Welcome to the course!

IMPORTING AND MANAGING FINANCIAL DATA IN R

Joshua Ulrich
Instructor

About me

- Author and/or maintainer of several packages
  - `TTR` , `xts` , `quantmod` , `blotter` , `quantstrat`
- R/Finance Conference Organizing Committee
- St. Louis R User Group

What is getSymbols()?

- Provides access to multiple data sources
- Returns `xts` object by default
- Can import data two ways:
  - Return data like an ordinary function
  - Create an object like `load()` does in base R

```r
getSymbols(Symbols = "AAPL", src = "yahoo")
```

```
"AAPL"
```

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

```
"AAPL"
```

```r
head(AAPL)
```

```
           AAPL.Open AAPL.High AAPL.Low AAPL.Close AAPL.Volume AAPL.Adjusted
2007-01-03     86.29     86.58    81.90      83.80   309579900      10.85709
2007-01-04     84.05     85.95    83.82      85.66   211815100      11.09807
2007-01-05     85.77     86.20    84.40      85.05   208685400      11.01904
2007-01-08     85.96     86.53    85.28      85.47   199276700      11.07345
2007-01-09     86.45     92.98    85.15      92.57   837324600      11.99333
2007-01-10     94.75     97.80    93.45      97.00   738220000      12.56728
```

getSymbols() data sources

Yahoo! Finance	YAHOO! FINANCE
Google Finance	Google Finance
FRED	FRED ECONOMIC DATA | ST. LOUIS FED
Oanda	OANDA
CSV	.CSV

Other getSymbols() data sources

- Yahoo! Finance Japan
- MySQL
- SQLite
- RData
- rds (created by `saveRDS()`)

getSymbols() example

```
# Load data like load()
getSymbols("AAPL", auto.assign = TRUE)
```

```
"AAPL"
```

```
head(AAPL, n = 3)
```

```
           AAPL.Open AAPL.High AAPL.Low AAPL.Close AAPL.Volume AAPL.Adjusted
2007-01-03     86.29     86.58    81.90      83.80   309579900      10.85709
2007-01-04     84.05     85.95    83.82      85.66   211815100      11.09807
2007-01-05     85.77     86.20    84.40      85.05   208685400      11.01904
```

getSymbols() example

```
# Return data like a normal function
aapl <- getSymbols("AAPL", auto.assign = FALSE)
head(aapl, n = 3)
```

```
           AAPL.Open AAPL.High AAPL.Low AAPL.Close AAPL.Volume AAPL.Adjusted
2007-01-03     86.29     86.58    81.90      83.80   309579900      10.85709
2007-01-04     84.05     85.95    83.82      85.66   211815100      11.09807
2007-01-05     85.77     86.20    84.40      85.05   208685400      11.01904
```

Let's practice!

Introduction to Quandl

IMPORTING AND MANAGING FINANCIAL DATA IN R

Joshua Ulrich
Instructor

What is Quandl?

- Data service:
  - https://www.quandl.com/
- R package:
  - https://CRAN.R-project.org/package=Quandl
- Function:
  - `Quandl::Quandl()`

Quandl() versus getSymbols()

- Both provide access to multiple data sources
- `Quandl()` always returns data (i.e. does not behave like `load()`)

```r
# Instrument and source specified by Symbols and src arguments
quantmod::getSymbols(Symbols = "DGS10", src = "FRED")
```

```r
# Instrument and source specified by code argument
dgs10 <- Quandl::Quandl(code = "FRED/DGS10")
```

Quandl() versus getSymbols()

- `type` argument controls class of return object:
  - "raw" (`data.frame` object), `xts` , `zoo` , `ts` , `timeSeries`

```r
# Return xts object instead of data.frame
dgs10 <- Quandl::Quandl(code = "FRED/DGS10", type = "xts")
```

- Defaults
  - `getSymbols()` returns `xts`
  - `Quandl()` returns `data.frame`

Let's practice!

IMPORTING AND MANAGING FINANCIAL DATA IN R

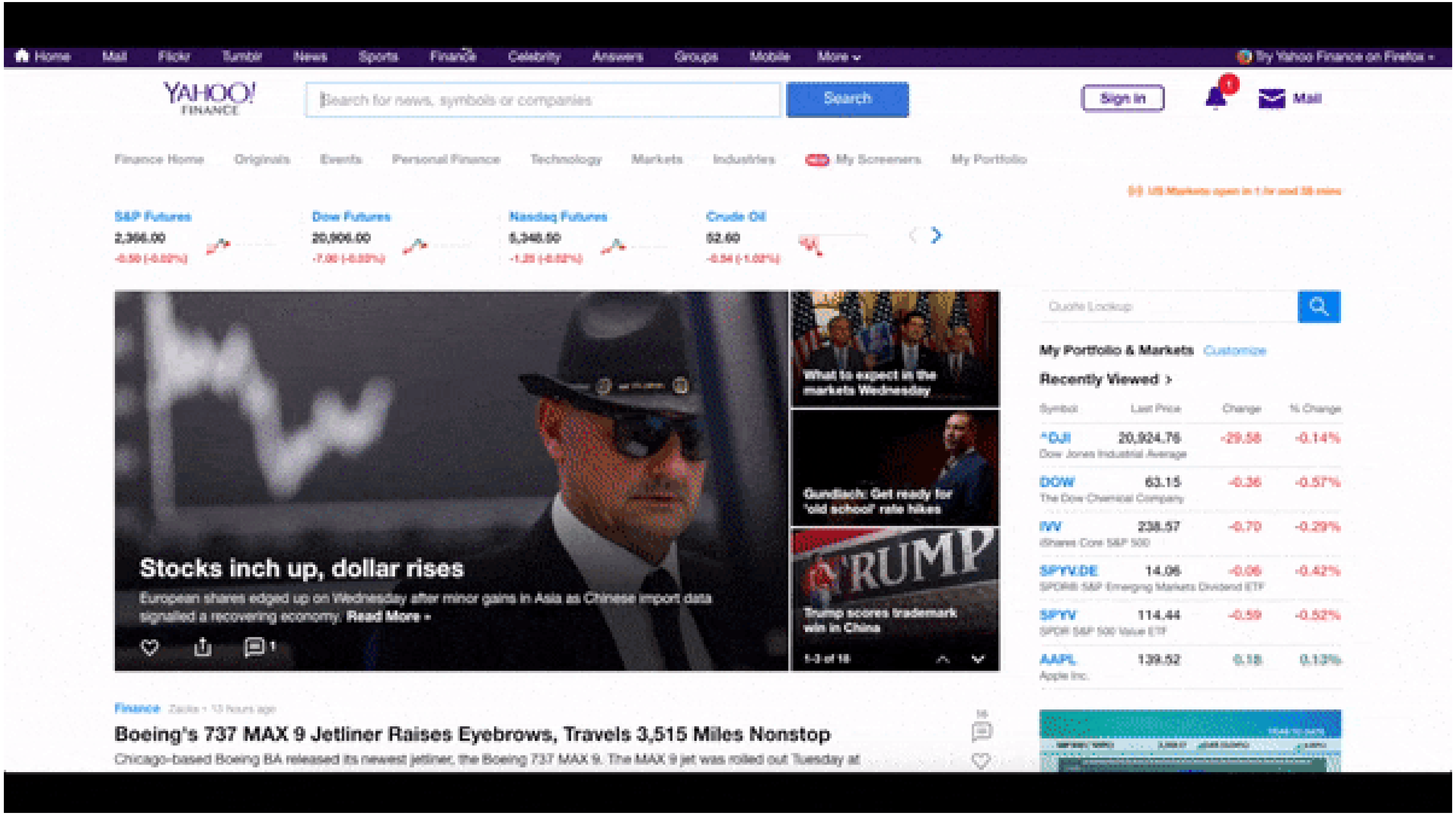
Finding data from internet sources

IMPORTING AND MANAGING FINANCIAL DATA IN R

Joshua Ulrich

Instructor

Yahoo Finance



US Markets open in 1 hr and 28 mins

S&P Futures	Dow Futures	Nasdaq Futures	Crude Oil
2,366.00	20,906.00	5,340.50	52.60
-0.50 (-0.02%)	-7.00 (-0.03%)	-1.25 (-0.02%)	-0.54 (-1.02%)

Stocks inch up, dollar rises

European shares edged up on Wednesday after minor gains in Asia as Chinese import data signalled a recovering economy. **Read More »**

What to expect in the markets Wednesday

Gundlach: Get ready for 'old school' rate hikes

Trump scores trademark win in China

My Portfolio & Markets

Recently Viewed ›

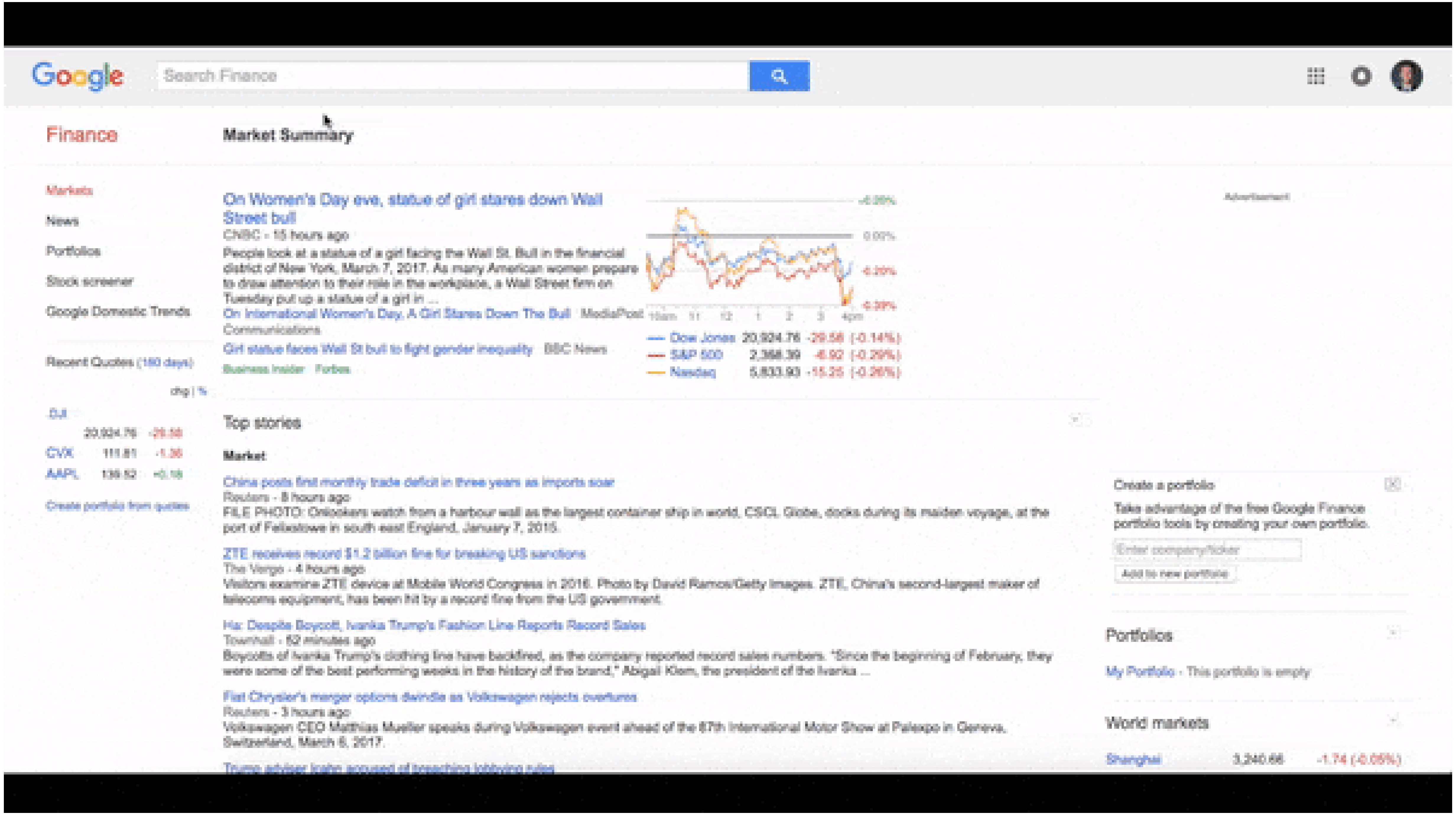
Symbol	Last Price	Change	% Change
^DJI Dow Jones Industrial Average	20,924.76	-29.58	-0.14%
DOW The Dow Chemical Company	63.15	-0.36	-0.57%
IVV iShares Core S&P 500	238.57	-0.70	-0.29%
SPYV.DE SPDR® S&P Emerging Markets Dividend ETF	14.06	-0.06	-0.42%
SPYV SPDR S&P 500 Value ETF	114.44	-0.59	-0.52%
AAPL Apple Inc.	139.52	0.18	0.13%

