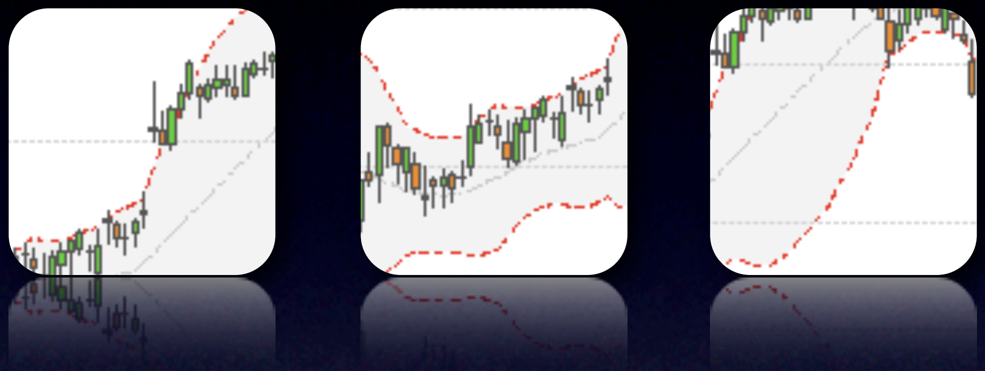


quantmod + xts

Presented by Jeffrey A. Ryan jeffrey.ryan@insightalgo.com

Computational Finance with R
Columbia University, New York
December 4, 2008

www.quantmod.com/Columbia2008

quantmod

Original Purpose:
Provide a unified interface to R for quantitative traders who are tired of Excel®

(Data)+(Visualization)+(Modelling)

quantmod

Original Purpose:
Provide a unified interface to R for quantitative traders who are tired of Excel®

(Data)+(Visualization)+(Modelling)

(Visualization)

Requirements

- Provide full financial charting abilities to R
- Allow interaction with charts
- Simple and fast interface and execution

(Visualization)

Requirements

- Provide full financial charting abilities to R
- Allow interaction with charts
- Simple and fast interface and execution

(Visualization)

Requirements

- Provide full financial charting abilities to R
- Allow interaction with charts
- Simple and fast interface and execution

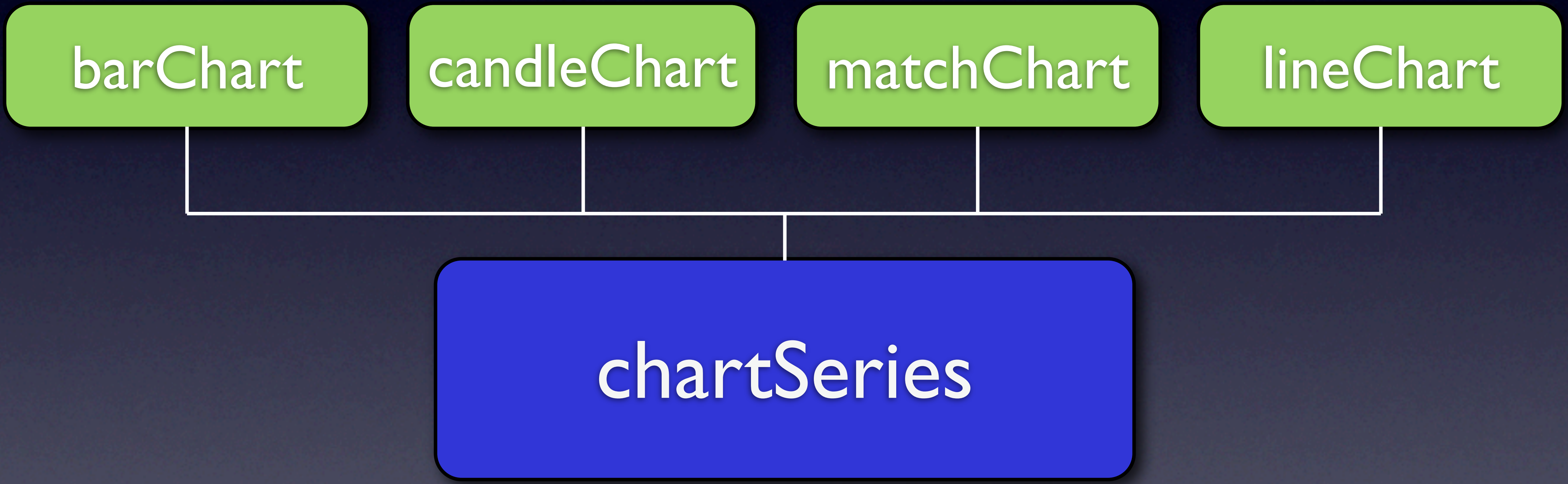
(Visualization)

Requirements

- Provide full financial charting abilities to R
- Allow interaction with charts
- Simple and fast interface and execution

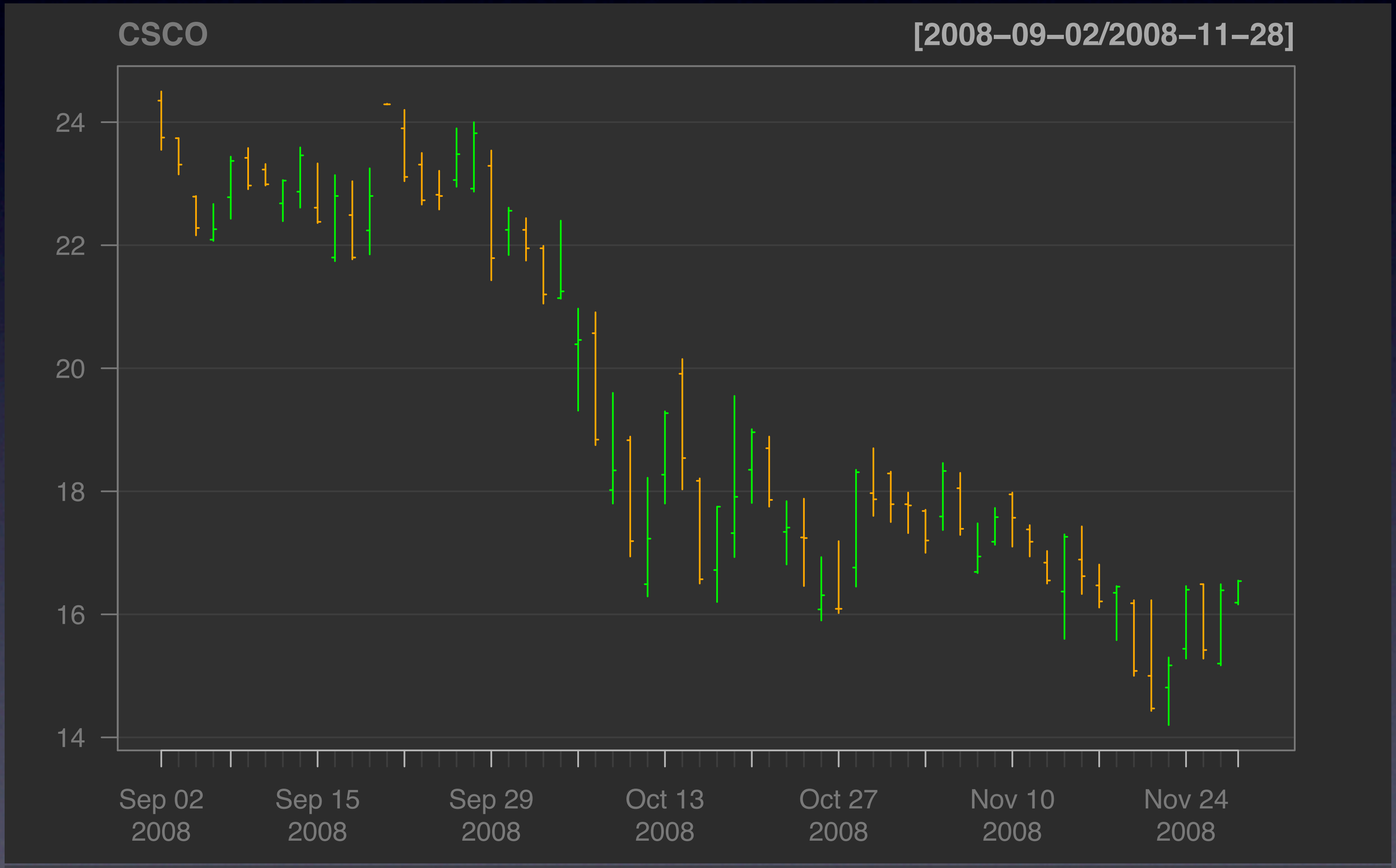
(Visualization)

The Basics



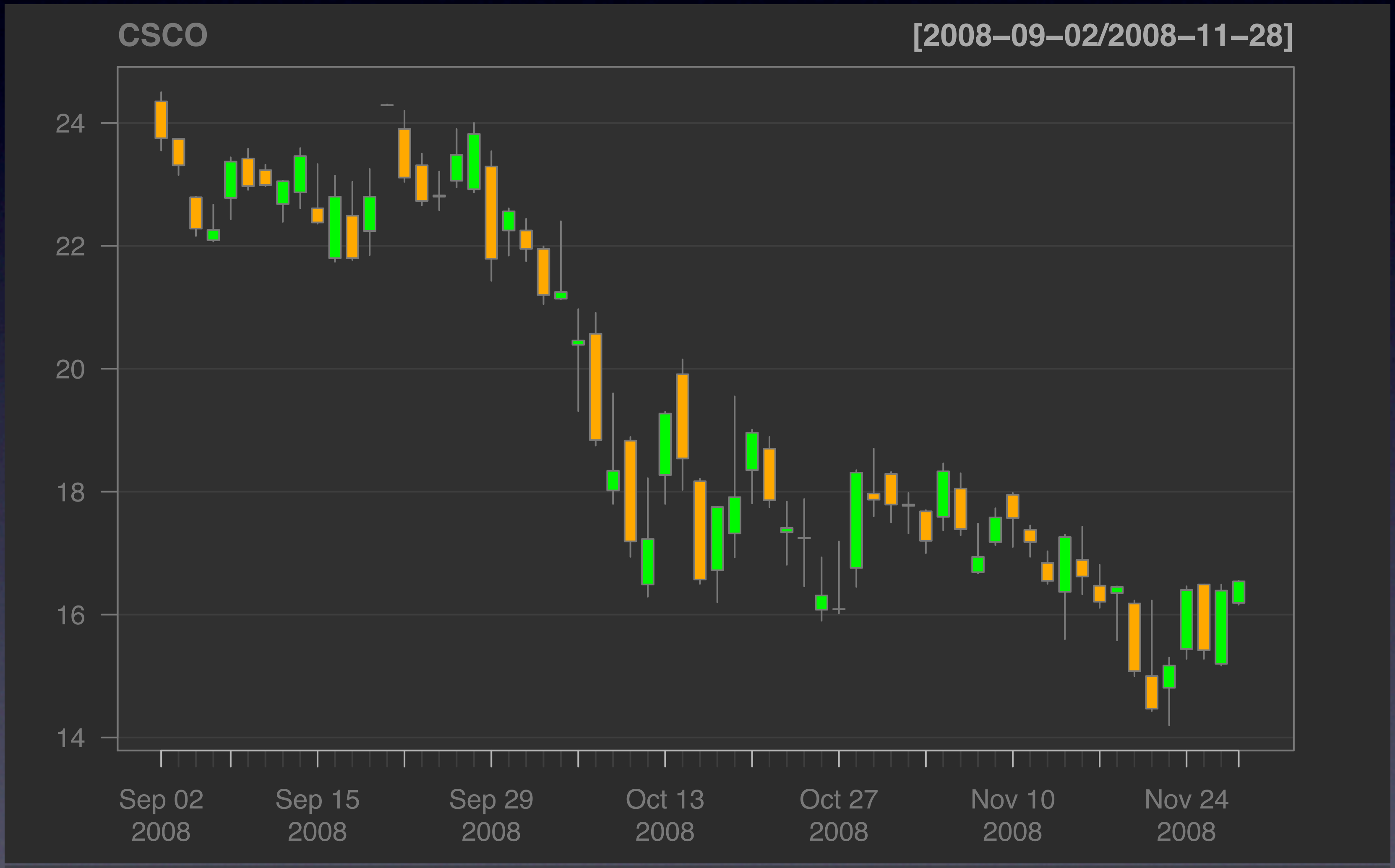
(Visualization)

barChart



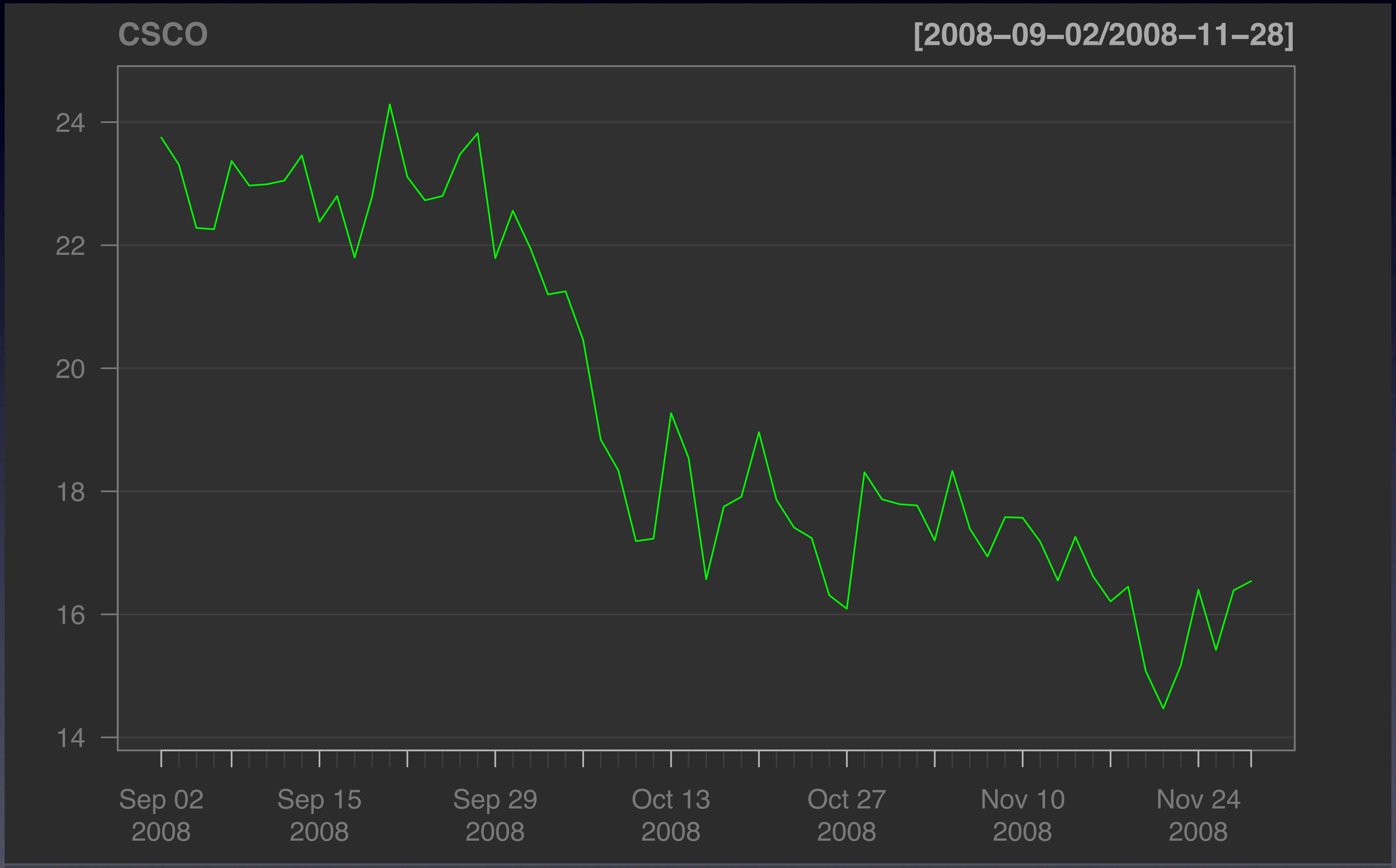
(Visualization)

candleChart



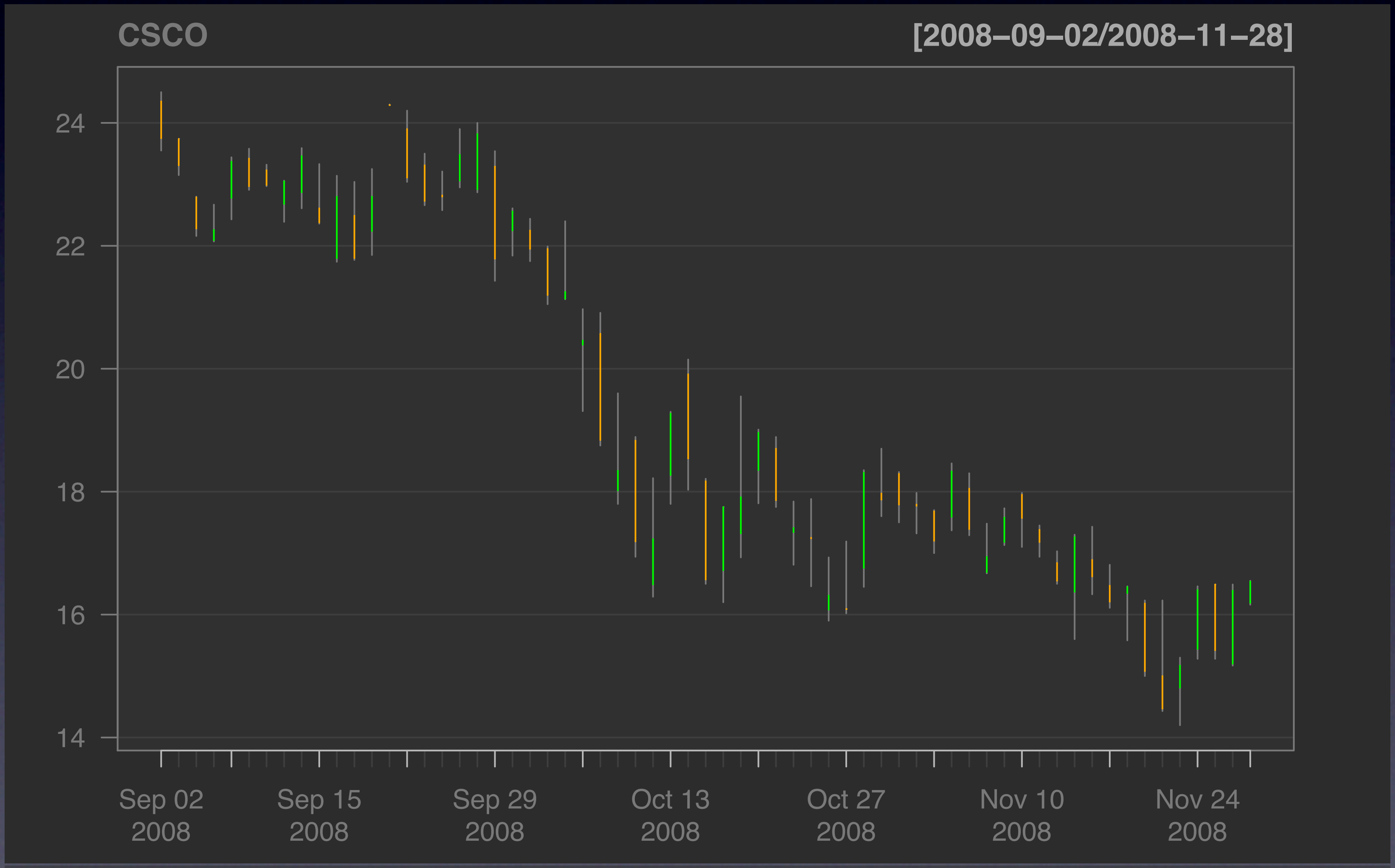
(Visualization)

lineChart



(Visualization)

matchChart



(Visualization)

