01-03 | 18.32 | 18.42 | 18.22 | 18.32 |
| 2008-01-06 | 18.37 | 18.47 | 18.27 | 18.37 |
| 2008-01-07 | 18.35 | 18.45 | 18.25 | 18.35 |
| 2008-01-08 | 18.27 | 18.37 | 18.17 | 18.27 |
| 2008-01-09 | 18.22 | 18.32 | 18.12 | 18.22 |
| 2008-01-10 | 18.14 | 18.24 | 18.04 | 18.14 |
| 2008-01-13 | 18.11 | 18.21 | 18.01 | 18.11 |
| 2008-01-14 | 18.11 | 18.21 | 18.01 | 18.11 |
| 2008-01-15 | 18.11 | 18.21 | 18.01 | 18.11 |
| 2008-01-16 | 18.11 | 18.21 | 18.01 | 18.11 |
| 2008-01-17 | 18.11 | 18.21 | 18.01 | 18.11 |
| 2008-01-21 | 18.11 | 18.21 | 18.01 | 18.11 |
| 2008-01-22 | 18.11 | 18.21 | 18.01 | 18.11 |
| 2008-01-23 | 18.11 | 18.21 | 18.01 | 18.11 |
| 2008-01-24 | 18.11 | 18.21 | 18.01 | 18.11 |


```

## quantmod

Quantitative Financial Modelling & Trading Framework for R

- quantmod
- news
- what's next
- documentation
- examples
- gallery
- download
- license
- feeds
- R/quant links
- add to del.icio.us

**{ quantmod }**

The **quantmod** package for R is designed to assist the quantitative trader in the development, testing, and deployment of statistically based trading models.

**What quantmod IS**

A rapid prototyping environment, where quant traders can quickly and cleanly explore and build trading models.

**What quantmod is NOT**


A replacement for anything statistical. It has no 'new' modelling routines or analysis tool to speak of. It *does* now offer **charting** not currently available elsewhere in R, but most everything else is more of a wrapper to what you already know and love about the language and packages you currently use.

quantmod makes modelling easier by removing the repetitive workflow issues surrounding data management, modelling interfaces, and performance analysis.

Explore what is currently possible in the [examples](#)

This software is written and maintained by Jeffrey A. Ryan. See license for details on copying and use. Copyright 2008.

Updated Charting Tools for 0.3-6!



Presented by Jeffrey A. Ryan [jeffrey.ryan@insightalgo.com](mailto:jeffrey.ryan@insightalgo.com)

[www.quantmod.com/Vienna2009](http://www.quantmod.com/Vienna2009)