Finance Zacks • 13 hours ago

Boeing's 737 MAX 9 Jetliner Raises Eyebrows, Travels 3,515 Miles Nonstop

Chicago-based Boeing BA released its newest jetliner, the Boeing 737 MAX 9. The MAX 9 jet was rolled out Tuesday at

Google Finance



Market Summary

On Women's Day eve, statue of girl stares down Wall Street bull
CNBC - 15 hours ago
People look at a statue of a girl facing the Wall St. Bull in the financial district of New York, March 7, 2017. As many American women prepare to draw attention to their role in the workplace, a Wall Street firm on Tuesday put up a statue of a girl in ...
On International Women's Day, A Girl Stares Down The Bull MediaPost Communications
Girl statue faces Wall St bull to fight gender inequality BBC News
Business Insider Forbes

— Dow Jones	20,924.76	-29.58	(-0.14%)
— S&P 500	2,368.39	-6.92	(-0.29%)
— Nasdaq	5,833.93	-15.25	(-0.26%)

Recent Quotes (180 days)

		chg | %
.DJI		
	20,924.76	-29.58
CVX	111.81	-1.36
AAPL	139.52	+0.18

Create portfolio from quotes

Top stories

Market

China posts first monthly trade deficit in three years as imports soar
Reuters - 8 hours ago
FILE PHOTO: Onlookers watch from a harbour wall as the largest container ship in world, CSCL Globe, docks during its maiden voyage, at the port of Felixstowe in south east England, January 7, 2015.

ZTE receives record $1.2 billion fine for breaking US sanctions
The Verge - 4 hours ago
Visitors examine ZTE device at Mobile World Congress in 2016. Photo by David Ramos/Getty Images. ZTE, China's second-largest maker of telecoms equipment, has been hit by a record fine from the US government.

Ha: Despite Boycott, Ivanka Trump's Fashion Line Reports Record Sales
Townhall - 52 minutes ago
Boycotts of Ivanka Trump's clothing line have backfired, as the company reported record sales numbers. "Since the beginning of February, they were some of the best performing weeks in the history of the brand," Abigail Klem, the president of the Ivanka ...

Fiat Chrysler's merger options dwindle as Volkswagen rejects overtures
Reuters - 3 hours ago
Volkswagen CEO Matthias Mueller speaks during Volkswagen event ahead of the 87th International Motor Show at Palexpo in Geneva, Switzerland, March 6, 2017.

Trump adviser Icahn accused of breaching lobbying rules

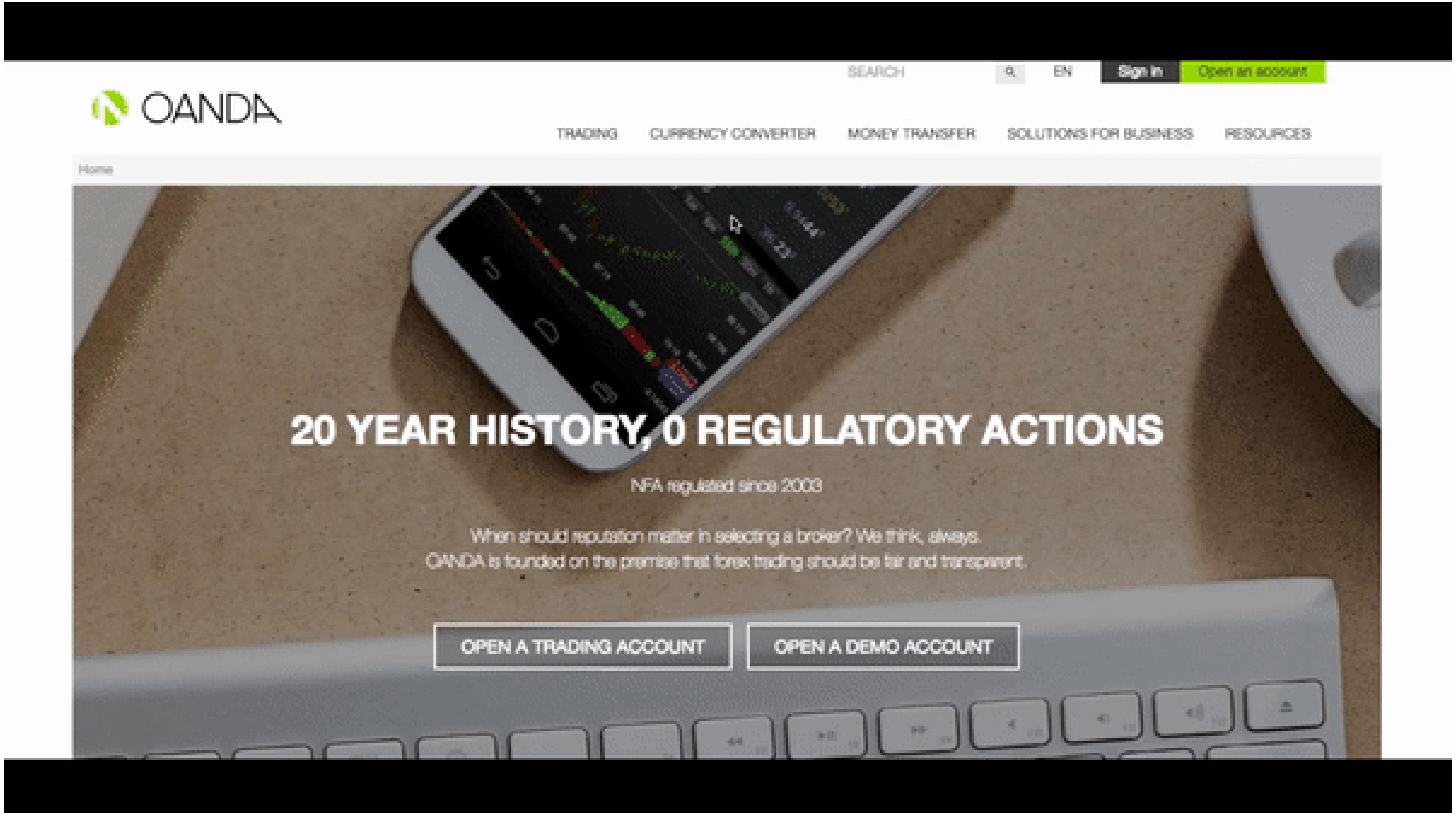
Create a portfolio
Take advantage of the free Google Finance portfolio tools by creating your own portfolio.
Enter company/ticker
Add to new portfolio

Portfolios
My Portfolio - This portfolio is empty

World markets

Shanghai	3,240.66	-1.74 (-0.05%)

Oanda

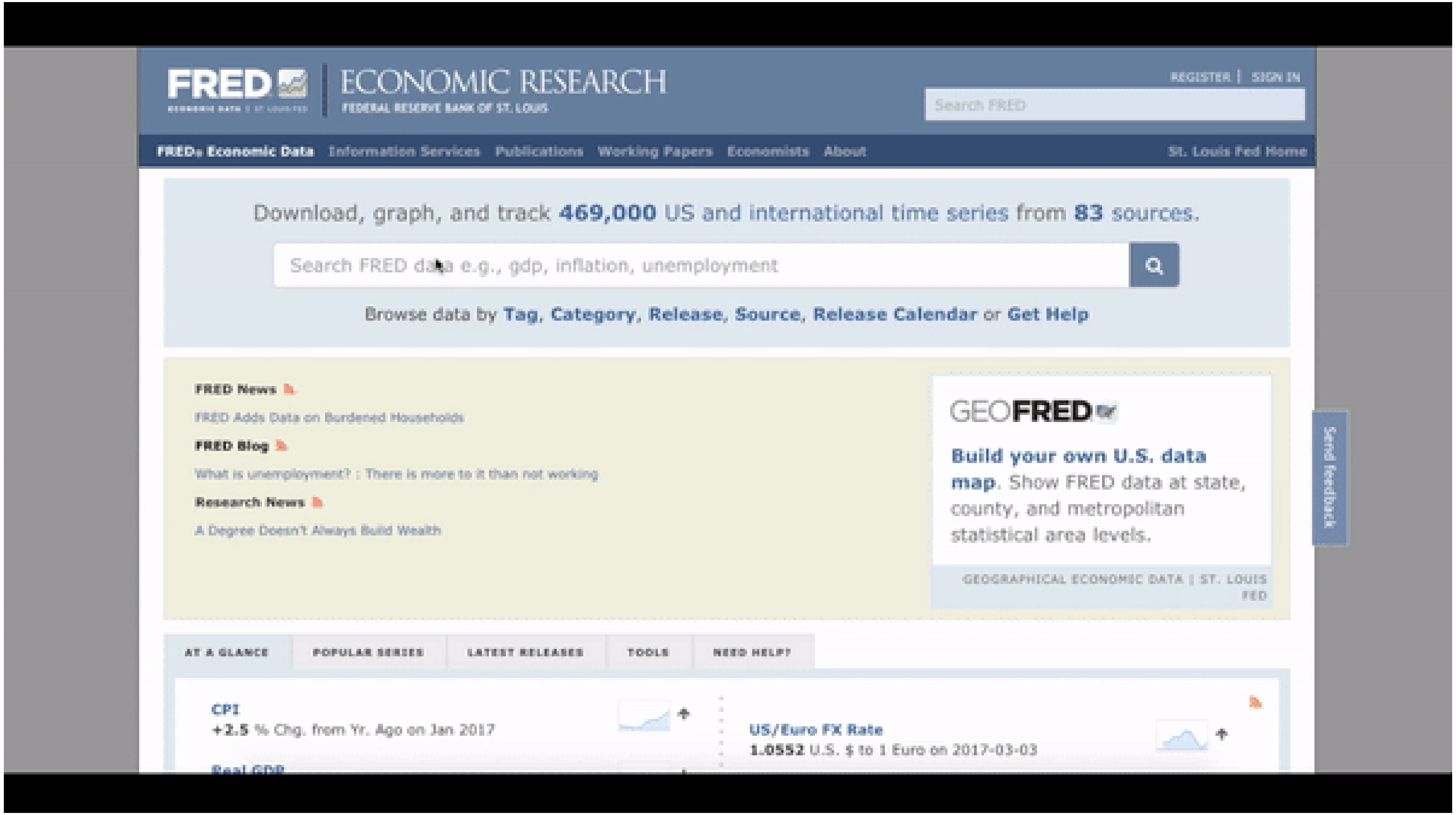


20 YEAR HISTORY, 0 REGULATORY ACTIONS

NFA regulated since 2003

When should reputation matter in selecting a broker? We think, always.
OANDA is founded on the premise that forex trading should be fair and transparent.