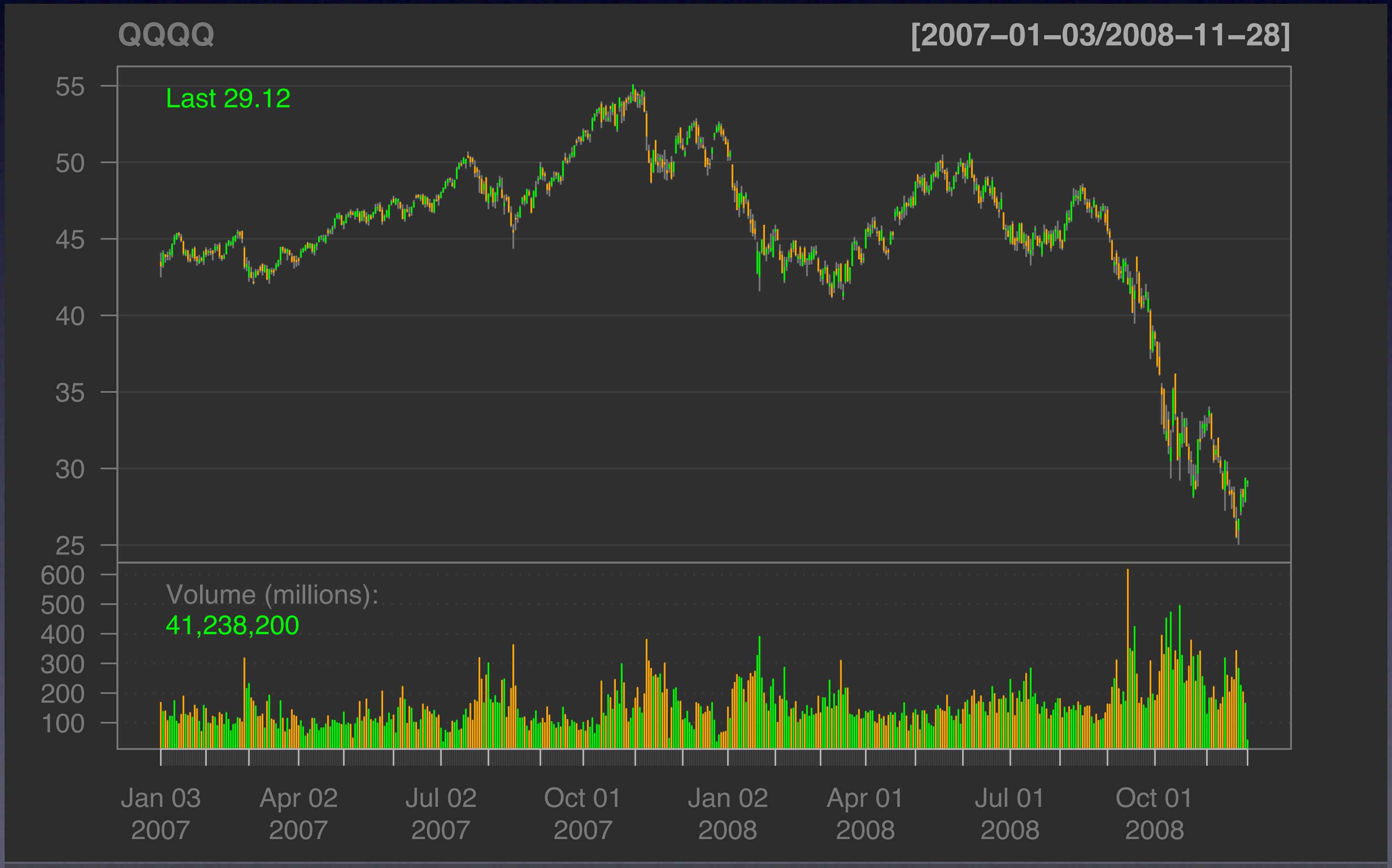
Creating charts is easy

```
getSymbols("QQQQ"); chartSeries(QQQQ)
```

(Visualization)

Creating charts is easy

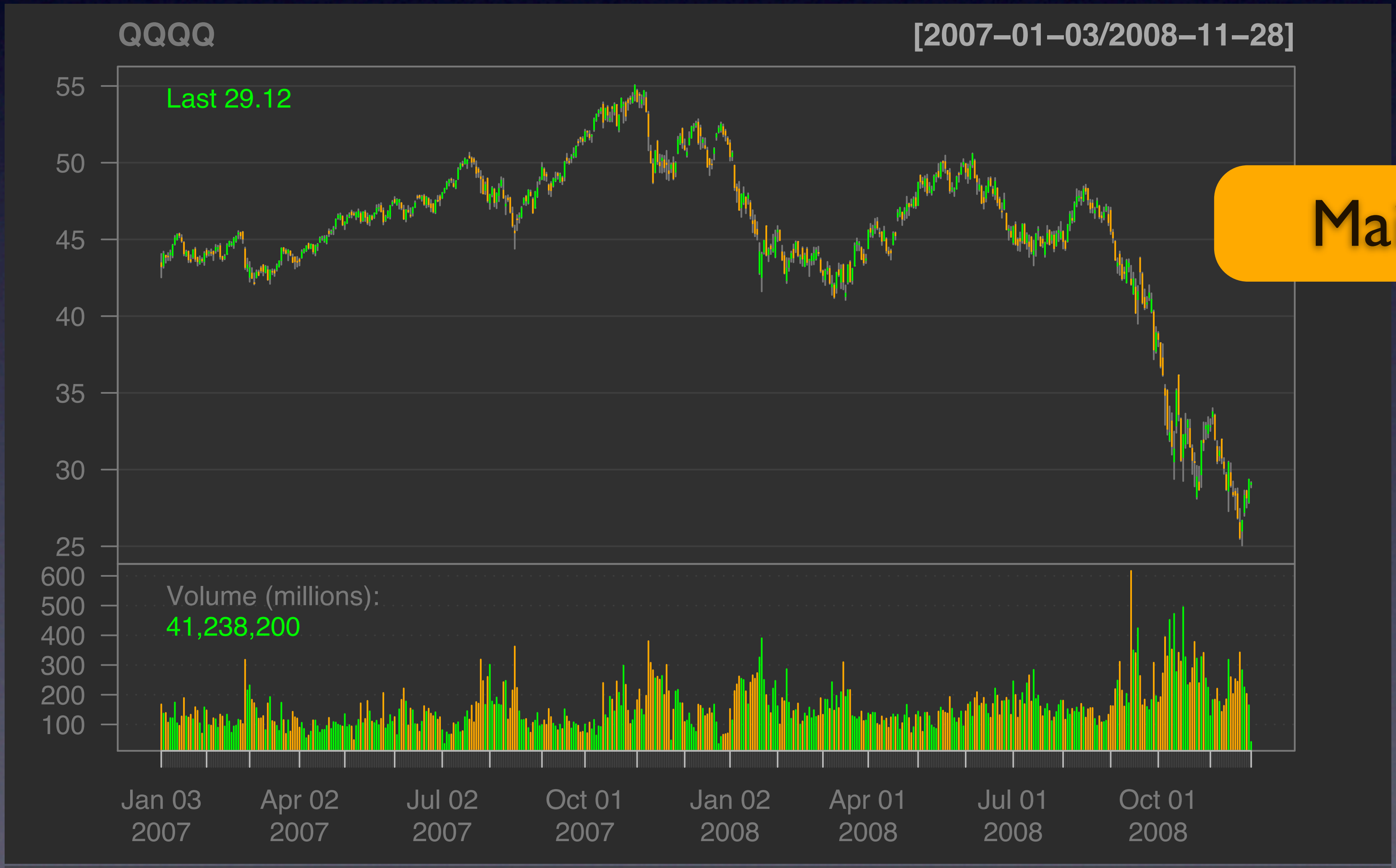
getSymbols("QQQQ"); chartSeries(QQQQ)



(Visualization)

Creating charts is easy

getSymbols(“QQQQ”); chartSeries(QQQQ)

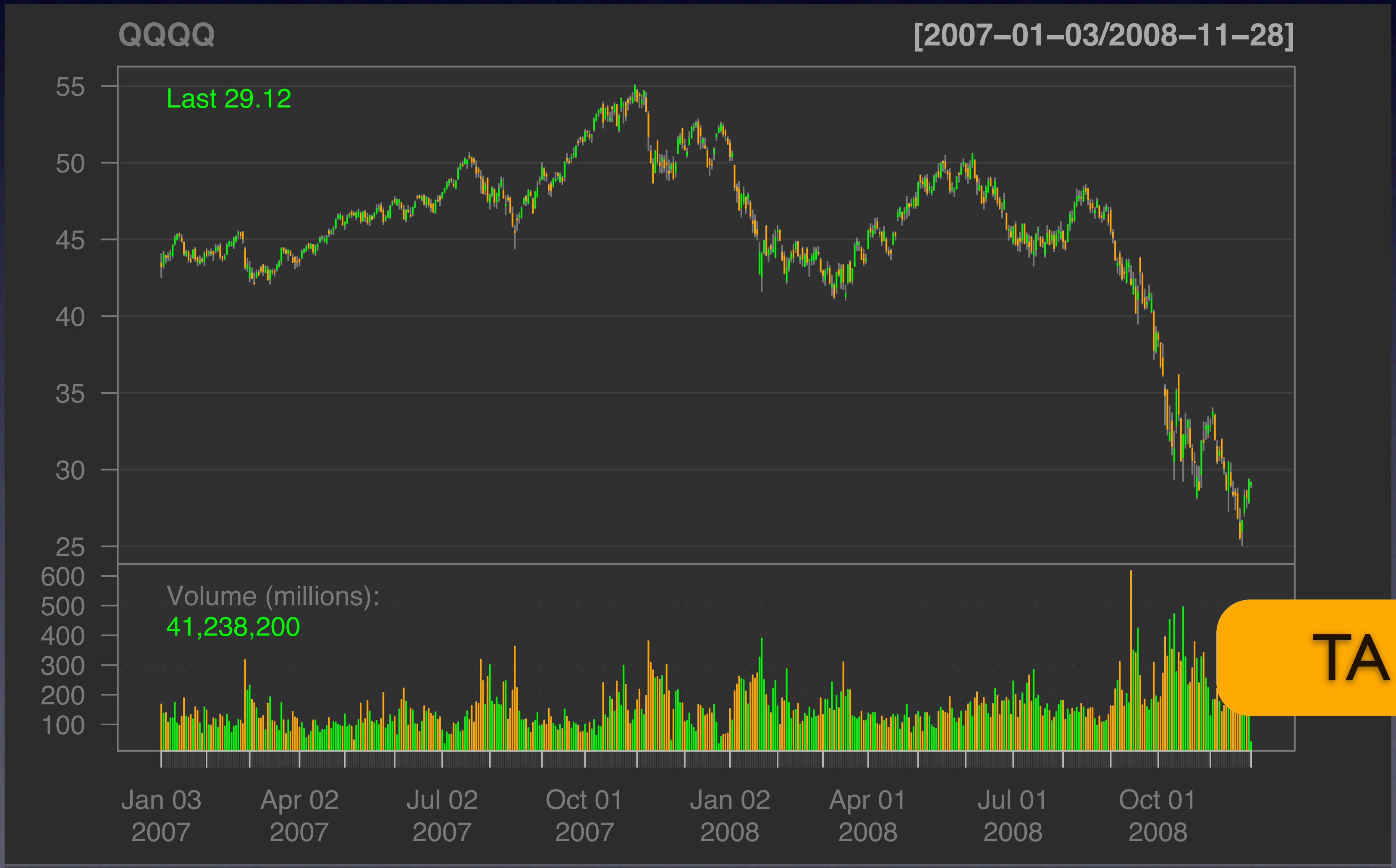


Main Series

(Visualization)

Creating charts is easy

getSymbols(“QQQQ”); chartSeries(QQQQ)



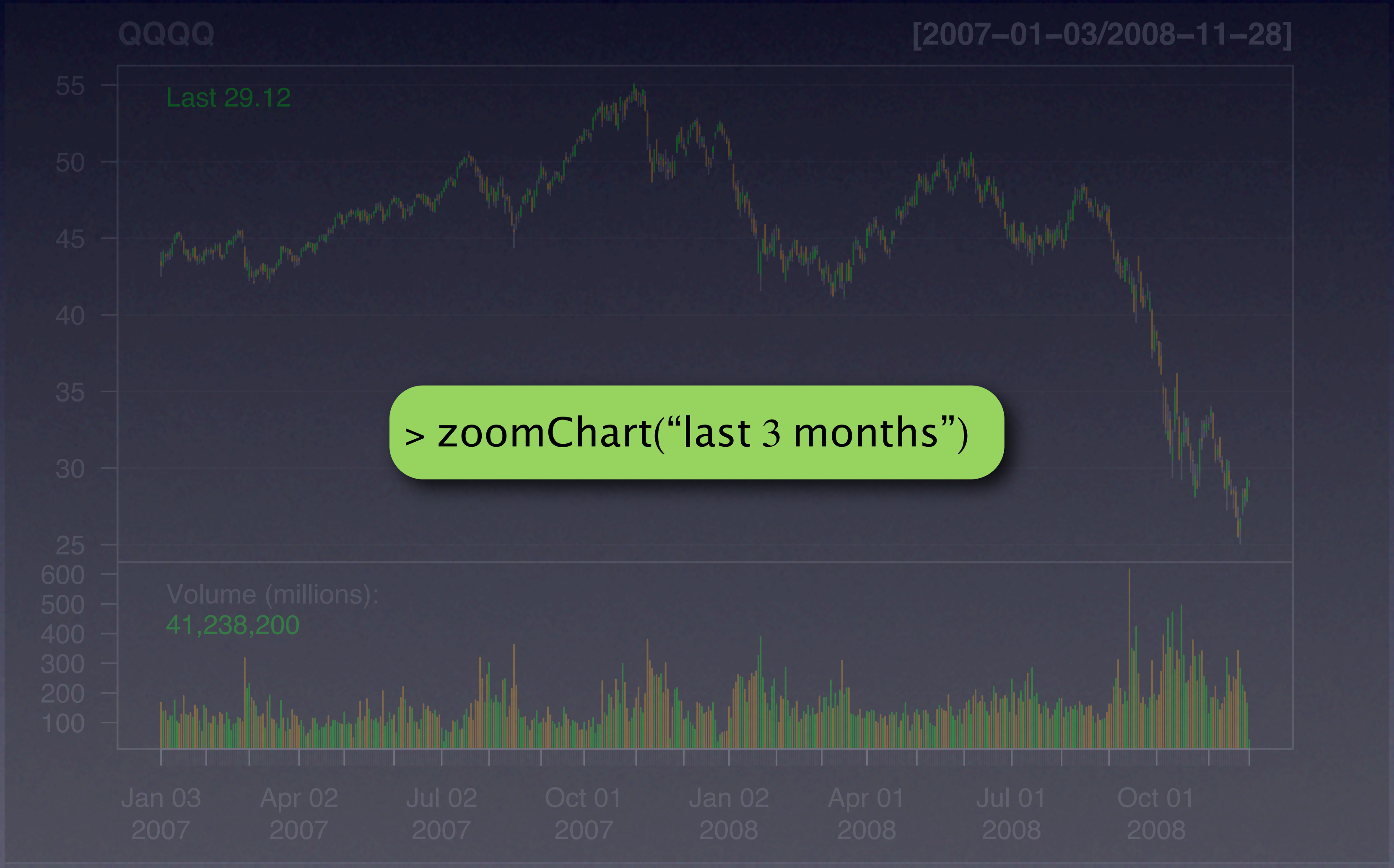
TA window

(Visualization)

QQQQ
[2007−01−03/2008−11−28]
Last 29.12
55
50
45
40
30
25
600
500
400
300
200
100
Volume (millions):
41,238,200
Jan 03 2007
Apr 02 2007
Jul 02 2007
Oct 01 2007
Jan 02 2008
Apr 01 2008
Jul 01 2008
Oct 01 2008

Change perspective and style!