FRED



Download, graph, and track **469,000** US and international time series from **83** sources.

Search FRED data e.g., gdp, inflation, unemployment

Browse data by **Tag**, **Category**, **Release**, **Source**, **Release Calendar** or **Get Help**

FRED News

FRED Adds Data on Burdened Households

FRED Blog

What is unemployment? : There is more to it than not working

Research News

A Degree Doesn't Always Build Wealth

GEOFRED

Build your own U.S. data map. Show FRED data at state, county, and metropolitan statistical area levels.

GEOGRAPHICAL ECONOMIC DATA | ST. LOUIS FED

CPI
+2.5 % Chg. from Yr. Ago on Jan 2017

US/Euro FX Rate
1.0552 U.S. $ to 1 Euro on 2017-03-03

Real GDP

Let's practice!

IMPORTING AND MANAGING FINANCIAL DATA IN R

Extracting columns from financial time series

IMPORTING AND MANAGING FINANCIAL DATA IN R

Joshua Ulrich
Instructor

OHLC

- Stands for "Open High Low Close"
- *Open and Close*: first and last observed prices
- *High and Low*: largest and smallest observed prices
- Often *Volume*: sum of all contracts traded

OHLC data

```
head(DC)
```

```
                    DC.Open DC.High DC.Low DC.Close DC.Volume
2016-01-16 01:00:00  20.845  20.850 20.835   20.845       157
2016-01-16 02:00:00  20.845  20.850 20.835   20.845       214
2016-01-16 03:00:00  20.845  20.850 20.835   20.845       103
2016-01-16 04:00:00  20.845  20.855 20.835   20.845       180
2016-01-16 05:00:00  20.845  20.845 20.845   20.845       211
2016-01-16 06:00:00  20.845  20.845 20.840   20.845        35
```

Single-column extractor functions

- `Op()` - opening price
- `Hi()` - high price
- `Lo()` - low price
- `Cl()` - close price
- `Vo()` - traded volume
- `Ad()` - adjusted close price

Single-column extractor functions

```
# Open price
dc_open <- Op(DC)
head(dc_open, 4)
```

```
                    DC.Open
2016-01-16 01:00:00   20.84
2016-01-16 02:00:00   20.85
2016-01-16 03:00:00   20.85
2016-01-16 04:00:00   20.85
```

```
# High price
dc_high <- Hi(DC)
head(dc_high, 4)
```

```
                    DC.High
2016-01-16 01:00:00   20.85
2016-01-16 02:00:00   20.85
2016-01-16 03:00:00   20.85
2016-01-16 04:00:00   20.85
```

Multi-column extractor functions

```
# Extract multiple columns
dc_ohlc <- OHLC(DC)
head(dc_ohlc)
```

```
                    DC.Open DC.High DC.Low DC.Close
2016-01-16 01:00:00   20.84   20.85  20.83    20.84
2016-01-16 02:00:00   20.85   20.85  20.83    20.85
2016-01-16 03:00:00   20.85   20.85  20.84    20.85
2016-01-16 04:00:00   20.85   20.85  20.84    20.85
2016-01-16 05:00:00   20.85   20.85  20.84    20.85
2016-01-16 06:00:00   20.84   20.85  20.84    20.85
```

getPrice()

- 3 arguments
  - `x` : object that contains data
  - `symbol` : optional symbol if `x` contains multiple symbols
  - `prefer` : optional preferred price
- If `prefer` not specified:
  - `"price"` , then `"trade"` , then `"close"`

```
head(DC)
```

```
                    Price Volume Bid.Price Bid.Size Ask.Price Ask.Size
2016-01-16 00:00:07    NA     NA     20.84      198     20.85      684
2016-01-16 00:00:08    NA     NA     20.84      198     20.85      683
2016-01-16 00:00:08    NA     NA     20.84      198     20.85      682
2016-01-16 00:00:11    NA     NA     20.84      198     20.85      683
2016-01-16 00:00:25    NA     NA     20.84      198     20.85      684
2016-01-16 00:00:44 20.84      1     20.84      198     20.85      684
```

```
dc_bid <- getPrice(DC, prefer = "bid")
head(dc_bid)
```

```
                    Bid.Price
2016-01-16 00:00:07     20.84
2016-01-16 00:00:08     20.84
2016-01-16 00:00:08     20.84
2016-01-16 00:00:11     20.84
2016-01-16 00:00:25     20.84
2016-01-16 00:00:44     20.84
```

Let's practice!

IMPORTING AND MANAGING FINANCIAL DATA IN R

Importing and transforming multiple instruments

IMPORTING AND MANAGING FINANCIAL DATA IN R

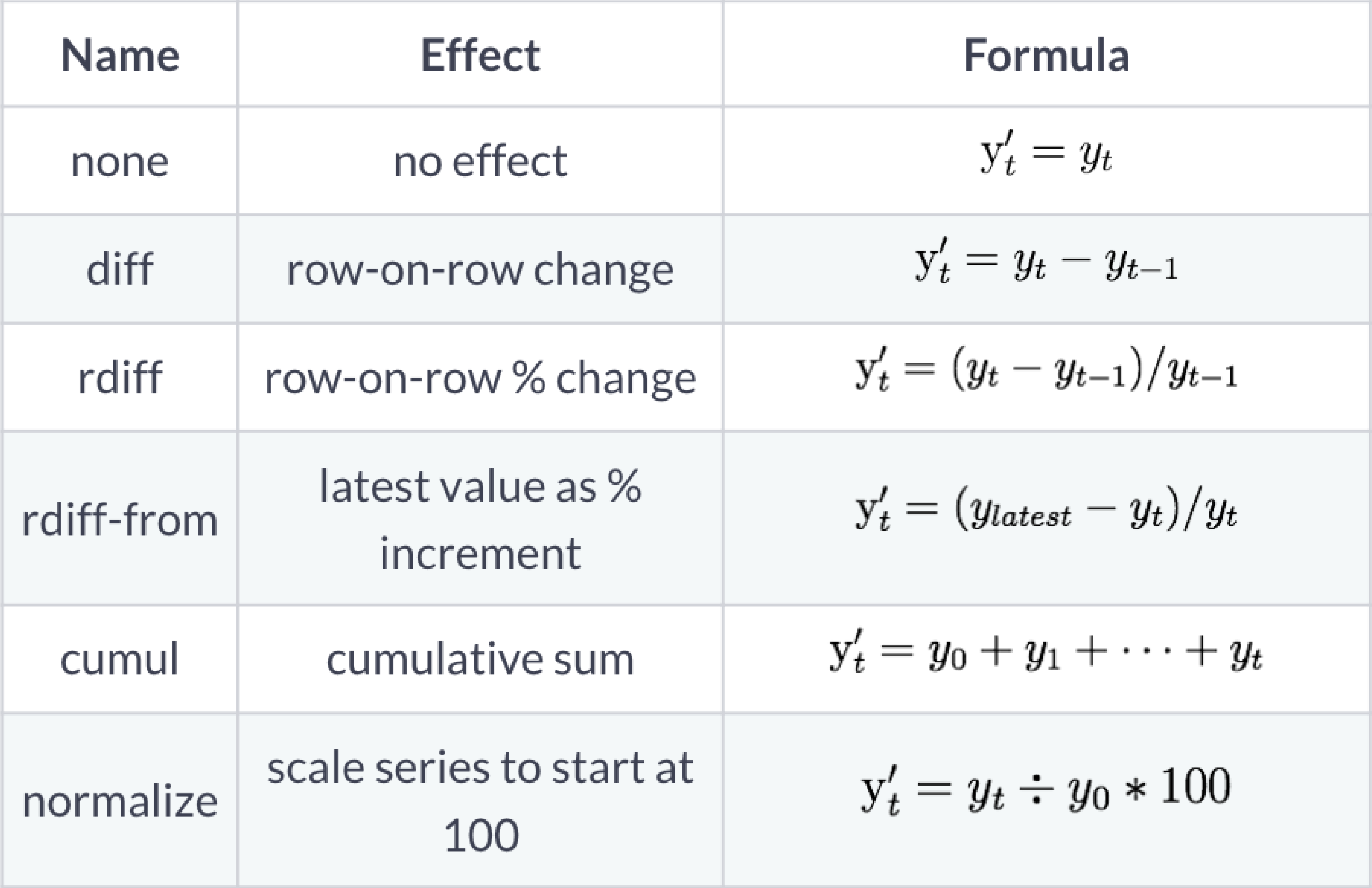
Joshua Ulrich
Instructor

Aggregating with Quandl()

- Use `collapse` argument to aggregate
  - daily
  - weekly
  - monthly
  - quarterly
  - annual
- Always returns last observation for given time period
  - Can cause issues for some columns (e.g., opening price)

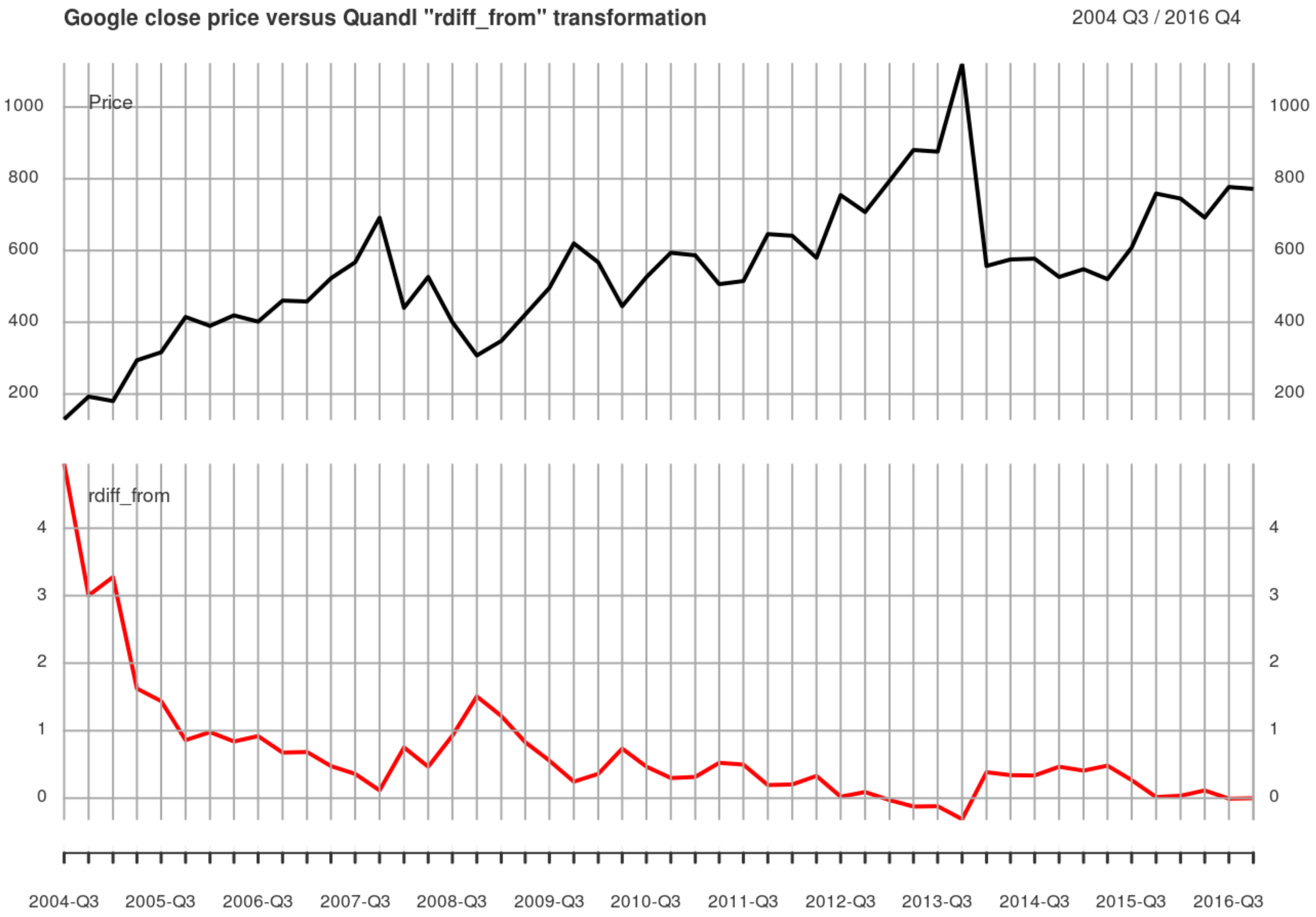
Transforming with Quandl()

- Use `transform` argument to perform simple calculations before downloading

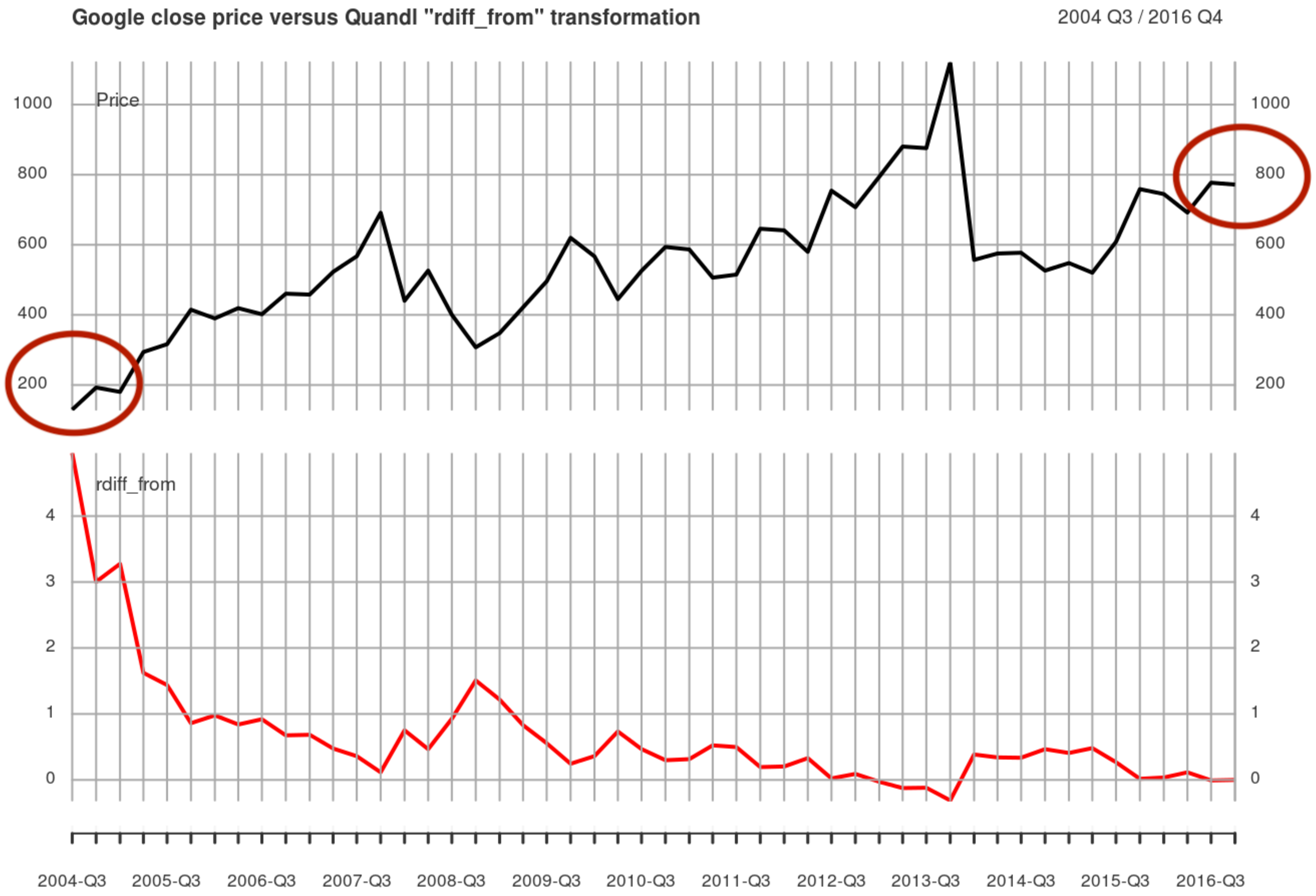


Name	Effect	Formula
none	no effect	$y'_t = y_t$
diff	row-on-row change	$y'_t = y_t - y_{t-1}$
rdiff	row-on-row % change	$y'_t = (y_t - y_{t-1})/y_{t-1}$
rdiff-from	latest value as % increment	$y'_t = (y_{latest} - y_t)/y_t$
cumul	cumulative sum	$y'_t = y_0 + y_1 + \cdots + y_t$
normalize	scale series to start at 100	$y'_t = y_t \div y_0 * 100$