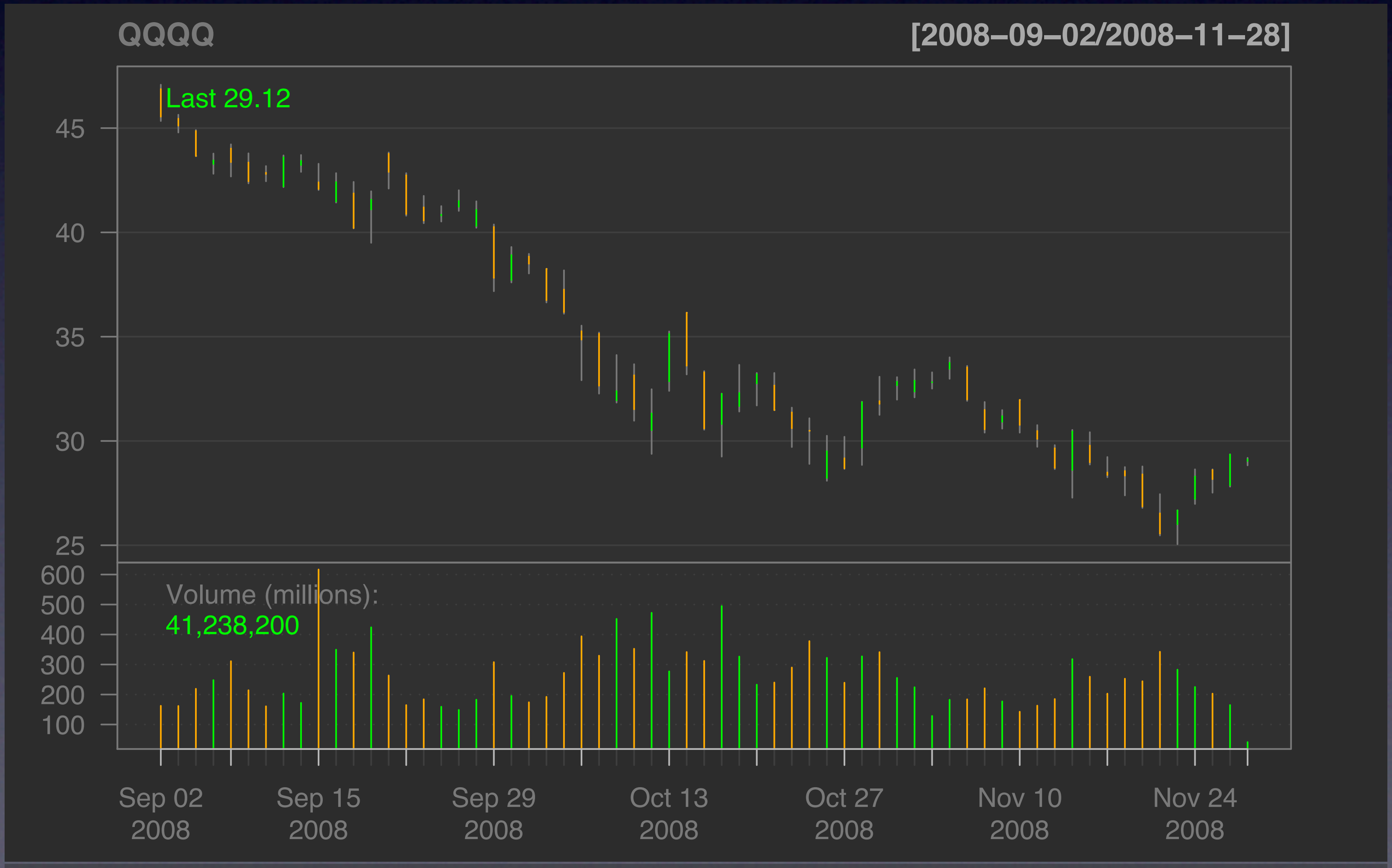
(Visualization)



```
> zoomChart("last 3 months")
```

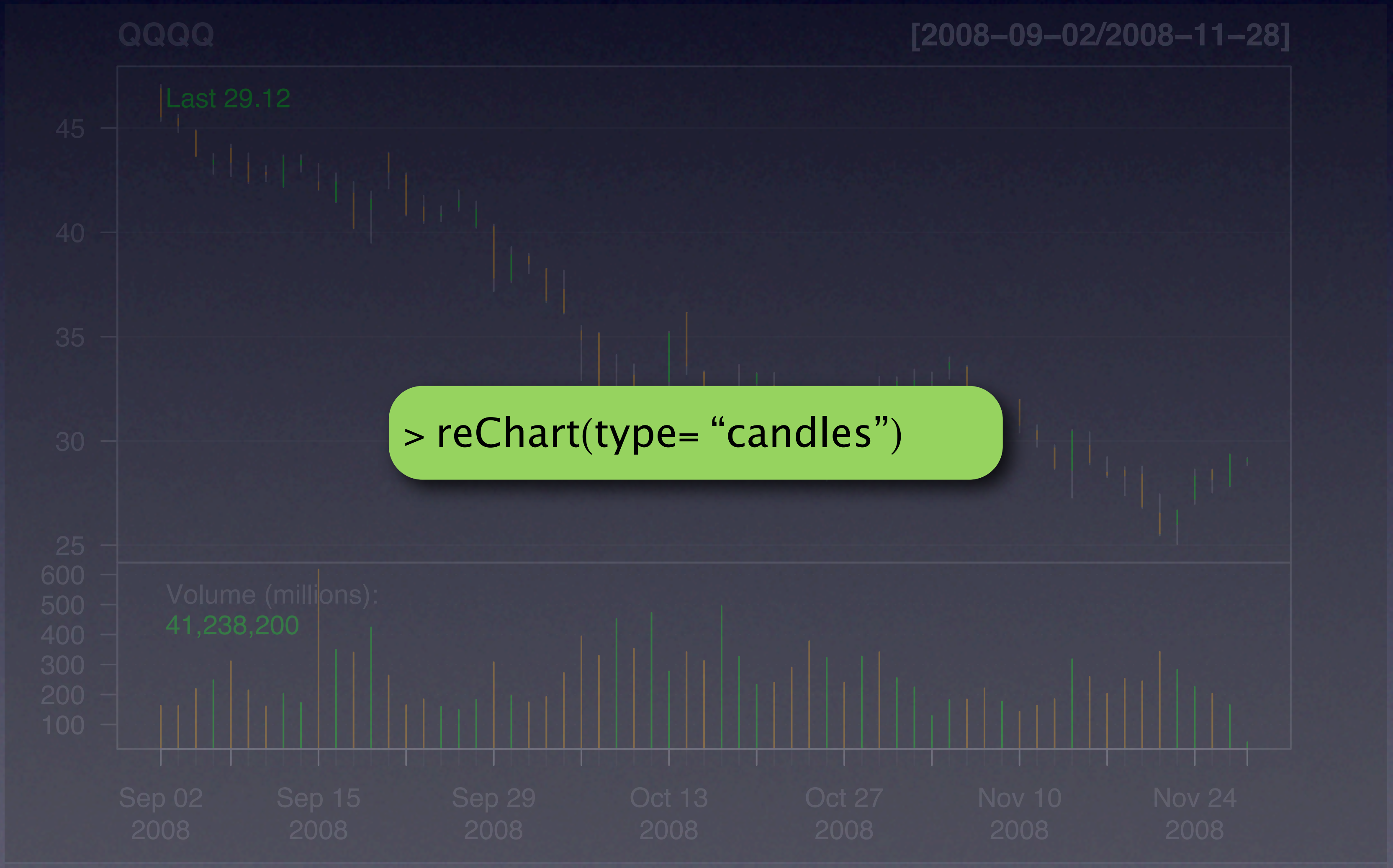
(Visualization)

Zoomed to "last 3 months"



(Visualization)

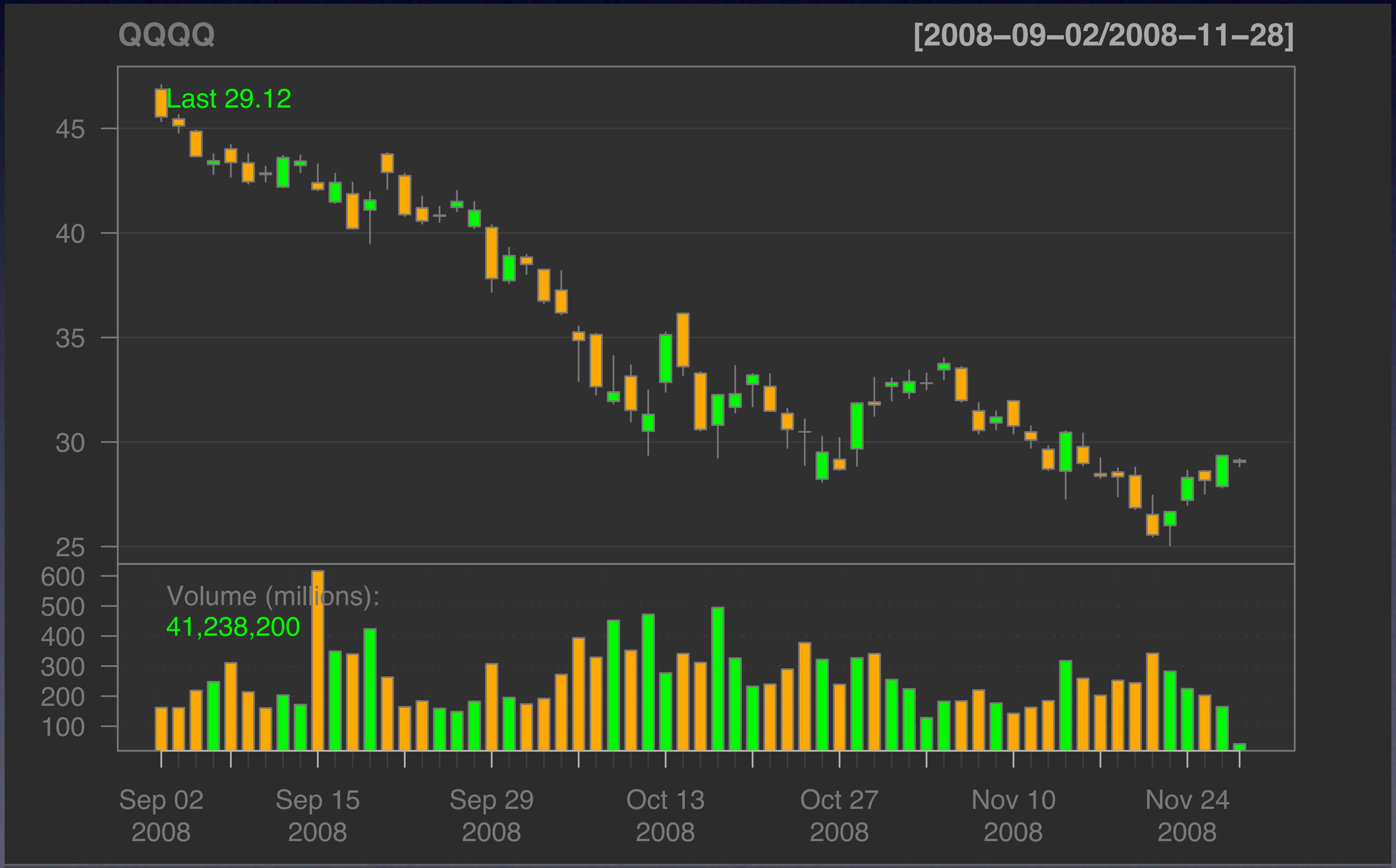
Switch to candlesticks



```
> reChart(type= “candles”)
```

(Visualization)

done!



(Visualization)

40+ Built-in Technical Indicators

(Visualization)

40+ Built-in Technical Indicators

Built-in TA functionality from quantmod and TTR

addADX	addATR	addAroon	addAroonOsc	addBBands
addCCI	addCLV	addCMF	addCMO	addChAD
addChVol	addDEMA	addDPO	addEMA	addEMV
addEnvelope	addEVWMA	addExpiry	addKST	addLines
addMACD	addMFI	addMomentum	addOBV	addPoints
addROC	addRSI	addSAR	addSMA	addSMI
addShading	addTDI	addTRIX	addVo	addVolatility
addWMA	addWPR	addZLEMA	addZigZag	... and more!

(Visualization)

40+ Built-in Technical Indicators

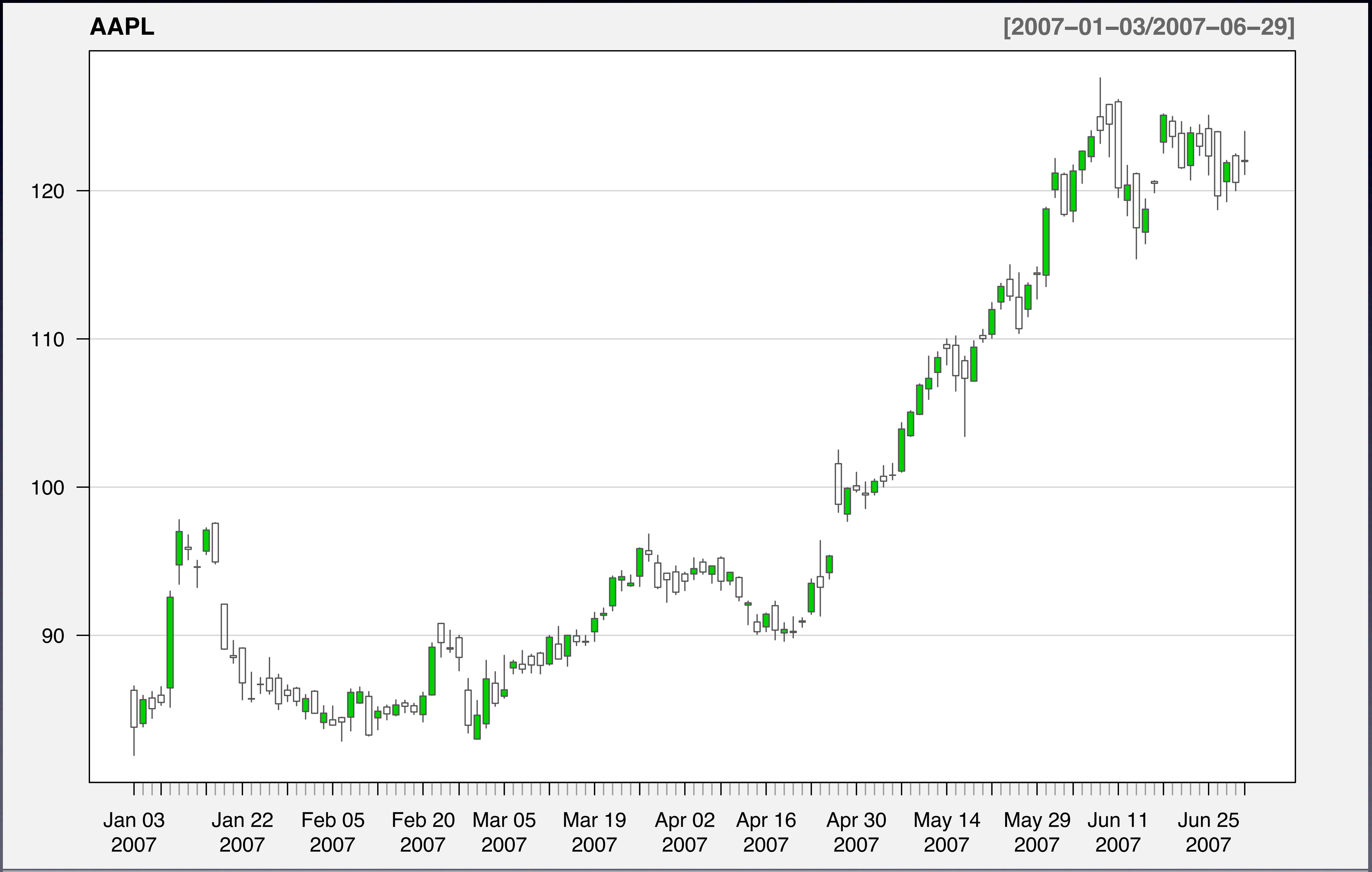
Built-in TA functionality from quantmod and TTR

addADX	addATR	addAroon	addAroonOsc	addBBands
addCCI	addCLV	addCMF	addCMO	addChAD
addChVol	addDEMA	addDPO	addEMA	addEMV
addEnvelope	addEVWMA	addExpiry	addKST	addLines
addMACD	addMFI	addMomentum	addOBV	addPoints
addROC	addRSI	addSAR	addSMA	addSMI
addShading	addTDI	addTRIX	addVo	addVolatility
addWMA	addWPR	addZLEMA	addZigZag	... and more!