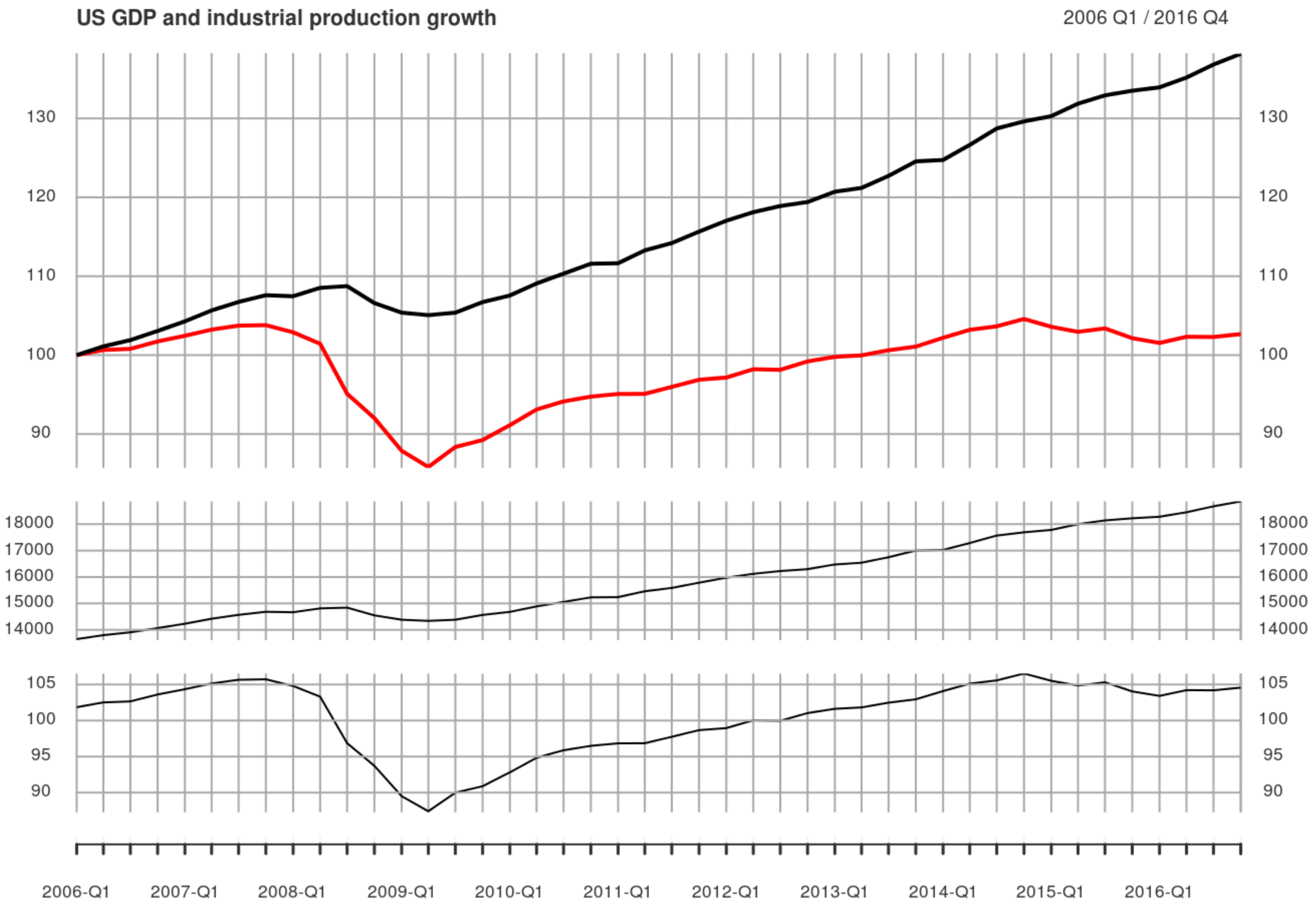
Quandl() rdiff_from transformation



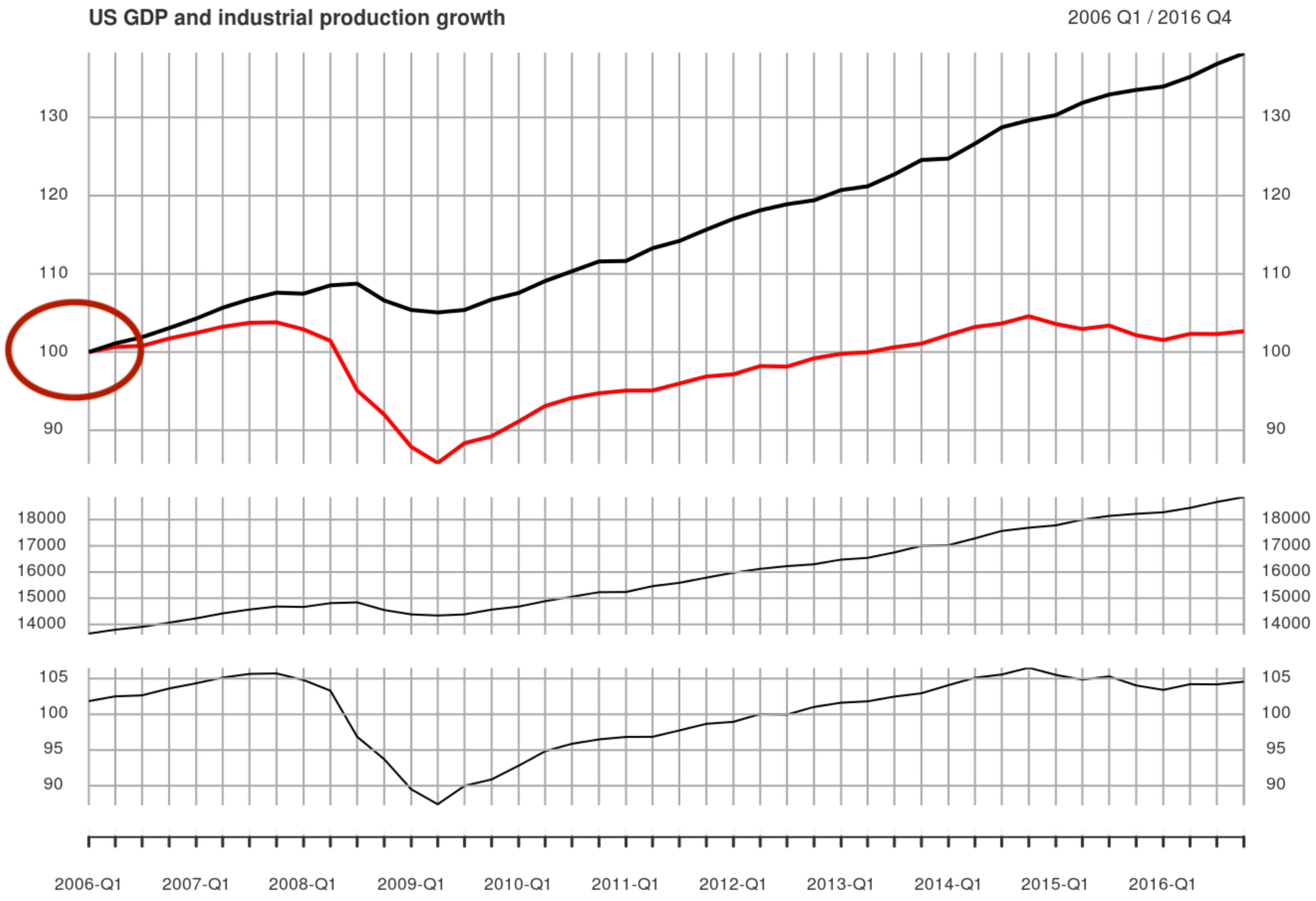
Quandl() rdiff_from transformation



Quandl() normalize transformation



Quandl() normalize transformation



Download instruments into a custom environment

```
# Create new environment
data_env <- new.env()
# Use getSymbols to load data into the environment
getSymbols(c("SPY", "QQQ"), env = data_env, auto.assign = TRUE)
```

```
"SPY" "QQQ"
```

```
# Look at a few rows of the SPY data
head(data_env$SPY, 3)
```

```
           SPY.Open SPY.High SPY.Low SPY.Close SPY.Volume SPY.Adjusted
 2007-01-03   142.25   142.86  140.57    141.37   94807600     114.8094
 2007-01-04   141.23   142.05  140.61    141.67   69620600     115.0530
 2007-01-05   141.33   141.40  140.38    140.54   76645300     114.1353
```

Using lapply()

- Loops over all objects in environment
- Combine list of function results into one object using `do.call()`
  - First argument (`what`) is the function to be called
  - Second argument (`args`) is a list of arguments to pass

Extract volume and merge into one object

```r
# Extract volume column from each object
adjusted_list <- lapply(data_env, Ad)
# Merge each list element into one object
adjusted <- do.call(merge, adjusted_list)
```

```r
head(adjusted)
```

```
           QQQ.Adjusted SPY.Adjusted
2007-01-03     39.47694     114.8094
2007-01-04     40.22558     115.0530
2007-01-05     40.03385     114.1353
2007-01-08     40.06124     114.6632
2007-01-09     40.26210     114.5658
2007-01-10     40.73684     114.9475
```

Let's practice!

IMPORTING AND MANAGING FINANCIAL DATA IN R

Setting default arguments for getSymbols()

IMPORTING AND MANAGING FINANCIAL DATA IN R

Joshua Ulrich
Instructor

getSymbols() “methods”

- `getSymbols()` doesn't contain code to import data
- Code for each data source are in a `getSymbols.[source]` "method"
- For example:

```
# You call getSymbols()
getSymbols("GDP", src = "FRED")

# getSymbols() calls source "method"
getSymbols.FRED("GDP")
```

- Users should not call `getSymbols()` "methods" directly

Use setDefaults() to change default data source

```
setDefaults(getSymbols, src = "FRED")
```

```
gdp <- getSymbols("GDP", auto.assign = FALSE)
# Note the 'src' attribute
str(gdp)
```

```
An 'xts' object on 1947-01-01/2016-10-01 containing:
  Data: num [1:280, 1] 243 246 250 260 266 ...
 - attr(*, "dimnames")=List of 2
  ..$ : NULL
  ..$ : chr "GDP"
  Indexed by objects of class: [Date] TZ: UTC
  xts Attributes:
List of 2
 $ src    : chr "FRED"
 $ updated: POSIXct[1:1], format: "2017-02-13 08:46:50"
```

setDefaults()

- Sets new default arguments using `name = value` pairs
- Only alters behavior for `getSymbols()`
- Stores values in global `options()`

Other arguments

- Find formal arguments for a `getSymbols()` source method
  - Use `args()` : `args(getSymbols.yahoo)`
  - Use `help()` : `help("getSymbols.yahoo")`

Default from and to values

```
args(getSymbols.yahoo)
 function (Symbols, env, return.class = "xts", index.class = "Date",
     from = "2007-01-01", to = Sys.Date(), ...)
```

```
setDefaults(getSymbols.yahoo, from = "2016-01-01", to = "2016-12-31")?
aapl <- getSymbols("AAPL", auto.assign = FALSE)
str(aapl)
```

```
 An 'xts' object on 2016-01-04/2016-12-30 containing:
   Data: num [1:252, 1:6] 102.6 105.8 100.6 98.7 98.6 ...
  - attr(*, "dimnames")=List of 2
   ..$ : NULL
   ..$ : chr [1:6] "AAPL.Open" "AAPL.High" "AAPL.Low" "AAPL.Close" ...
   Indexed by objects of class: [Date] TZ: UTC
   xts Attributes:
 List of 2
  $ src    : chr "yahoo"
  $ updated: POSIXct[1:1], format: "2017-02-13 08:46:50"
```

getDefaults()

```
getDefaults()
```

```
"getSymbols.yahoo"
```

```
getDefaults(getSymbols.yahoo)
```

```
$from
"'2016-01-01'"

$to
"'2016-12-31'"
```

- Values returned do not imply those functions to accept user-specified defaults

```
setDefaults(load,
            file = "my_file.RData")

# Will not alter behavior
getDefaults(load)
```

```
$file
"'my_file.RData'"
```

Let's practice!

IMPORTING AND MANAGING FINANCIAL DATA IN R

Setting per-instrument default arguments

IMPORTING AND MANAGING FINANCIAL DATA IN R

Joshua Ulrich
Instructor

Use setSymbolLookup() to set data source

```
setSymbolLookup(AAPL = "google")
```

```
aapl <- getSymbols("AAPL", auto.assign = FALSE)
str(aapl)  # note the 'src' attribute
```

```
An ‘xts’ object on 2007-01-03/2017-02-22 containing:
  Data: num [1:2552, 1:5] 12.3 12 12.2 12.3 12.3 ...
 - attr(*, "dimnames")=List of 2
  ..$ : NULL
  ..$ : chr [1:5] "AAPL.Open" "AAPL.High" "AAPL.Low" "AAPL.Close" ...
  Indexed by objects of class: [Date] TZ: UTC
  xts Attributes:
List of 2
 $ src    : chr "google"
 $ updated: POSIXct[1:1], format: "2017-02-23 14:16:55"
```

Use setSymbolLookup() to set other arguments

```
setSymbolLookup(MSFT = list(src = "google", from = "2016-01-01"))
```

```
msft <- getSymbols("MSFT", auto.assign = FALSE)
str(msft)  # note the 'src' attribute and first date
```

```
An ‘xts’ object on 2016-01-04/2017-02-27 containing:
  Data: num [1:290, 1:5] 54.3 54.9 54.3 52.7 52.4 ...
 - attr(*, "dimnames")=List of 2
  ..$ : NULL
  ..$ : chr [1:5] "MSFT.Open" "MSFT.High" "MSFT.Low" "MSFT.Close" ...
  Indexed by objects of class: [Date] TZ: UTC
  xts Attributes:
List of 2
 $ src    : chr "google"
 $ updated: POSIXct[1:1], format: "2017-02-23 14:20:21"
```

Save and restore defaults (1)

```r
# Set default
setSymbolLookup(AAPL = "google")
```

```r
# Verify the default changed
getSymbolLookup()
```

```
$AAPL
$AAPL$src
"google"
```

```r
# Save lookup
saveSymbolLookup("symbol_lookup.rda")
```

```r
# Remove lookup
setSymbolLookup(AAPL = NULL)
```

Save and restore defaults (2)

```r
# Verify the default is removed
getSymbolLookup()
```

```
named list()
```

```r
# Load lookup
loadSymbolLookup("symbol_lookup.rda")
```

```r
# Verify the default is restored
getSymbolLookup()
```

```
$AAPL
$AAPL$src
"google"
```

Let's practice!