Easy to add to charts

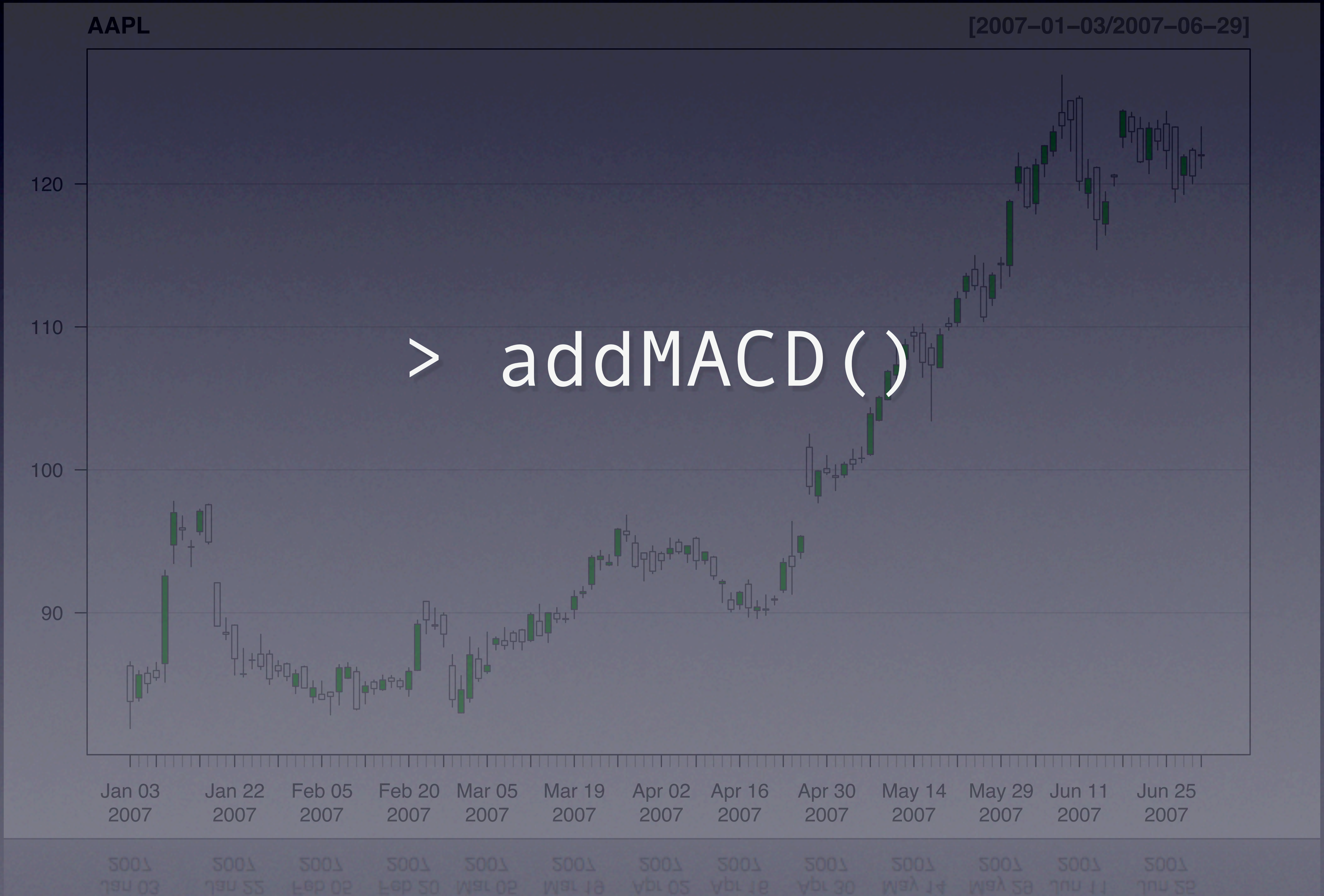
(Visualization)

Start with a chart of AAPL (in happier times)



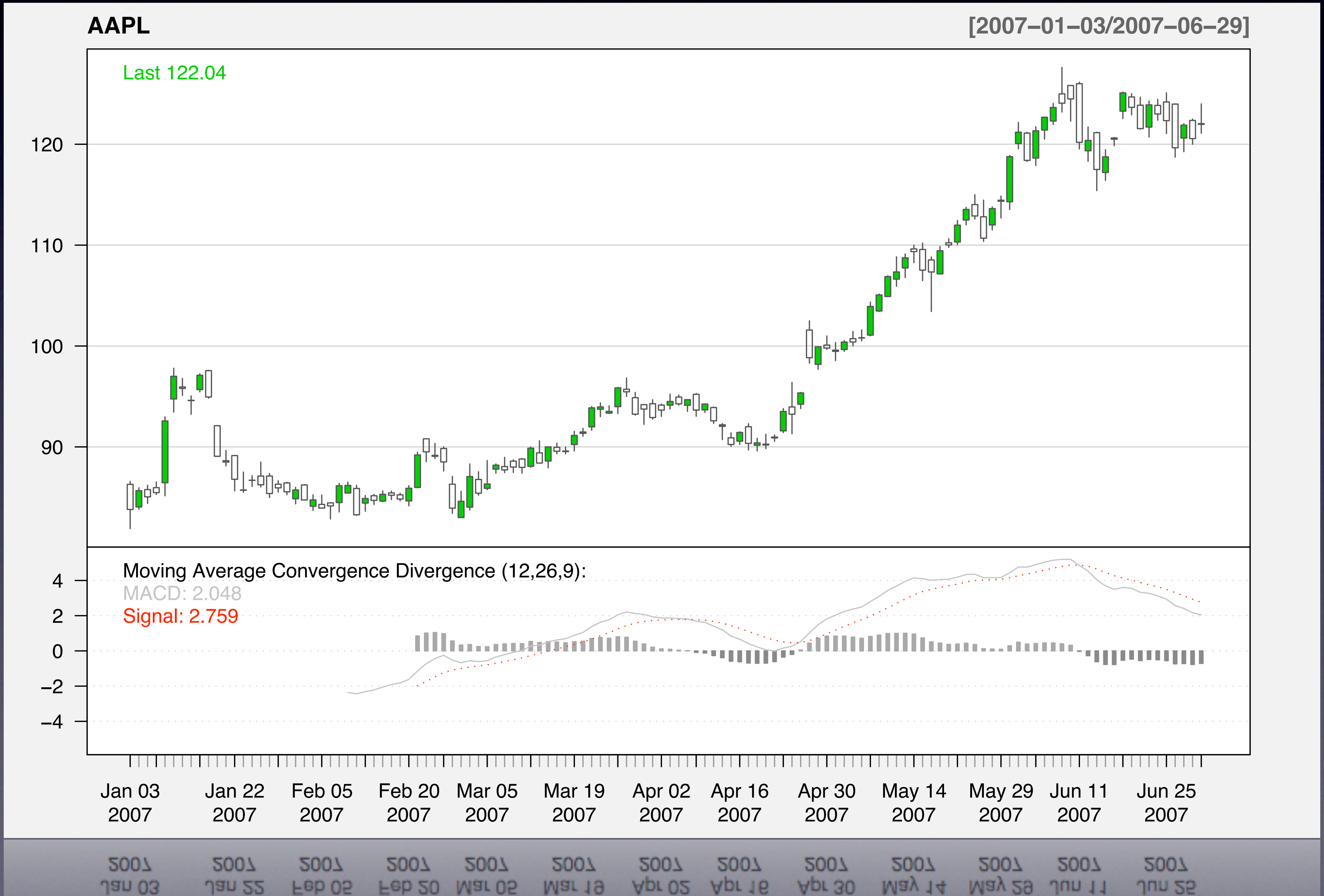
(Visualization)

...add Moving Average Convergence Divergence



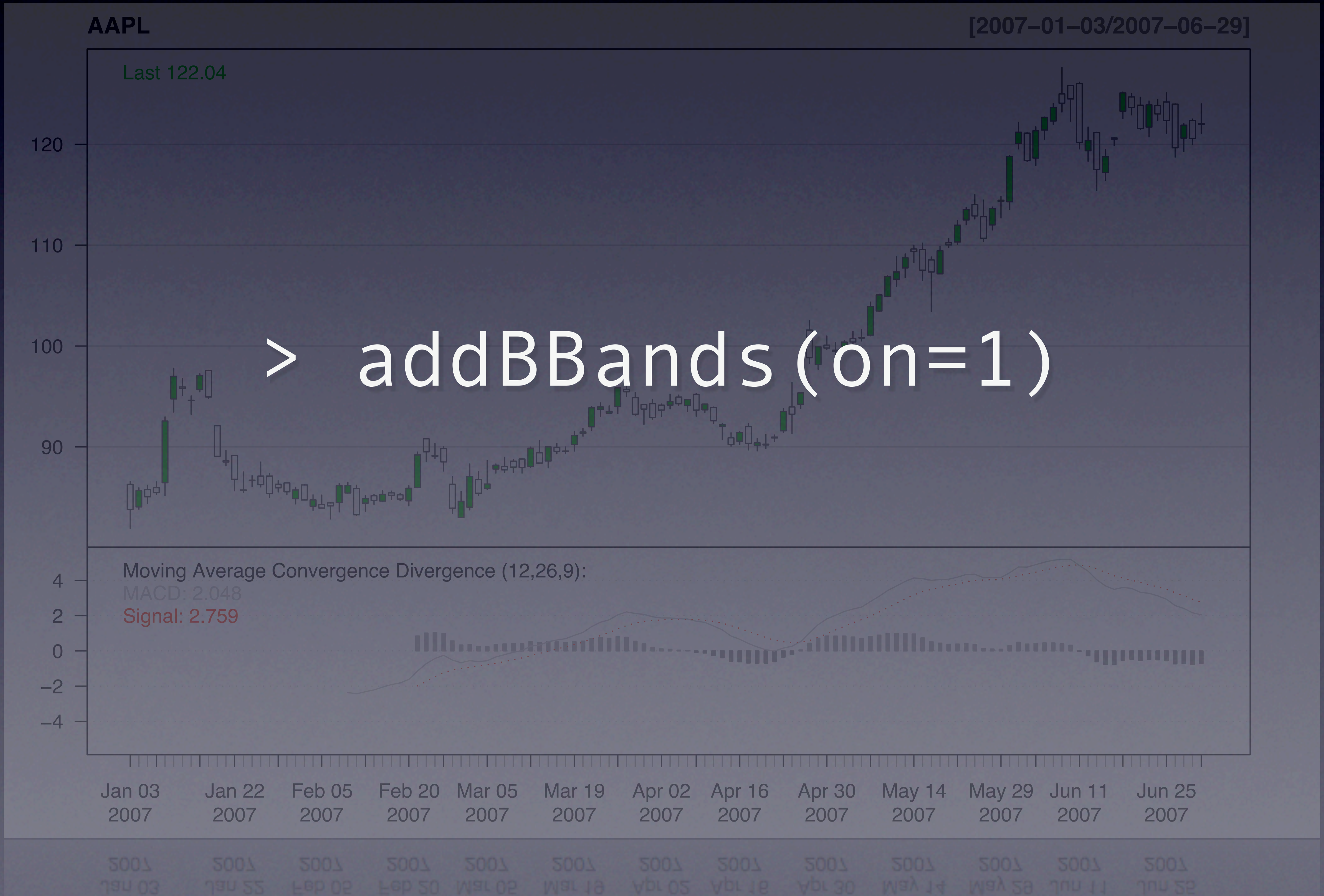
```
> addMACD()
```

(Visualization)



(Visualization)

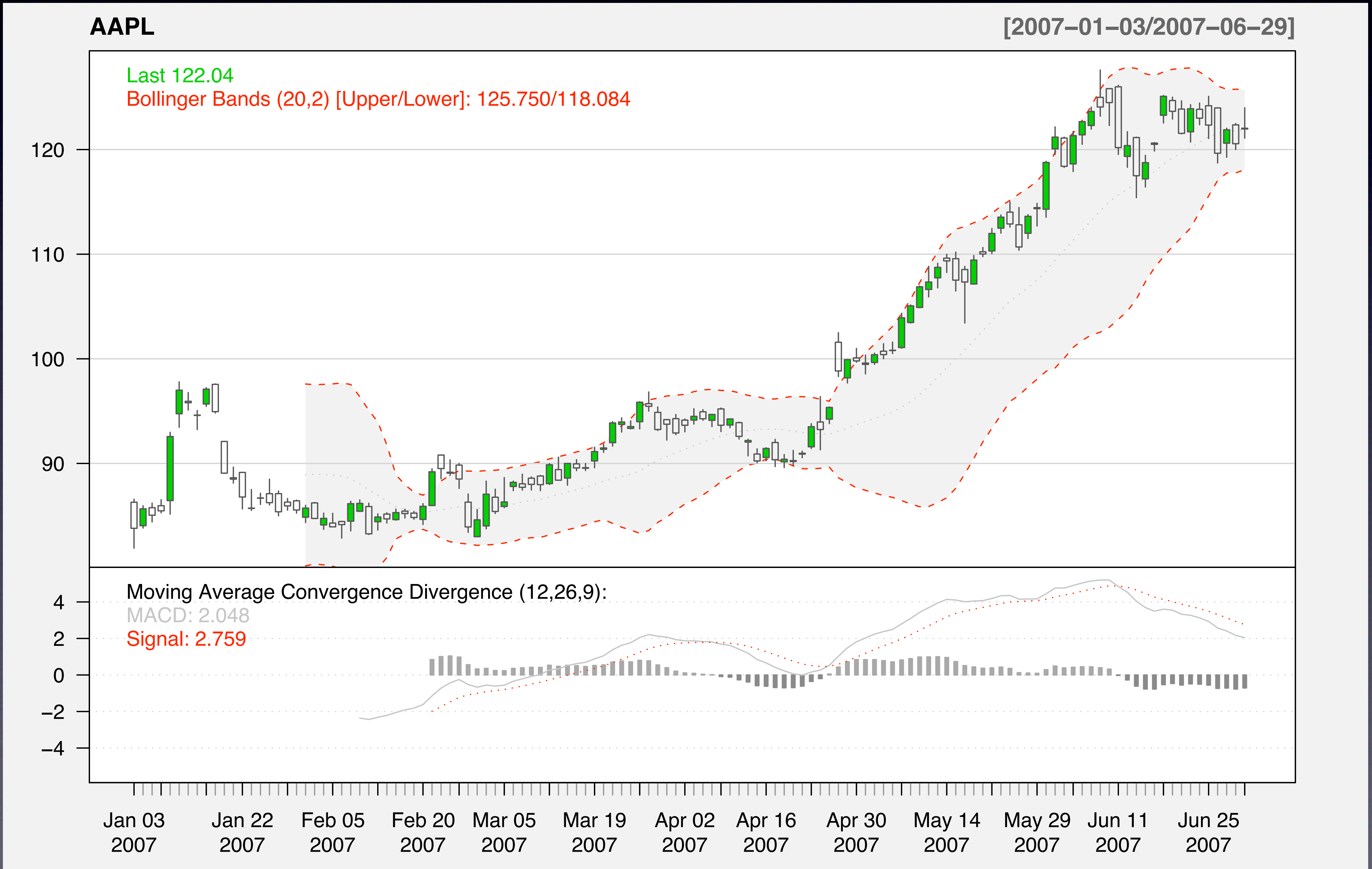
...add Bollinger Bands



```
>  addBBands(on=1)
```

(Visualization)

done!



(Visualization)

Customizing: setTA, theme and layout

(Visualization)

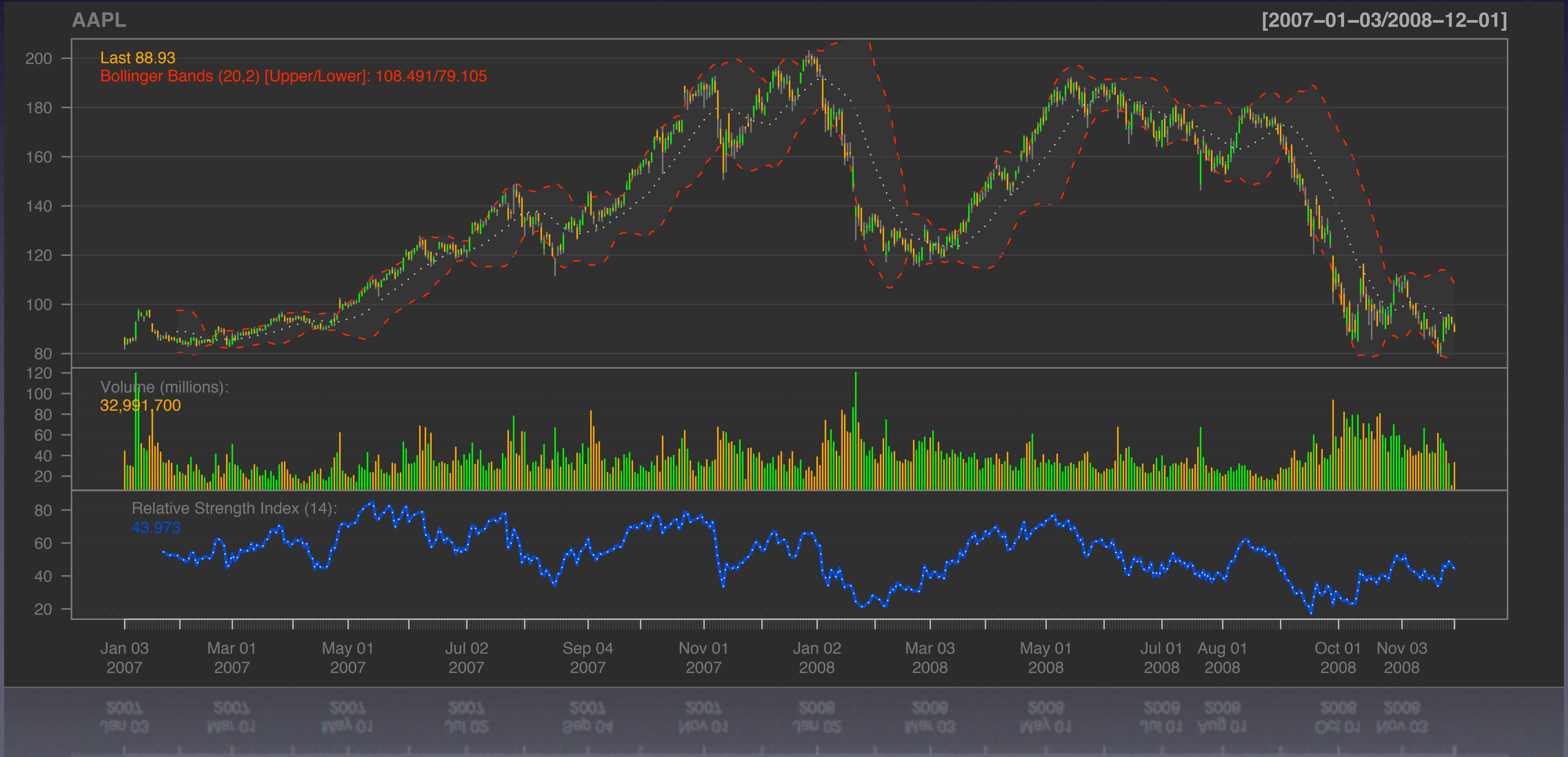
Customizing: setTA, theme and layout

```
> chartSeries(AAPL, TA= “addVo();addRSI()”)
> addBBands()
```

(Visualization)

Customizing: setTA, theme and layout

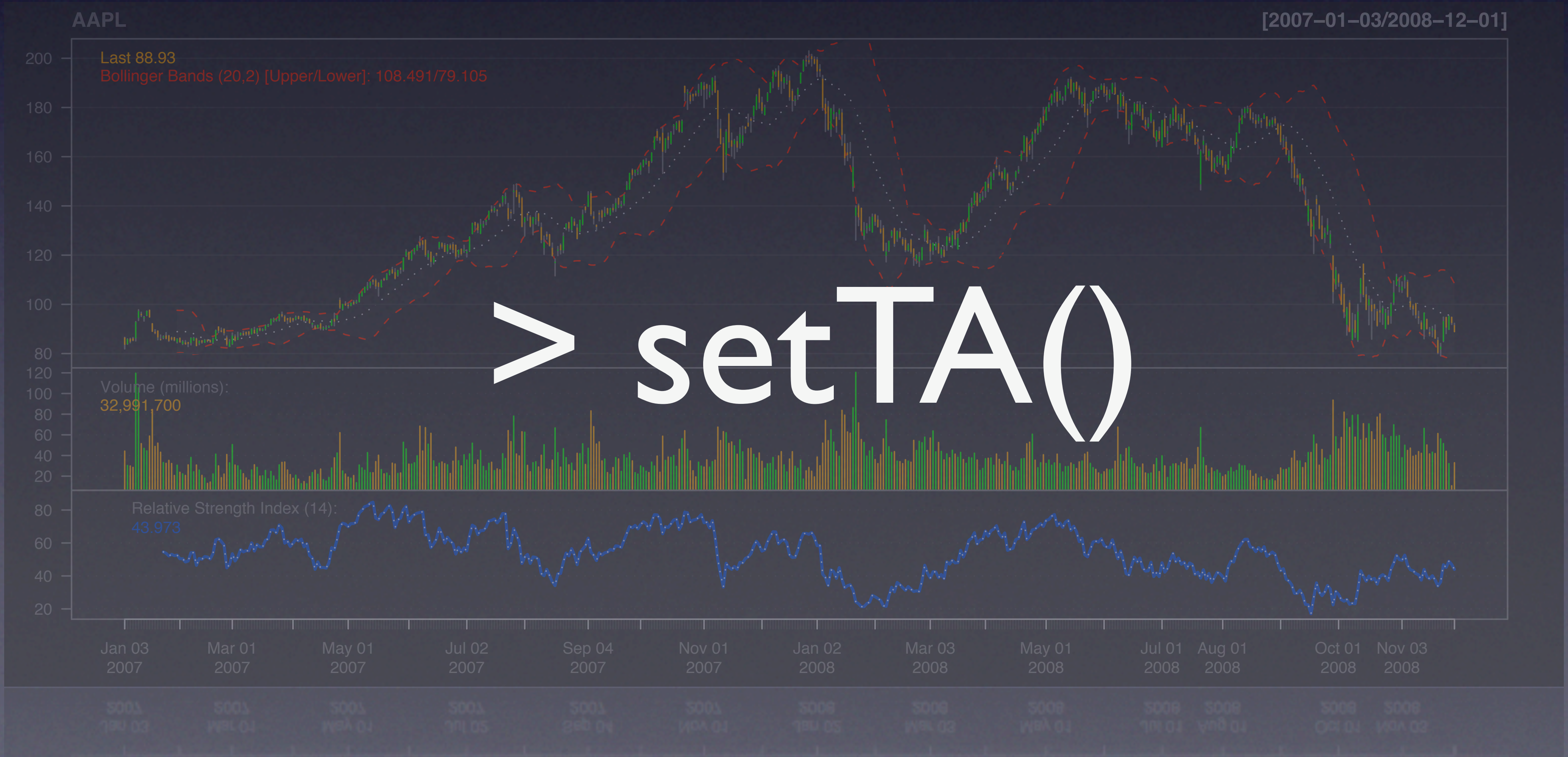
```
> chartSeries(AAPL, TA= “addVo();addRSI()”)
> addBBands()
```



(Visualization)

Customizing: setTA, theme and layout

```
> chartSeries(AAPL, TA= “addVo();addRSI()”)
> addBBands()
```



```
> setTA()
```

(Visualization)

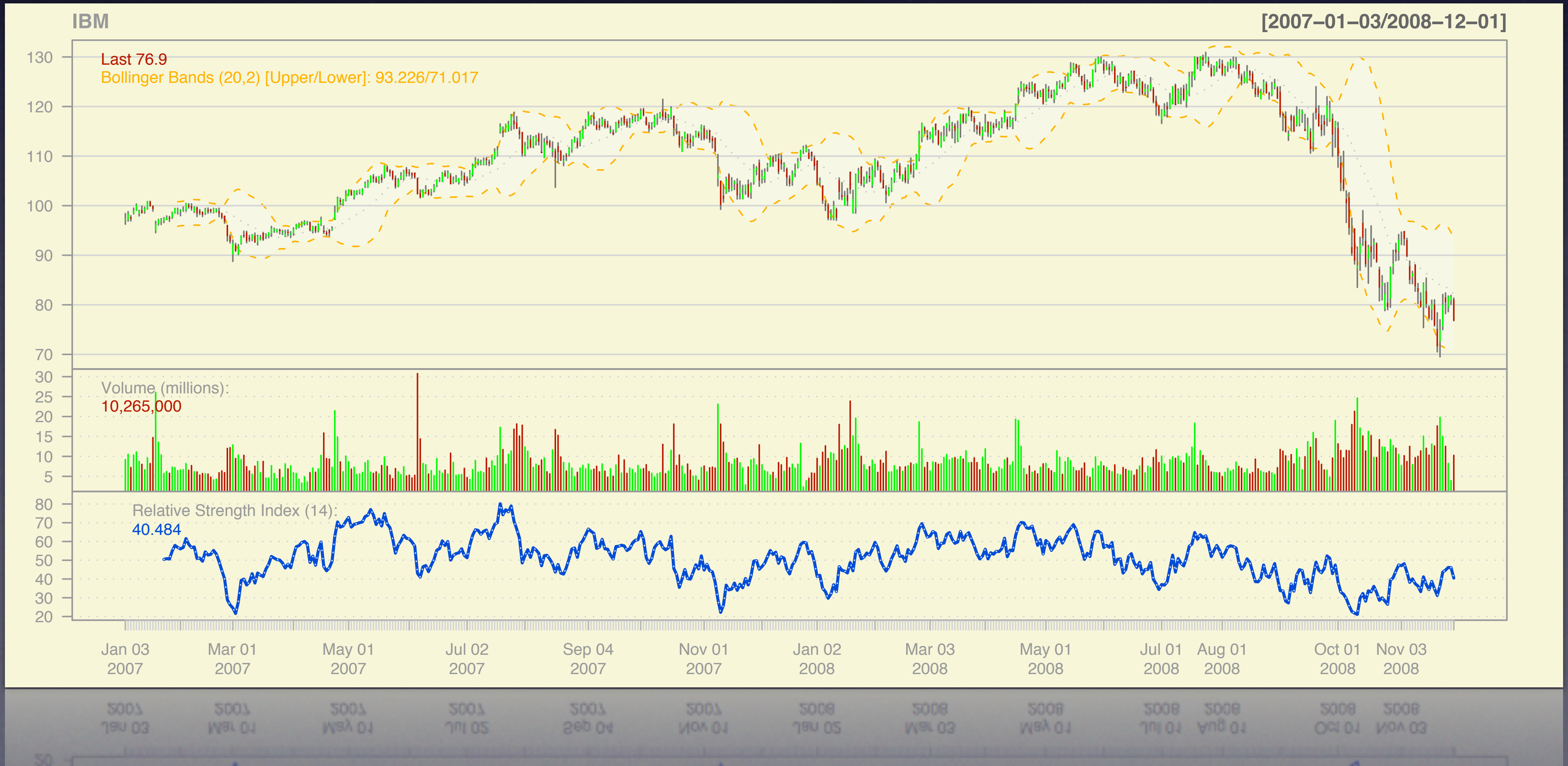
Customizing: setTA, theme and layout

```
> getSymbols("IBM")
> chartSeries(IBM, theme= "beige")
```

(Visualization)

Customizing: setTA, theme and layout

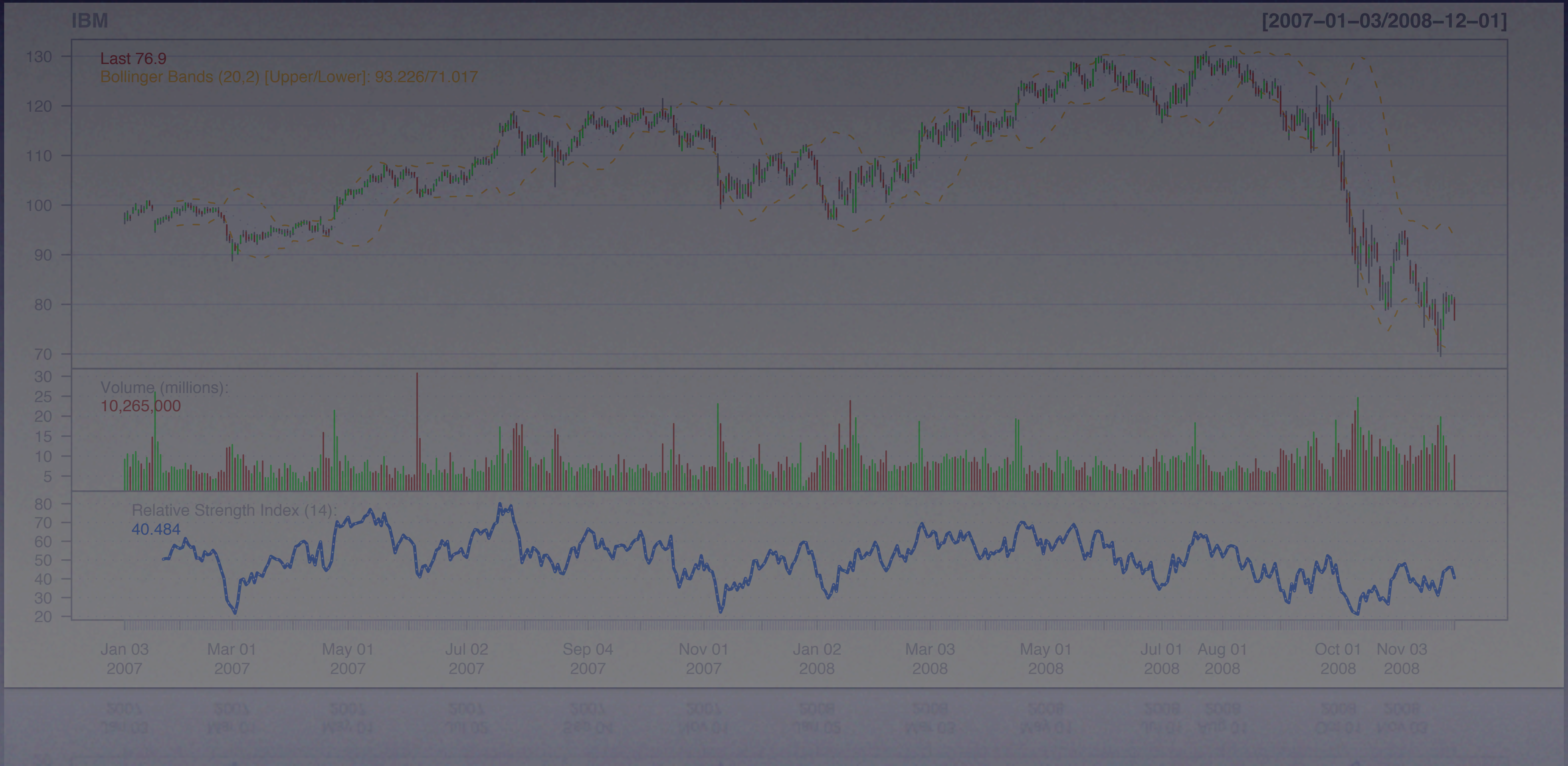
```
> getSymbols("IBM")
> chartSeries(IBM, theme= "beige")
```



(Visualization)

Customizing: setTA, theme and layout

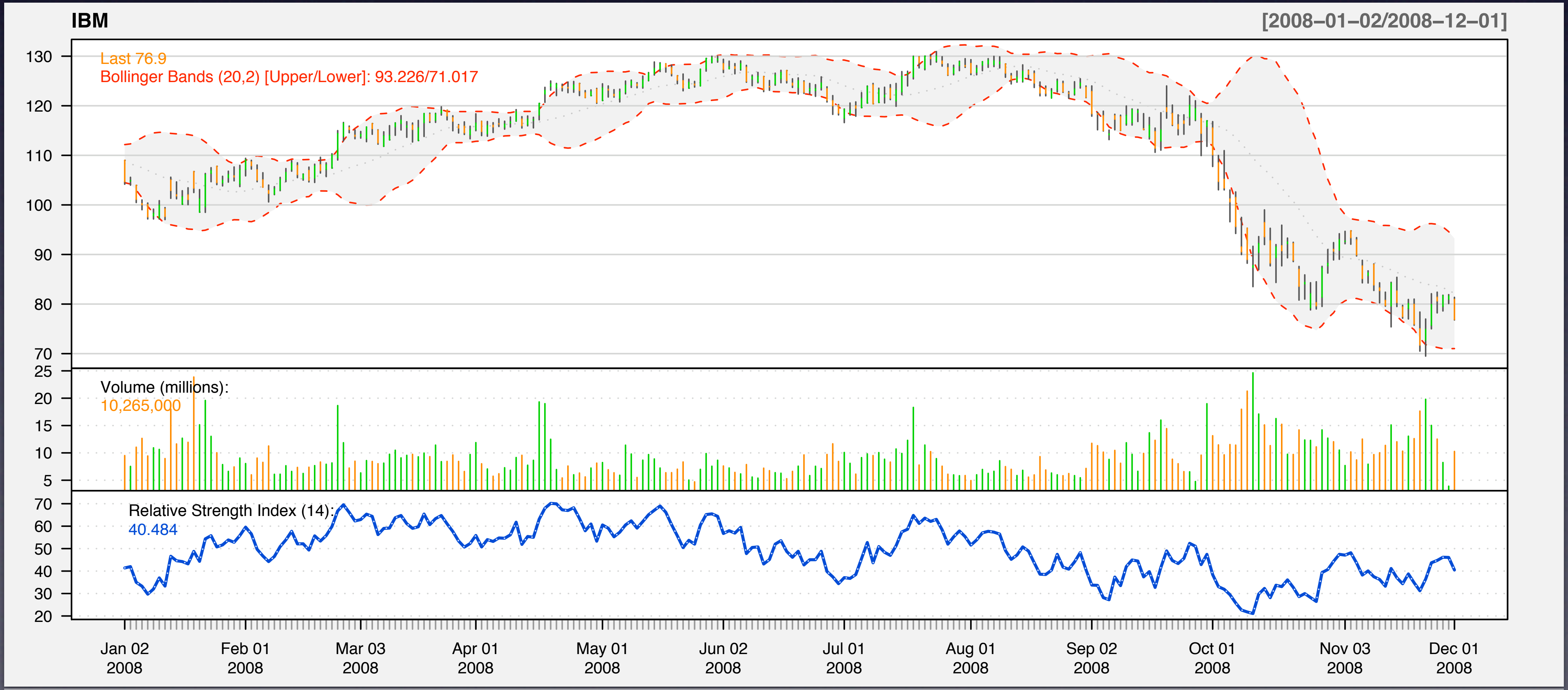
```
> getSymbols("IBM")
> chartSeries(IBM, theme= "beige")
> reChart(theme= "white", subset= "2008")
```



(Visualization)

Customizing: setTA, theme and layout

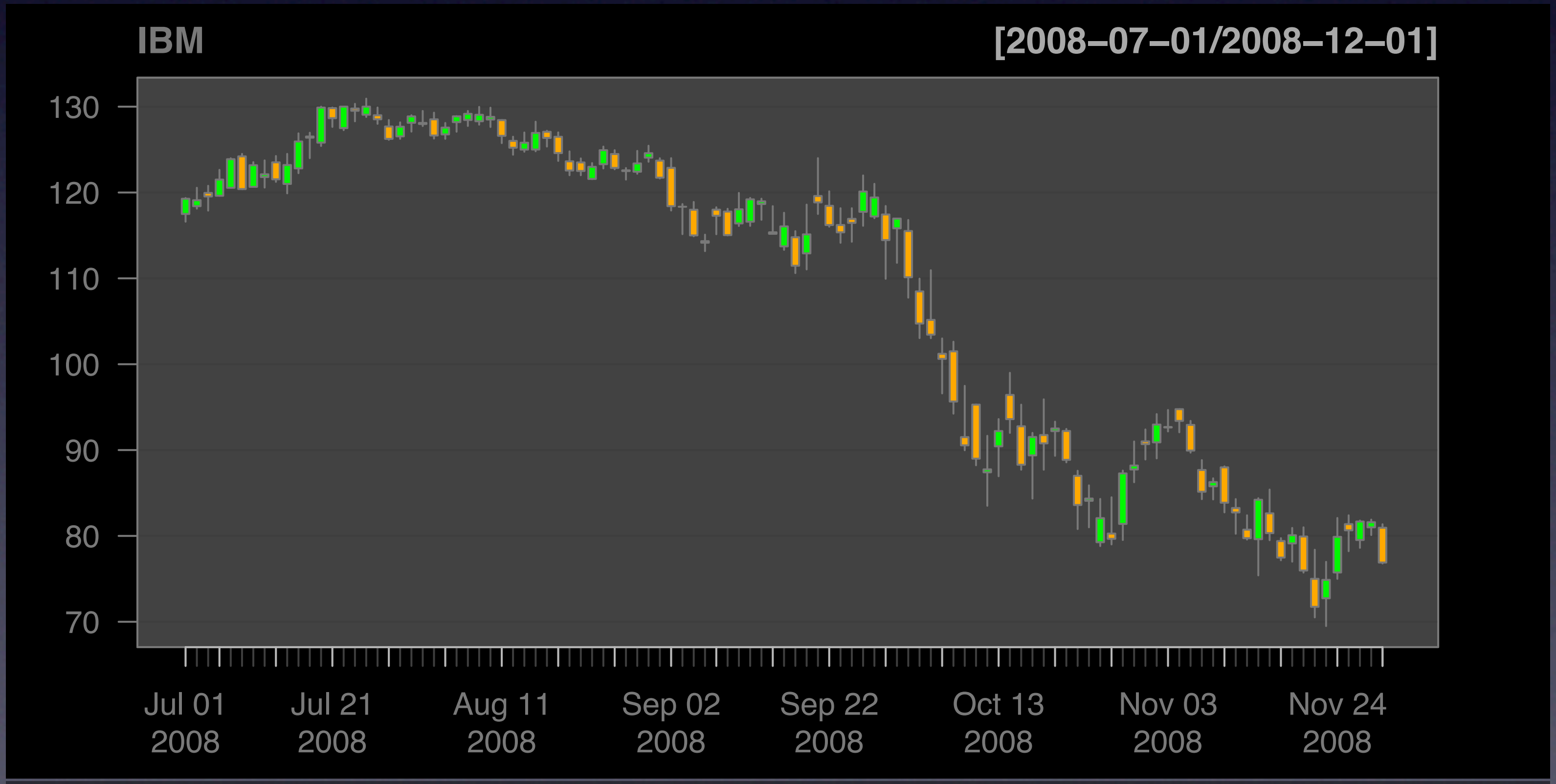
```
> getSymbols("IBM")
> chartSeries(IBM, theme= "beige")
> reChart(theme= "white", subset= "2008")
```