IMPORTING AND MANAGING FINANCIAL DATA IN R

Handling instrument symbols that clash or are not valid R names

IMPORTING AND MANAGING FINANCIAL DATA IN R

Joshua Ulrich
Instructor

Syntactically valid names

- Valid names contain letters, numbers, `.` and `_`
- Must start with a letter, or a `.` followed by a non-number
- May not be one of the **reserved** words
- Not valid:
    - `.4times` , `_one` , `for`

Accessing objects with non-syntactic names (1)

- `getSymbols()` makes some names valid
  - S&P 500 Index: `"^GSPC"`

```
getSymbols("^GSPC")
```

```
"GSPC"
```

```
head(GSPC, 3)
```

```
           GSPC.Open GSPC.High GSPC.Low GSPC.Close GSPC.Volume GSPC.Adjusted
2007-01-03   1418.03   1429.42  1407.86    1416.60  3429160000       1416.60
2007-01-04   1416.60   1421.84  1408.43    1418.34  3004460000       1418.34
2007-01-05   1418.34   1418.34  1405.75    1409.71  2919400000       1409.71
```

Accessing objects with non-syntactic names (2)

- Some ticker symbols are not valid names
  - Shanghai Stock Exchange Composite Index: `"000001.SS"`

```r
getSymbols("000001.SS", auto.assign = TRUE)
```

```
"000001.SS"
```

```r
str(000001.SS)
```

```
Error: unexpected symbol in "str(000001.SS)"
```

```
head(`000001.SS`, n = 3)
```

```
            000001.SS.Open   000001.SS.High     000001.SS.Low
2007-01-04         2715.72          2715.72           2715.72
2007-01-05         2641.33          2641.33           2641.33
2007-01-08         2707.20          2707.20           2707.20
           000001.SS.Close 000001.SS.Volume 000001.SS.Adjusted
2007-01-04         2715.72                0            2715.72
2007-01-05         2641.33                0            2641.33
2007-01-08         2707.20                0            2707.20
```

```
head(get("000001.SS"), n = 3)
```

```
            000001.SS.Open   000001.SS.High     000001.SS.Low
2007-01-04         2715.72          2715.72           2715.72
2007-01-05         2641.33          2641.33           2641.33
2007-01-08         2707.20          2707.20           2707.20
           000001.SS.Close 000001.SS.Volume 000001.SS.Adjusted
2007-01-04         2715.72                0            2715.72
2007-01-05         2641.33                0            2641.33
2007-01-08         2707.20                0            2707.20
```

Valid name for one instrument

- Assign `getSymbols()` output to valid name
- Convert column names to valid names

```r
sse <- getSymbols("000001.SS", auto.assign = FALSE)
```

```r
colnames(sse) <- paste("SSE",
                         c("Open", "High", "Low", "Close",
                           "Volume", "Adjusted"), sep = ".")
head(sse, n = 2)
```

```
            SSE.Open    SSE.High     SSE.Low
2007-01-04   2715.72     2715.72     2715.72
2007-01-05   2641.33     2641.33     2641.33
           SSE.Close  SSE.Volume  SSE.Adjusted
2007-01-04   2715.72           0       2715.72
2007-01-05   2641.33           0       2641.33
```

- Create symbol-to-R object mapping with `setSymbolLookup()`

```
setSymbolLookup(SSE = list(name = "000001.SS"),
                  FORD = list(name = "F"))
getSymbols(c("SSE", "FORD"))
```

```
"SSE"  "FORD"
```

```
head(SSE, n = 2)
```

```
               SSE.Open  SSE.High  SSE.Low  SSE.Close  SSE.Volume  SSE.Adjusted
2007-01-04      2715.72   2715.72  2715.72    2715.72           0       2715.72
2007-01-05      2641.33   2641.33  2641.33    2641.33           0       2641.33
```

```
head(FORD, n = 2)
```

```
           FORD.Open  FORD.High FORD.Low  FORD.Close  FORD.Volume  FORD.Adjusted
2007-01-03      7.56     7.67       7.44       7.51     78652200       6.150263
2007-01-04      7.56     7.72       7.43       7.70     63454900       6.305862
```

Let's practice!

IMPORTING AND MANAGING FINANCIAL DATA IN R

Making irregular data regular

IMPORTING AND MANAGING FINANCIAL DATA IN R

Joshua Ulrich
Instructor

Regular date-time sequences

- Time observations are same distance apart
- Create regular date-time sequences using `seq()` methods:
  - `seq.Date()`
  - `seq.POSIXt()` (`POSIXct` and `POSIXlt`)

```
from_date <- as.Date("2017-01-01")
to_date <- as.Date("2017-01-03")
date_seq <- seq(from = from_date, to = to_date, by = "day")
```

- `start()` first index value
- `end()` last index value

```
regular_xts <- xts(seq_along(date_seq), order.by = date_seq)
start(regular_xts)
```

```
"2017-01-01"
```

```
end(regular_xts)
```

```
"2017-01-03"
```

```
seq(from = start(regular_xts), to = end(regular_xts), by = "day")
```

```
"2017-01-01" "2017-01-02" "2017-01-03"
```

Zero-width xts objects

- `xts` object with an index, no data

```r
zero_width_xts <- xts(, order.by = date_seq)
zero_width_xts
```

```
Data:
numeric(0)
Index:
 Date[1:3], format: "2017-01-01" "2017-01-02" "2017-01-03"
```

```r
str(zero_width_xts)
```

```
An 'xts' object of zero-width
```

Creating regular from irregular data

- Add observation at each date-time in regular sequence
- `NA` in the result

Merge irregular xts with regular zero-width xts

```
irregular
```

```
           Price
2017-01-02 20.01
2017-01-04 20.02
2017-01-10 20.05
```

```
date_seq <- seq(from = start(irregular),
                to = end(irregular),
                by = "day")
```

```
regular_xts <- xts(, date_seq)
```

Merge irregular xts with regular zero-width xts

```
merge(irregular, regular_xts)
```

```
           Price
2017-01-02 20.01
2017-01-03    NA
2017-01-04 20.02
2017-01-05    NA
2017-01-06    NA
2017-01-07    NA
2017-01-08    NA
2017-01-09    NA
2017-01-10 20.05
```

Filling missing values

```
merged_xts <- merge(irregular, regular_xts)
na.locf(merged_xts)
```

```
           Price
2017-01-02 20.01
2017-01-03 20.01
2017-01-04 20.02
2017-01-05 20.02
2017-01-06 20.02
2017-01-07 20.02
2017-01-08 20.02
2017-01-09 20.02
2017-01-10 20.05
```

Filling missing values

```
merge(irregular, regular_xts, fill = na.locf)
```

```
           Price
2017-01-02 20.01
2017-01-03 20.01
2017-01-04 20.02
2017-01-05 20.02
2017-01-06 20.02
2017-01-07 20.02
2017-01-08 20.02
2017-01-09 20.02
2017-01-10 20.05
```

Let's practice!

IMPORTING AND MANAGING FINANCIAL DATA IN R

Aggregating to lower frequency

IMPORTING AND MANAGING FINANCIAL DATA IN R

Joshua Ulrich
Instructor

Low frequency data

- Timestamps have too much resolution
- Represent the first quarter of 2017
  - `"2017-01-01"` (first)
  - `"2017-03-31"` (last)
  - `"2017-02-01"` (middle)

Example

- Compare the daily 10-year Treasury constant maturity rate with USA Gross Domestic Product (quarterly)
- FRED symbols:
  - `DGS10`
  - `GDP`

Merge aggregated data with low-frequency data

```
# Aggregate to quarterly
QGS10 <- apply.quarterly(DGS10, median, na.rm = TRUE)
# Merge quarterly aggregate with quarterly GDP
QGS10_GDP <- merge(QGS10, GDP)
QGS10_GDP
```

```
           DGS10     GDP
2015-01-01    NA 17783.6
2015-03-31  1.97      NA
2015-04-01    NA 17998.3
2015-06-30  2.19      NA
2015-07-01    NA 18141.9
2015-09-30  2.20      NA
2015-10-01    NA 18222.8
2015-12-31  2.23      NA
```

Low frequency date-time classes

- `yearmon()` for monthly data
- `yearqtr()` for quarterly data

```
as.Date("2017-01-01")
```

```
"2017-01-01"
```

```
as.yearmon("2017-01-01")
```

```
"Jan 2017"
```

```
as.yearqtr("2017-01-01")
```

```
"2017 Q1"
```

Convert index to lowest frequency

```
# Convert both indexes to yearqtr
index(QGS10) <- as.yearqtr(index(QGS10))
index(GDP) <- as.yearqtr(index(GDP))
```

```
# Merging 'just works'
merge(QGS10, GDP)
```

```
        DGS10     GDP
2015 Q1  1.97 17783.6
2015 Q2  2.19 17998.3
2015 Q3  2.20 18141.9
2015 Q4  2.23 18222.8
```

Align with beginning-of-period timestamp

```
# Last observation carried backward
QGS10_GDP_locb <- na.locf(QGS10_GDP, fromLast = TRUE)
```

```
# Subset by beginning-of-period index
QGS10_GDP_first_period <- QGS10_GDP_locb[index(GDP)]
QGS10_GDP_first_period
```

```
           DGS10     GDP
2015-01-01  1.97 17783.6
2015-04-01  2.19 17998.3
2015-07-01  2.20 18141.9
2015-10-01  2.23 18222.8
```

Let's practice!

IMPORTING AND MANAGING FINANCIAL DATA IN R

Aggregating and combining intraday data

IMPORTING AND MANAGING FINANCIAL DATA IN R

Joshua Ulrich
Instructor

Timezones!

[1] Source: https://www.shutterstock.com/video/search/time-zone-clocks

Timezones!

- Internally, `xts` index is seconds since midnight `1970-01-01` in UTC
- `merge()` uses internal index
- `merge()` result will have timezone of the first object

Timezones!

```r
datetime <- as.POSIXct("2017-01-18 10:00:00", tz = "UTC")
london <- xts(1, datetime, tzone = "Europe/London")
tokyo <- xts(1, datetime, tzone = "Asia/Tokyo")
```

```r
merge(london, tokyo)
```

```
                    london tokyo
2017-01-18 10:00:00      1     1
```

```r
merge(tokyo, london)
```

```
                    tokyo london
2017-01-18 19:00:00     1      1
```

Creating regular intraday data

```
head(dc_trades)
```

```
                    Price
2016-01-16 08:00:58 20.85
2016-01-16 08:01:56 20.85
2016-01-16 08:03:35 20.85
2016-01-16 08:07:44 20.84
2016-01-16 08:45:58 20.85
2016-01-16 08:46:49 20.85
```

Creating regular intraday data

```r
datetimes <- seq(from = as.POSIXct("2016-01-16 08:00"),
                 to = as.POSIXct("2016-01-17 18:00"),
                 by = "1 min")
```

```r
regular_xts <- xts(, order.by = datetimes)
```

```r
merged_xts <- merge(dc_trades, regular_xts)
head(merged_xts)
```

```
                    Price
2016-01-16 08:00:00    NA
2016-01-16 08:00:58 20.85
2016-01-16 08:01:00    NA
2016-01-16 08:01:56 20.85
2016-01-16 08:02:00    NA
2016-01-16 08:03:00    NA
```

Subset to trading hours

```
# All observations should be NA
all(is.na(merged_xts["2016-01-16 19:00/2016-01-17 07:00"]))
```

```
TRUE
```

```
# xts time-of-day subsetting
merged_trade_day <- merged_xts["T08:00/T18:00"]
```

```
# Now there are no observations
nrow(merged_trade_day["2016-01-16 19:00/2016-01-17 07:00"])
```

```
0
```

Fill missing values by trading day

- `split()` - `lapply()` - `rbind()` paradigm from **this DataCamp course about manipulating time series**

```
# split() data into list of non-overlapping chunks
trade_day_list <- split(merged_trade_day, "days")
```

```
# lapply() a function to each chunk (list element)
filled_trade_day_list <- lapply(trade_day_list, na.locf)
```

```
# Combine list of chunks using do.call() and rbind()
filled_trade_day <- do.call(rbind, filled_trade_day_list)
```

Aggregate irregular intraday data

- Aggregate dense intraday data with `to.period()`
  - `period` : new periodicity (e.g. seconds, hours, days, etc)
  - `k` : number of periods per new observation

Aggregate irregular intraday data (1)

```
head(dc_price)
```

```
                    DC.Price
2016-01-16 00:00:07 20.84224
2016-01-16 00:00:08 20.84225
2016-01-16 00:00:08 20.84225
2016-01-16 00:00:11 20.84225
2016-01-16 00:00:25 20.84224
2016-01-16 00:00:44 20.84224
```

```
xts_5min <- to.period(dc_price, period = "minutes", k = 5)
head(xts_5min, n = 4)
```

```
                    dc_price.Open  dc_price.High  dc_price.Low  dc_price.Close
2016-01-16 00:03:49      20.84224       20.84227      20.84140        20.84160
2016-01-16 00:09:50      20.84160       20.84160      20.84156        20.84156
2016-01-16 00:14:57      20.84156       20.84156      20.84154        20.84154
2016-01-16 00:19:23      20.84154       20.84154      20.83206        20.83211
```

```
xts_aligned <- align.time(xts_5min, n = 60 * 5)
head(xts_aligned, n = 4)
```

```
                    dc_price.Open  dc_price.High  dc_price.Low  dc_price.Close
2016-01-16 00:05:00      20.84224       20.84227      20.84140        20.84160
2016-01-16 00:05:00      20.84160       20.84160      20.84156        20.84156
2016-01-16 00:15:00      20.84156       20.84156      20.84154        20.84154
2016-01-16 00:20:00      20.84154       20.84154      20.83206        20.83211
```

Let's practice!