(Visualization)

Custom layouts

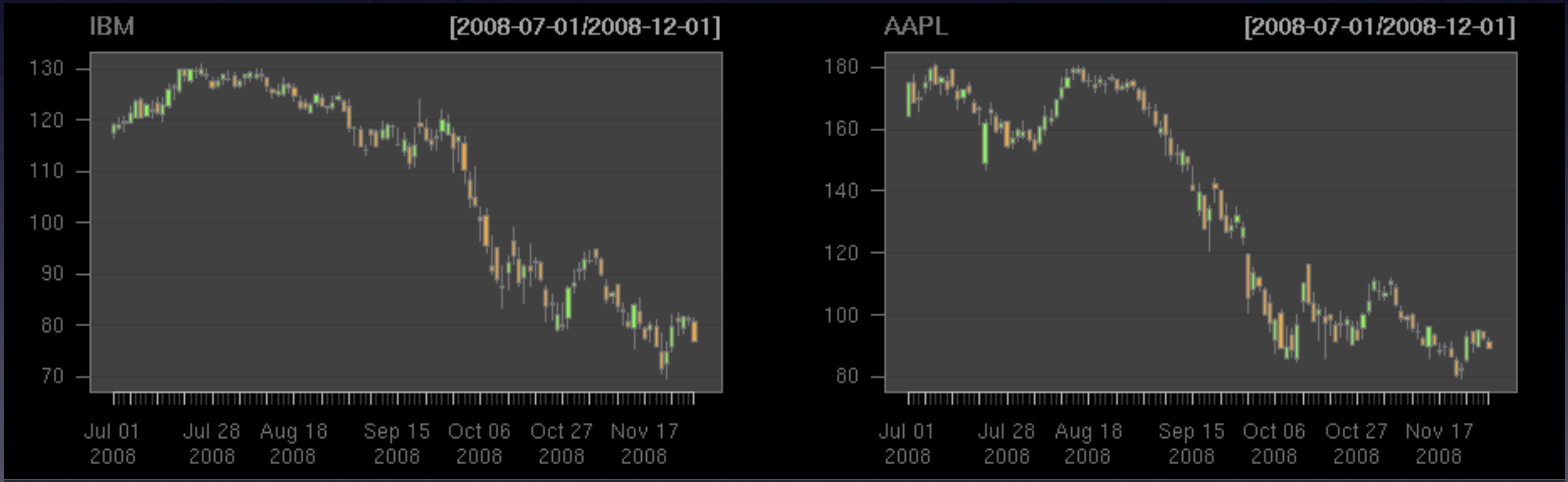
One main series



(Visualization)

Custom layouts

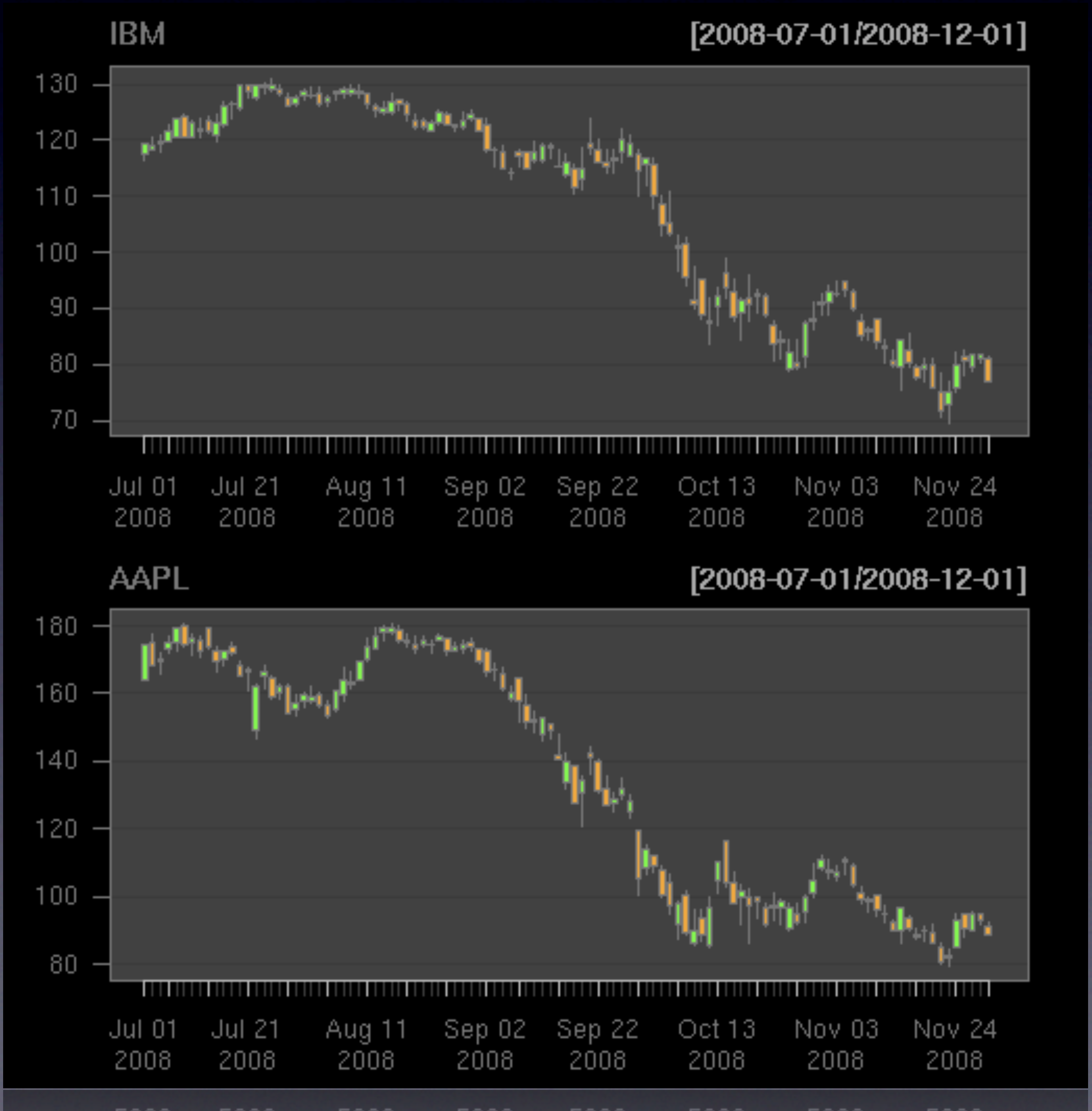
Two Series



(Visualization)

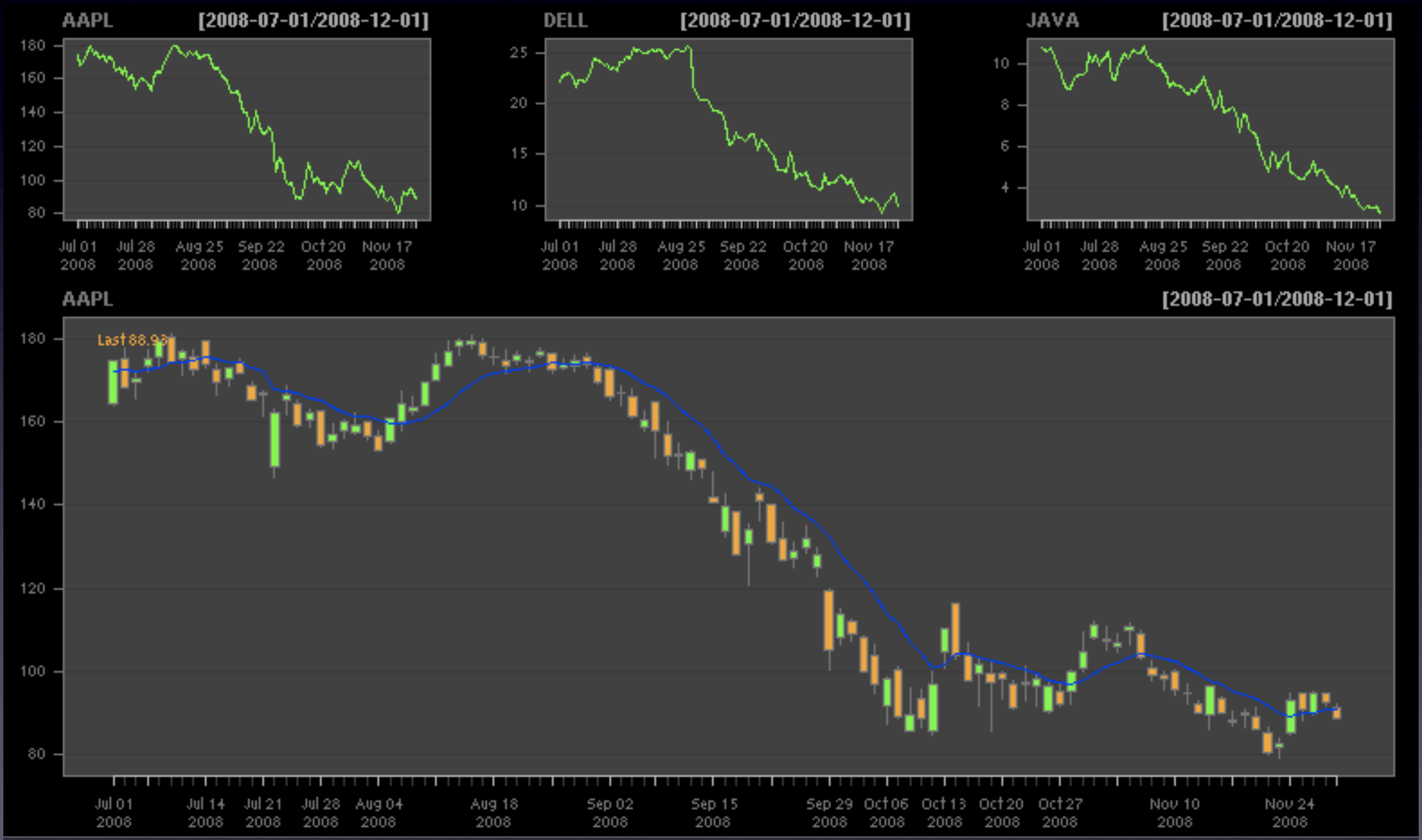
Custom layouts

Two Series
(up and down)



(Visualization)

Custom layouts



(Visualization)

Custom indicators

addTA

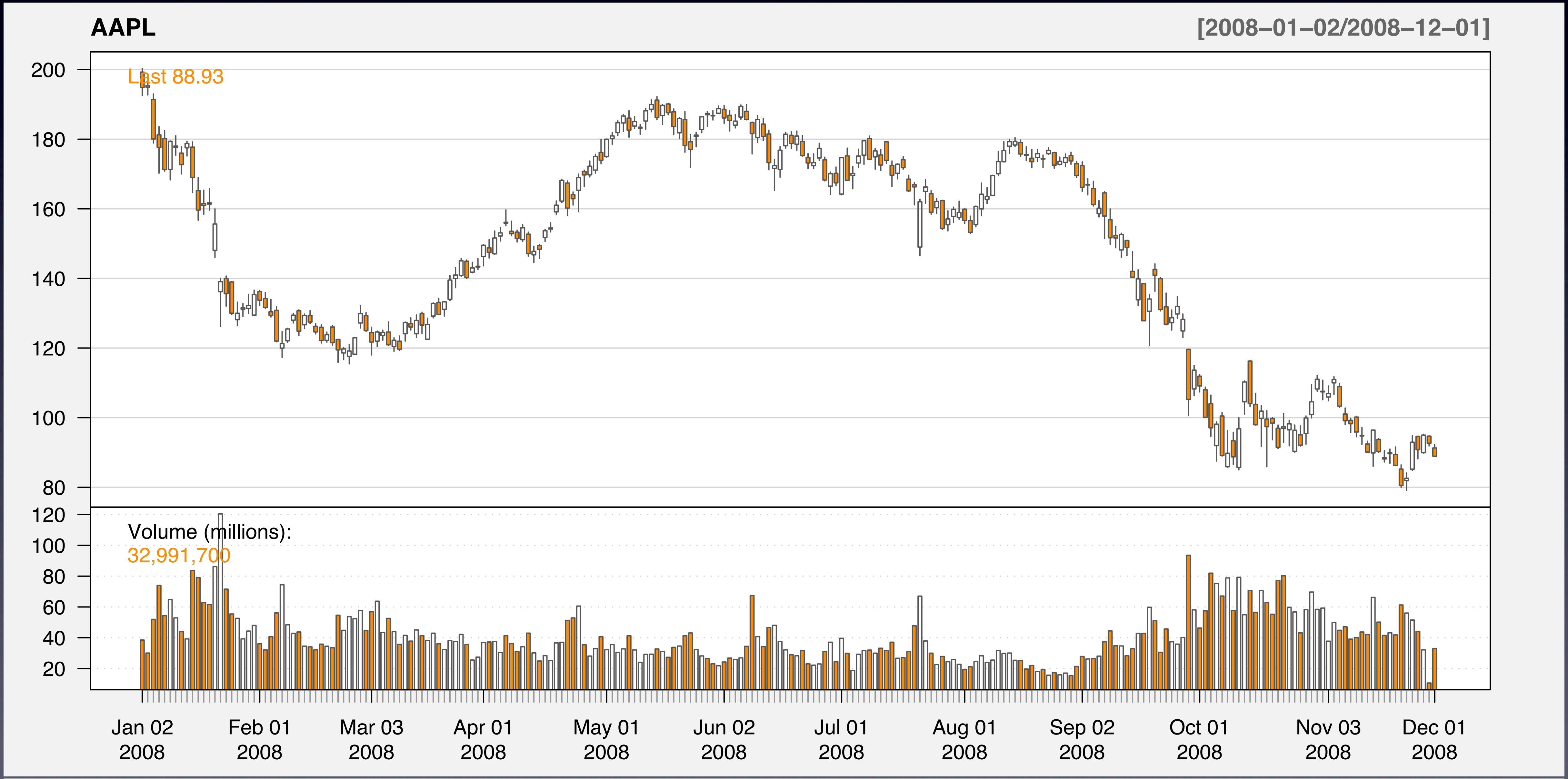
add xtsible or raw data directly to a chart

newTA

create a new TA function like the built-in ones

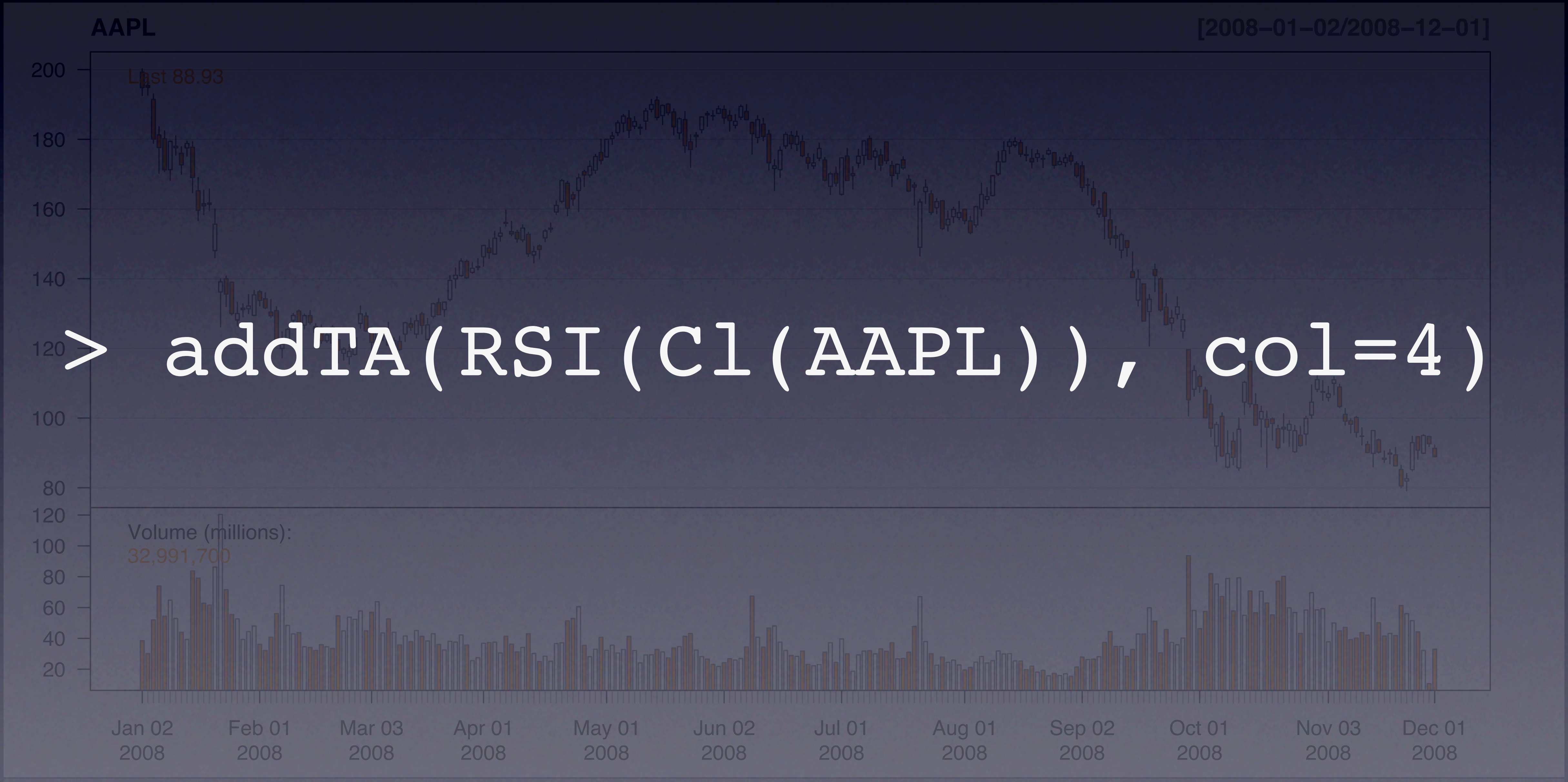
(Visualization)

addTA



(Visualization)

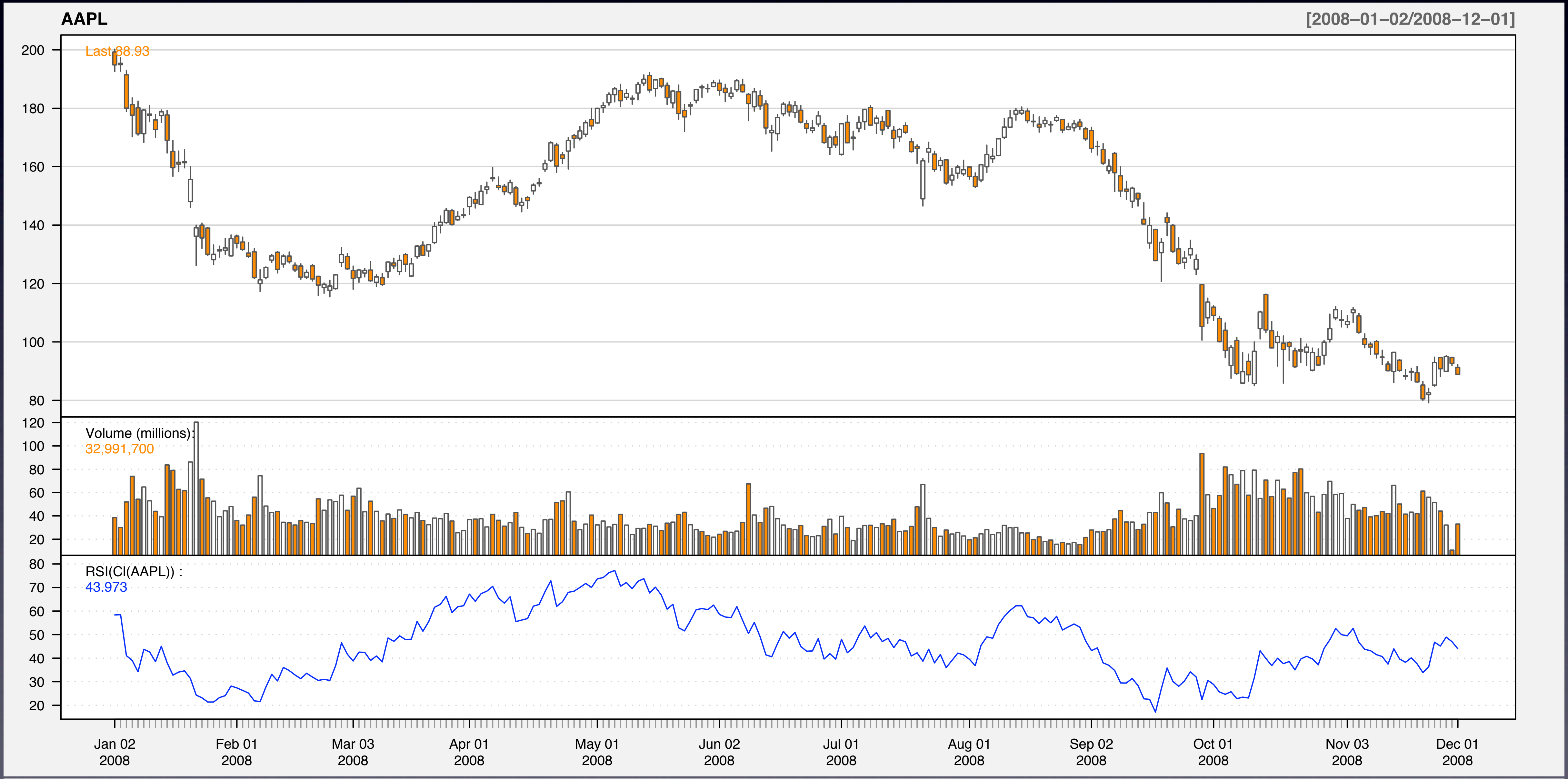
addTA



```
> addTA(RSI(Cl(AAPL)), col=4)
```

(Visualization)

with our own RSI



(Visualization)

RSI above 70 rule as shaded region?

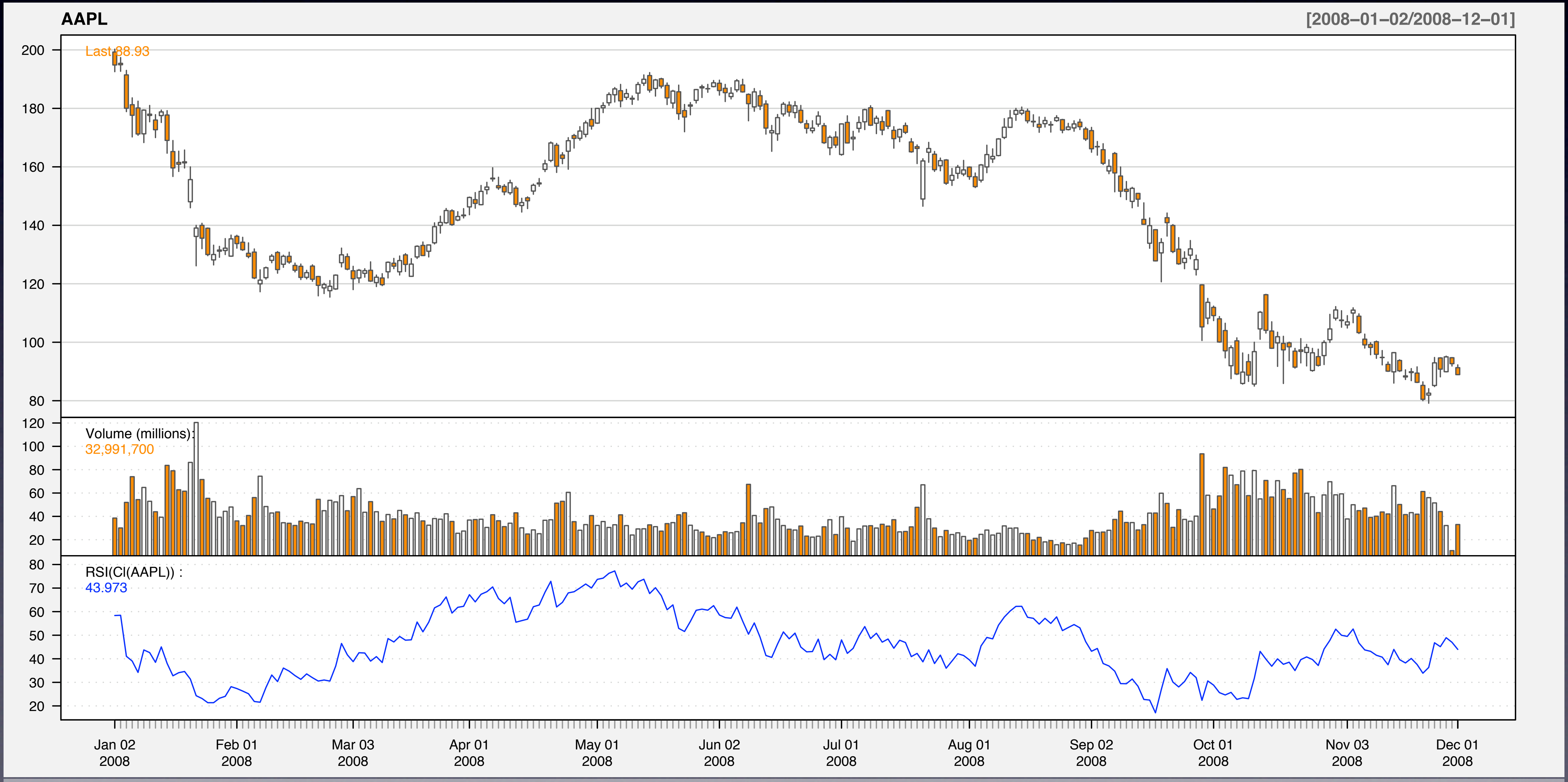
(Visualization)

RSI above 70 rule as shaded region?

Pass a logical vector to addTA

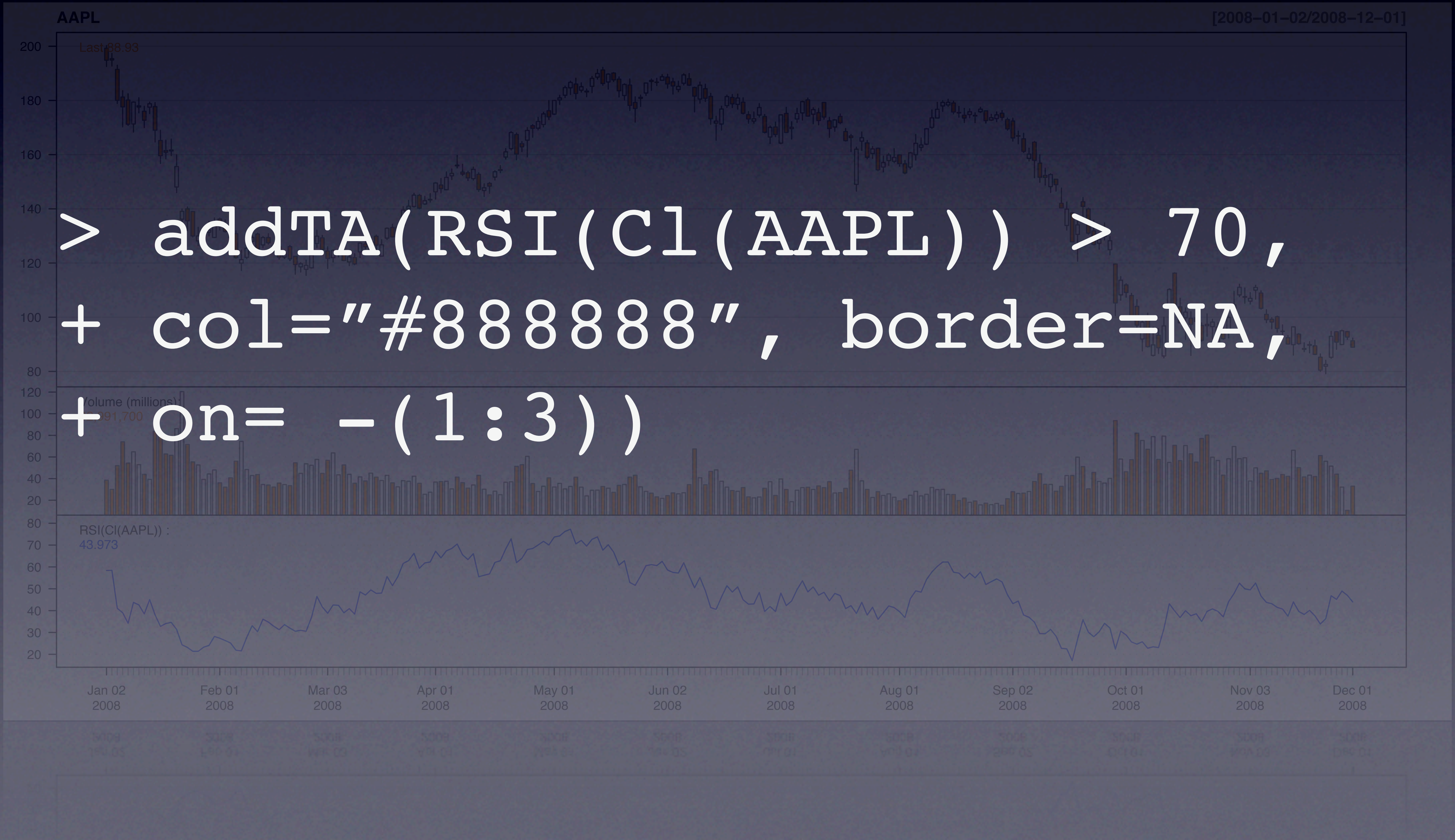
(Visualization)

same chart...



(Visualization)

same chart...



```
> addTA(RSI(Cl(AAPL)) > 70,
+ col="#888888", border=NA,
+ on= -(1:3))
```

(Visualization)

same chart...

AAPL
[2008-01-02/2008-12-01]
Last 88.93
Volume (millions)
RSI(Cl(AAPL)) :
43.973
Jan 02 2008
Feb 01 2008
Mar 03 2008
Apr 01 2008
May 01 2008
Jun 02 2008
Jul 01 2008
Aug 01 2008
Sep 02 2008
Oct 01 2008
Nov 03 2008
Dec 01 2008

```
> addTA(RSI(Cl(AAPL)) > 70,
+ col="#888888", border=NA,
+ on= -(1:3))
```

(Visualization)

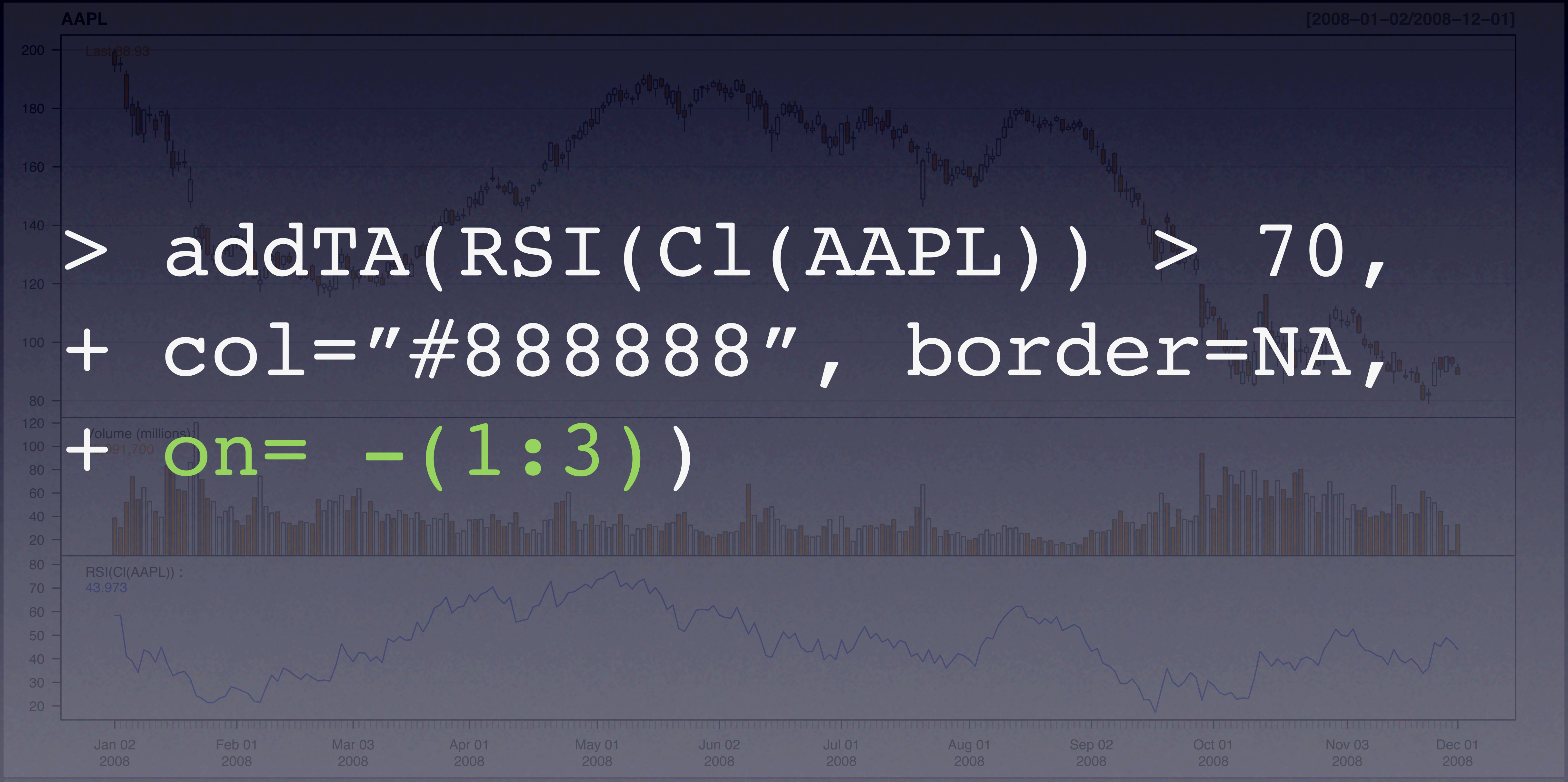
same chart...