IMPORTING AND MANAGING FINANCIAL DATA IN R

Importing text files

IMPORTING AND MANAGING FINANCIAL DATA IN R

Joshua Ulrich
Instructor

getSymbols() with CSV files

- Well-formatted
  - One instrument per file
  - Columns: date, open, high, low, close, volume, adjusted close
- Files named `"[symbol].csv"`
- Can use `dir` argument to specify directory

getSymbols() with CSV files

- AMZN.csv

```
"Date","AMZN.Open","AMZN.High","AMZN.Low","AMZN.Close","AMZN.Volume","AMZN.Adjusted"
2002-01-02,11.13,11.01,10.46,10.87,6674703,10.87
2002-01-03,11.26,12.25,10.76,11.99,11441553,11.99
2002-01-04,12.46,12.62,11.71,12.1,12619402,12.1
```

```
getSymbols("AMZN", src = "csv")
```

```
"AMZN"
```

```
head(AMZN, 3)
```

```
           AMZN.Open AMZN.High AMZN.Low AMZN.Close AMZN.Volume AMZN.Adjusted
2002-01-02     11.13     11.01    10.46      10.87     6674703         10.87
2002-01-03     11.26     12.25    10.76      11.99    11441553         11.99
2002-01-04     12.46     12.62    11.71      12.10    12619402         12.10
```

read.zoo()

- AMZN.csv

```
"Date","AMZN.Open","AMZN.High","AMZN.Low","AMZN.Close","AMZN.Volume","AMZN.Adjusted"
2002-01-02,11.13,11.01,10.46,10.87,6674703,10.87
2002-01-03,11.26,12.25,10.76,11.99,11441553,11.99
2002-01-04,12.46,12.62,11.71,12.1,12619402,12.1
```

```r
amzn_zoo <- read.zoo("AMZN.csv", sep = ",", header = TRUE)
amzn_xts <- as.xts(amzn_zoo)
head(amzn_xts, n = 3)
```

```
           AMZN.Open AMZN.High AMZN.Low AMZN.Close AMZN.Volume AMZN.Adjusted
2002-01-02     11.13     11.01    10.46      10.87     6674703         10.87
2002-01-03     11.26     12.25    10.76      11.99    11441553         11.99
2002-01-04     12.46     12.62    11.71      12.10    12619402         12.10
```

Date and time in separate columns

- FOO.csv

```
"Date","Time","Open","High","Low","Close"
2016-11-08,09:05:00,80.9,81,80.87,81
2016-11-08,09:10:00,80.92,80.93,80.89,80.89
2016-11-08,09:15:00,80.93,80.94,80.92,80.93
```

```
foo_zoo <- read.zoo("FOO.csv", sep = ",", header = TRUE,
                      index.column = c("Date", "Time"))
head(foo_zoo,  n = 3)
```

```
                     Open  High   Low Close
2016-11-08 09:05:00 80.90 81.00 80.87 81.00
2016-11-08 09:10:00 80.92 80.93 80.89 80.89
2016-11-08 09:15:00 80.93 80.94 80.92 80.93
```

File contains multiple instruments

- BAR.csv

```
Date,Symbol,Type,Price
2016-01-01 10:43:01,A,Bid,58.23
2016-01-01 10:43:01,A,Ask,58.24
2016-01-01 10:43:01,B,Bid,28.96
2016-01-01 10:43:01,B,Ask,28.98
```

```r
bar_zoo <- read.zoo("BAR.csv",
                    split = c("Symbol", "Type"),
                    sep = ",", header = TRUE)
bar_zoo
```

```
                    A.Ask B.Ask A.Bid B.Bid
2016-01-01 10:43:01 58.24 28.98 58.23 28.96
2016-01-01 10:43:02 58.25 28.99 58.24 28.97
```

Let's practice!

IMPORTING AND MANAGING FINANCIAL DATA IN R

Checking for weirdness

IMPORTING AND MANAGING FINANCIAL DATA IN R

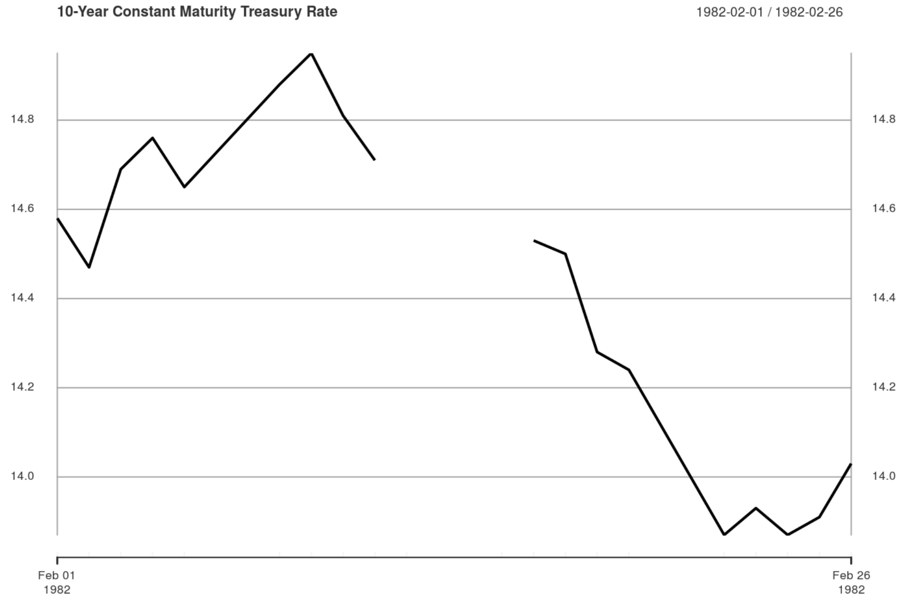
Joshua Ulrich
Instructor

Visualize Data

```
getSymbols("DGS10", src = "FRED")
```

```
"DGS10"
```

```
treasury_10 <- DGS10["1982-02"]
plot(treasury_10, main = "10-Year Constant Maturity Treasury Rate")
```



Handle missing values

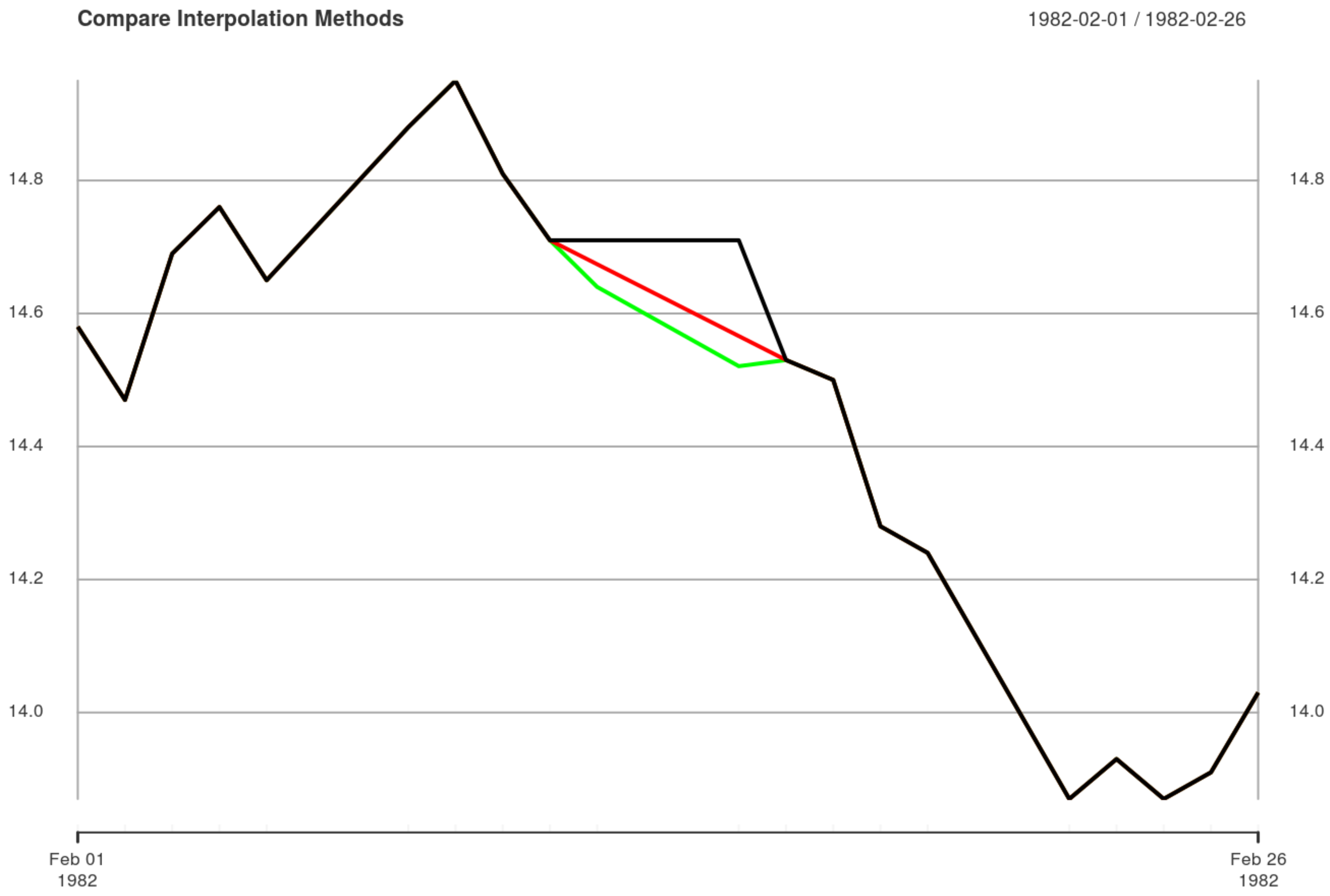
```r
# Fill NA using last observation carried forward
locf <- na.locf(treasury_10)
```

```r
# Fill NA using linear interpolation
approx <- na.approx(treasury_10)
```