AAPL
[2008-01-02/2008-12-01]
Last 88.93
Volume (millions):
RSI(Cl(AAPL)) :
43.973
Jan 02 2008
Feb 01 2008
Mar 03 2008
Apr 01 2008
May 01 2008
Jun 02 2008
Jul 01 2008
Aug 01 2008
Sep 02 2008
Oct 01 2008
Nov 03 2008
Dec 01 2008

```
> addTA(RSI(Cl(AAPL)) > 70,
+ col="#888888", border=NA,
+ on= -(1:3))
```

(Visualization)

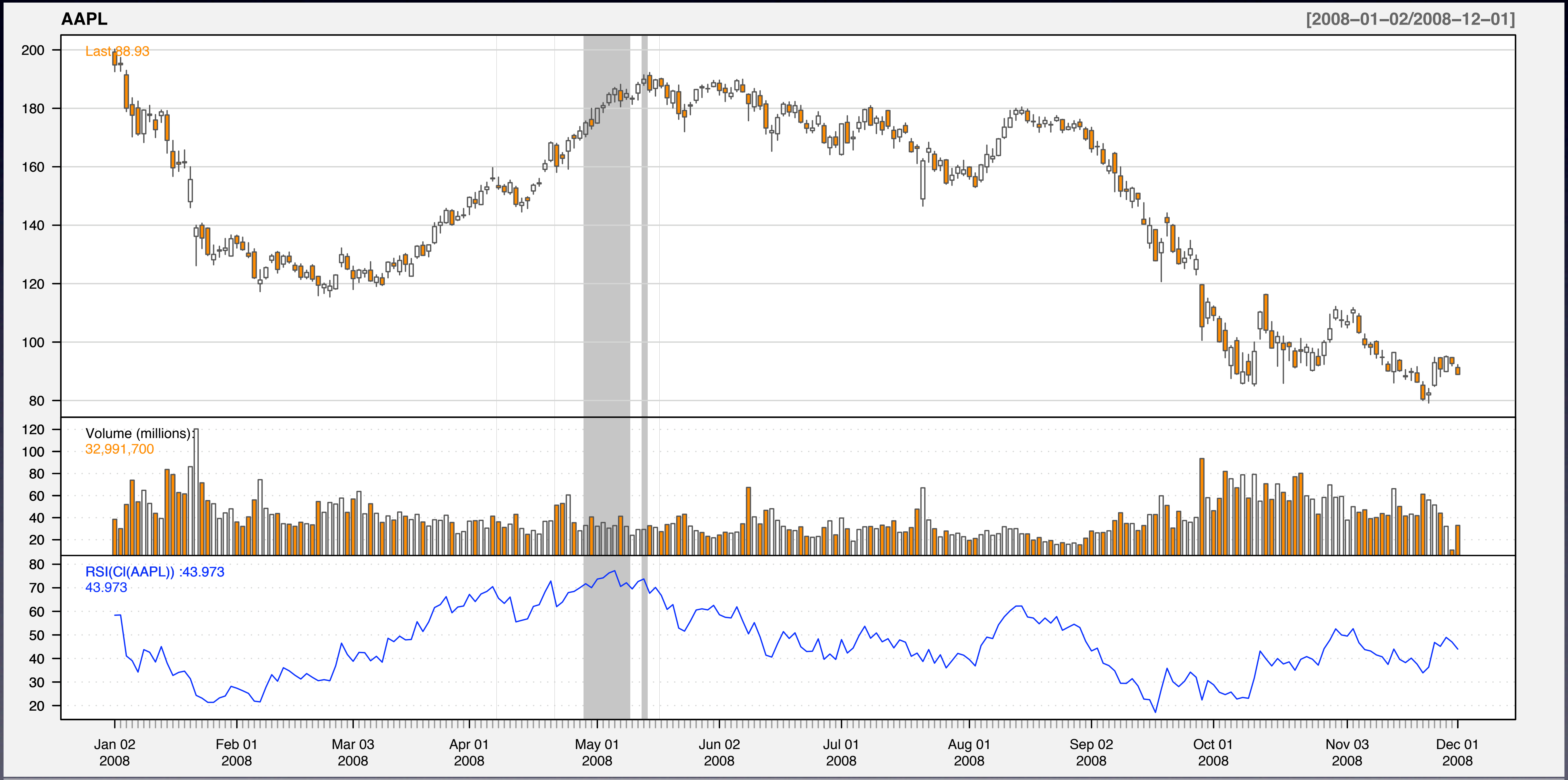
same chart...



```
> addTA(RSI(Cl(AAPL)) > 70,
+ col="#888888", border=NA,
+ on= -(1:3))
```

(Visualization)

Bands are automatically created!



(Visualization)

Custom indicators: newTA

GMMA

Guppy Multiple Moving Average

(Visualization)

Custom indicators: newTA

```
> # 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)
+ }
>
```

(Visualization)

Custom indicators: newTA

```
> # 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=Cl,
+                                 col=c(rep(3,6),
+                                         rep("#333333",6)),
+                                 legend="GMMA")
> class(addGuppy)
[1]  "function"
```

(Visualization)

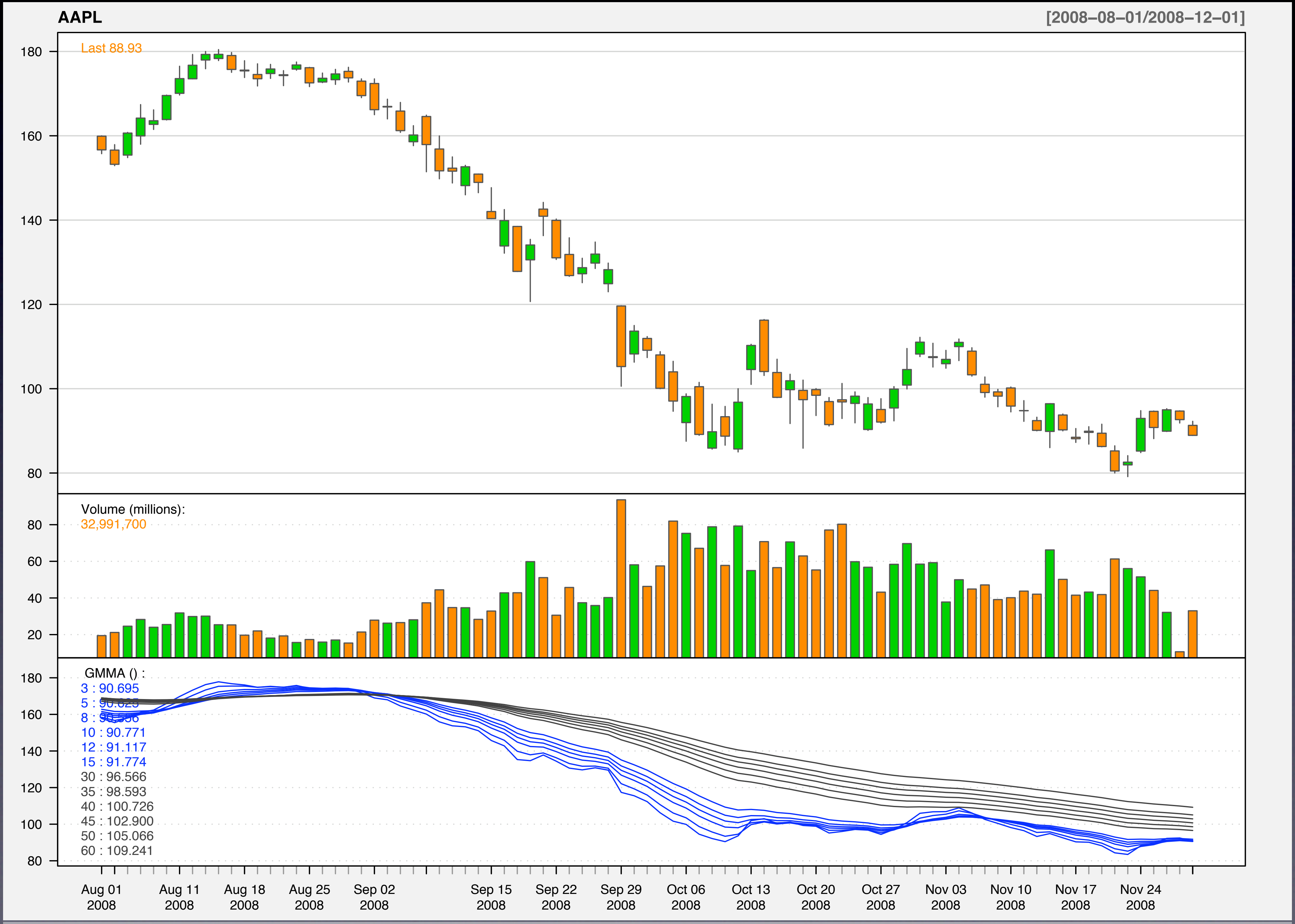
Custom indicators: newTA

```
> # 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=Cl,
+                                    col=c(rep(3,6),
+                                              rep("#333333",6)),
+                                    legend="GMMA")
> class(addGuppy)
[1]  "function"
```

(Visualization)



(Visualization)

The Future

book/depth data 2D/3D/4D

option surfaces/payoffs

real-time updating...

xts: extensible time series

Jeffrey A. Ryan & Joshua M. Ulrich

New Release 0.6-2 !

xts: extensible time series

Q: What is xts?

(and *why* another time-series?)

xts: extensible time series

Q: What is xts?

(and *why* another time-series?)

A: xts is a matrix plus a time index.

(formally extending zoo)

xts: extensible time series

Q: Why another time-series?

We needed a tool that was time-aware, not just ordered...

xts: extensible time series

Q: Why another time-series?

We needed a tool that was time-aware, not just ordered...

and had the ability to handle all time-series classes equally --- a developer's time-series.

xts: extensible time series

structure

xts: extensible time series

structure

- S3 class extending zoo and matrix
- index attribute holds time-based index
- arbitrary attributes can be attached

xts: extensible time series

structure

- S3 class extending zoo and matrix
- index attribute holds time-based index
- arbitrary attributes can be attached
- index must be time-based
- no rownames allowed
- special formatting tools (time and attr)

xts: extensible time series

unique features

xts: extensible time series

unique features

- ISO 8601 style subsetting by time
  x['200701'] returns January of 2007
  x['2000/200803'] start of '00 to Mar '08

xts: extensible time series

unique features

- ISO 8601 style subsetting by time
  x['200701'] returns January of 2007
  x['2000/200803'] start of '00 to Mar '08
- Advanced *lossless* conversion utilities
  try.xts: coerce data to xts, if possible
  reclass: reconvert automatically
  speed development, add flexibility!

xts: extensible time series

unique features

- ISO 8601 style subsetting by time
  x['200701'] returns January of 2007
  x['2000/200803'] start of '00 to Mar '08
- Advanced *lossless* conversion utilities
  try.xts: coerce data to xts, if possible
  reclass: reconvert automatically
  speed development, add flexibility!
- Time-based utilities:
  periodicity, to.period, endpoints
  period.apply, axTicksByTime, plotting, ...

xts: extensible time series

New features for 0.6-2

xts: extensible time series

Internal Structure Changes

- index is now POSIXct representation (int or double)
- index class is *preserved* and used for index(), as well as for printing, and conversion with as.
- xts.compat.zoo.lag global option

xts: extensible time series

Internal Algorithm Changes

- [.xts subsetting is carried out with a binary search

xts: extensible time series

Internal Algorithm Changes

- [.xts subsetting is carried out with a binary search
- Optimized on 10’s of millions of observations

xts: extensible time series

Internal Algorithm Changes

- [.xts subsetting is carried out with a binary search
- Optimized on 10's of millions of observations
- 3200+ lines of C code specific to xts

  merge, rbind, cbind, lag, Ops, diff, ...

xts: extensible time series

Performance Benchmarks*

	matrix	vector	ts	timeSeries	fts	zoo/xts	xts (0.6-2)
construct	0.052	0.537	0.022	65.00*	0.128	1.032	0.055
subset by time	0.130	0.132	0.003	103.40*	0.247	0.453	0.007
merge/cbind*	0.031*	0.031	0.257	170.00*	1.146*	16.77	0.052
rbind	0.05	0.035	0.024	0.30**	1.853	9.527	0.048
diff	0.164	0.205	1.049	56.35*	0.133	11.49	0.053
lag	0.047	0.052	0.016	57.55*	0.024	1.226	0.024
x + x	<u>0.018</u>	0.028	1.068	0.270*	1.403	8.920	0.018
x + x[-1]	error	error	error	error	1.403	9.200	0.089

* memory limits limited timeSeries objects to 100,000 obs, so these are extrapolated timings

* results in an *unordered* time series *cbind for fts

* timing on a very modest 2.2 GHz MacBook with 2GB RAM calling: .xts(1:1e6L, 1:1e6L)

xts: extensible time series

New C level API

- Access xts functionality from C code linked to R
- Worked package example installed in api_example/
- #include "xts.h" & linkingTo in DESCRIPTION

xts: extensible time series

Future Direction

- Column attributes

xts: extensible time series

Future Direction

- Column attributes
- In-memory database functionality -- keys, joins

xts: extensible time series

Future Direction

- Column attributes
- In-memory database functionality -- keys, joins
- Persistent storage mechanisms

xts: extensible time series

Future Direction

- Column attributes
- In-memory database functionality -- keys, joins
- Persistent storage mechanisms
- Mixed factor/numeric support in xts objects

xts: extensible time series

Future Direction

- Column attributes
- In-memory database functionality -- keys, joins
- Persistent storage mechanisms
- Mixed factor/numeric support in xts objects
- (xts)data.frame style object, i.e. xts lists

xts: extensible time series

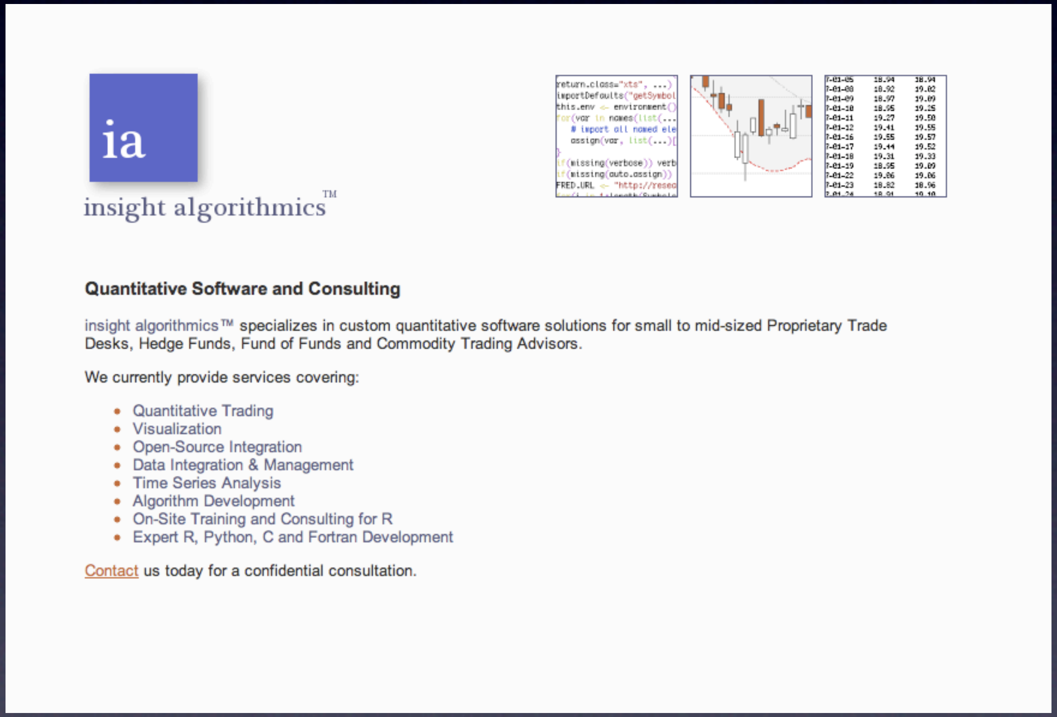
Future Direction

- Column attributes
- In-memory database functionality -- keys, joins
- Persistent storage mechanisms
- Mixed factor/numeric support in xts objects
- (xts)data.frame style object, i.e. xts lists
- support for *data.table*, *bigmemory* or *ff* in place of matrix objects

More Information

www.insightalgo.com

www.quantmod.com



ia

insight algorithmics™

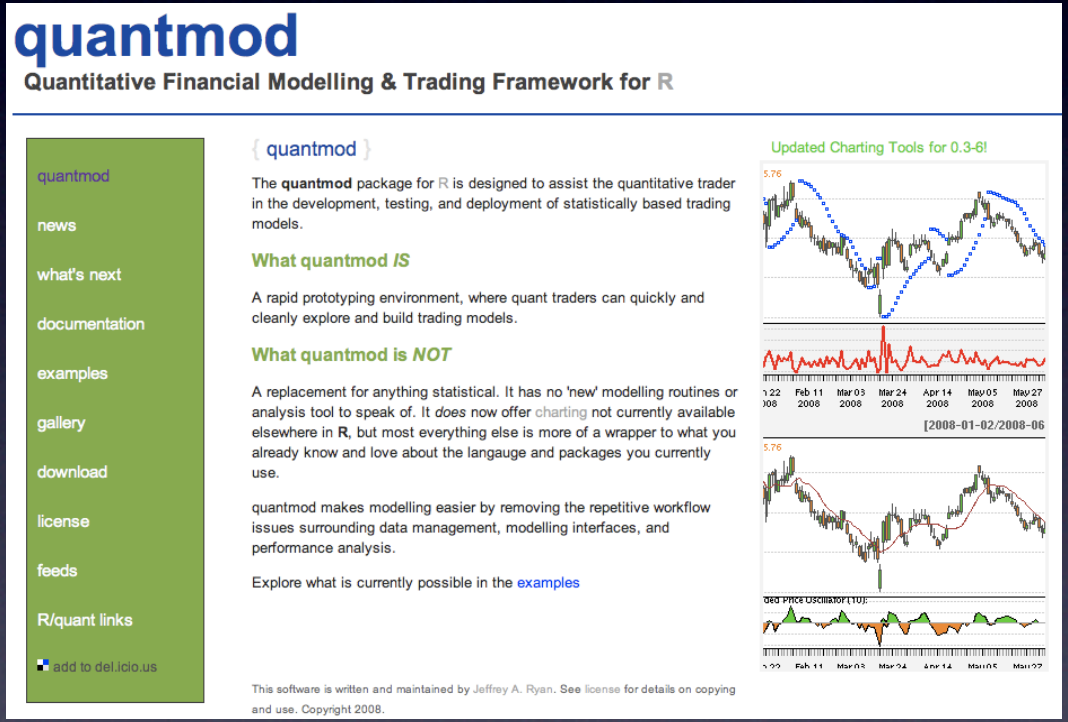
Quantitative Software and Consulting

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.



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 langauge 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

www.quantmod.com/Columbia2008