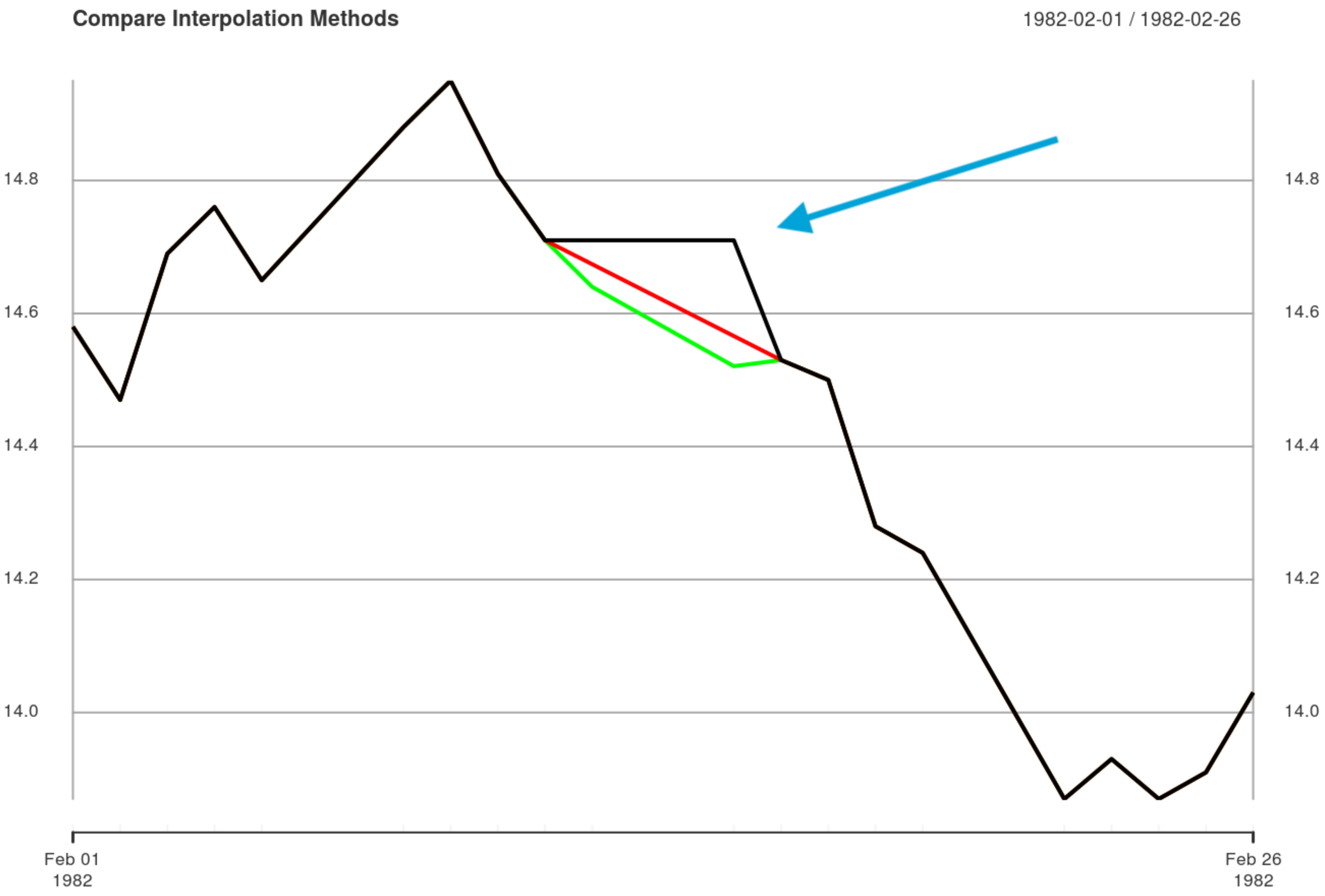
```r
# Fill NA using spline interpolation
spline <- na.spline(treasury_10)
```

```r
# Merge into one object
na_filled <- merge(locf, approx, spline)
# Plot combined object
plot(na_filled, col = c("black", "red", "green"),
     main = "Compare Interpolation Methods")
```

Handle missing values



Handle missing values

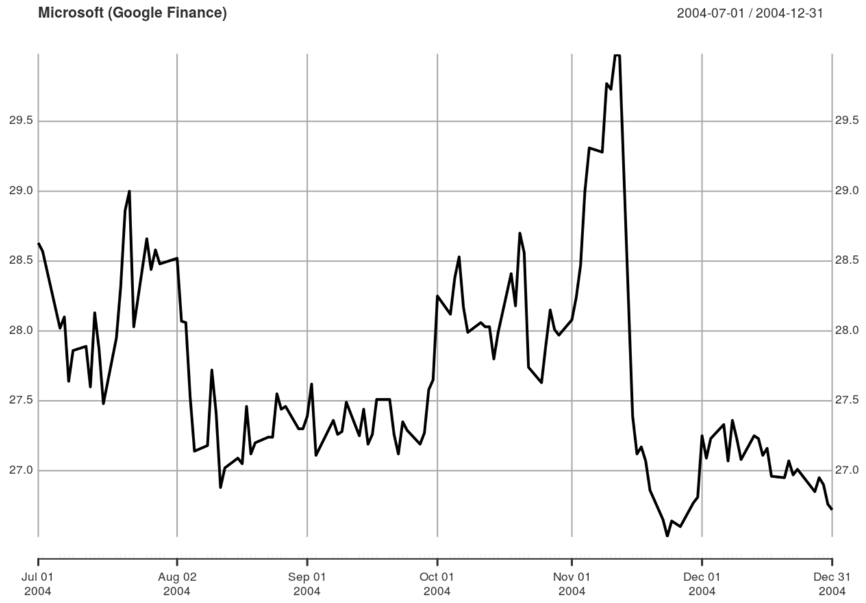


Visualize data

```
getSymbols("MSFT", from = "2004-07-01", to = "2004-12-31", src = "google")
```

```
"MSFT"
```

```
plot(Cl(MSFT), main = "Microsoft (Google Finance)")
```

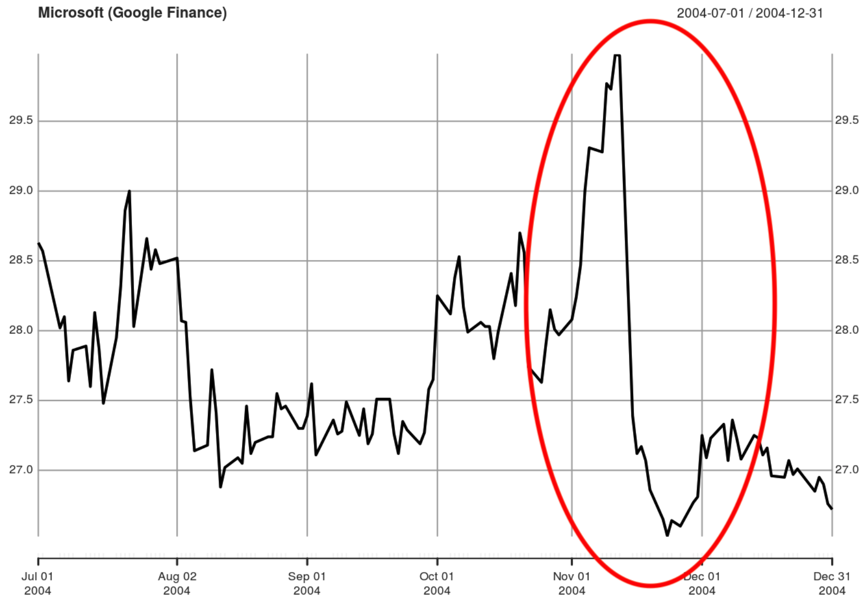


Visualize data

```
getSymbols("MSFT", from = "2004-07-01", to = "2004-12-31", src = "google")
```

```
"MSFT"
```

```
plot(Cl(MSFT), main = "Microsoft (Google Finance)")
```

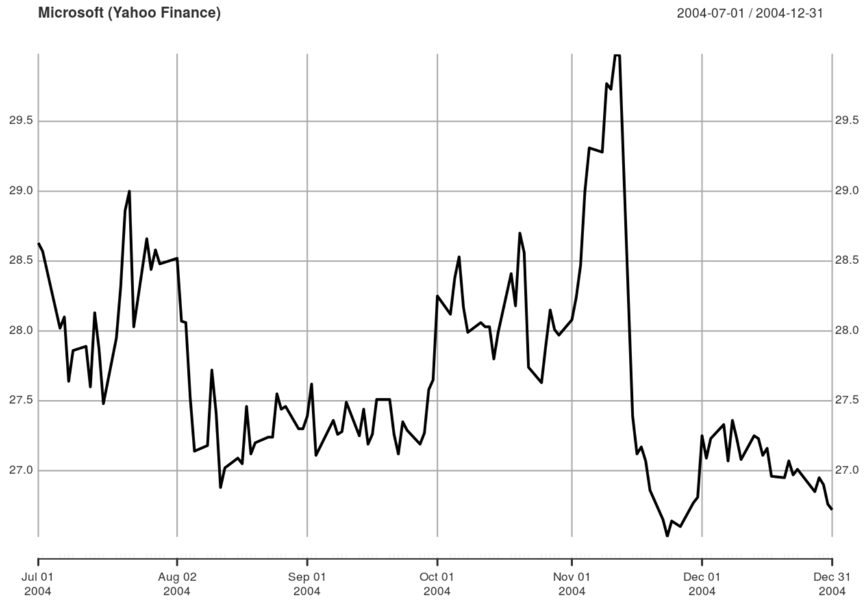


Cross-reference sources

```
getSymbols("MSFT", from = "2004-07-01", to = "2004-12-31")
```

```
"MSFT"
```

```
plot(Cl(MSFT), main = "Microsoft (Yahoo Finance)")
```

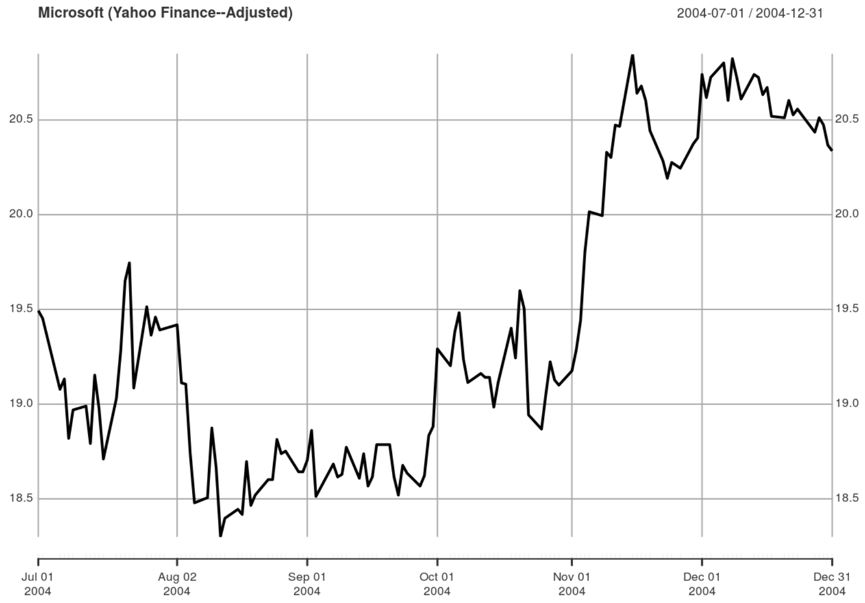


Cross-reference sources

```
getSymbols("MSFT", from = "2004-07-01", to = "2004-12-31")
```

```
"MSFT"
```

```
plot(Ad(MSFT), main = "Microsoft (Yahoo Finance-Adjusted)")
```

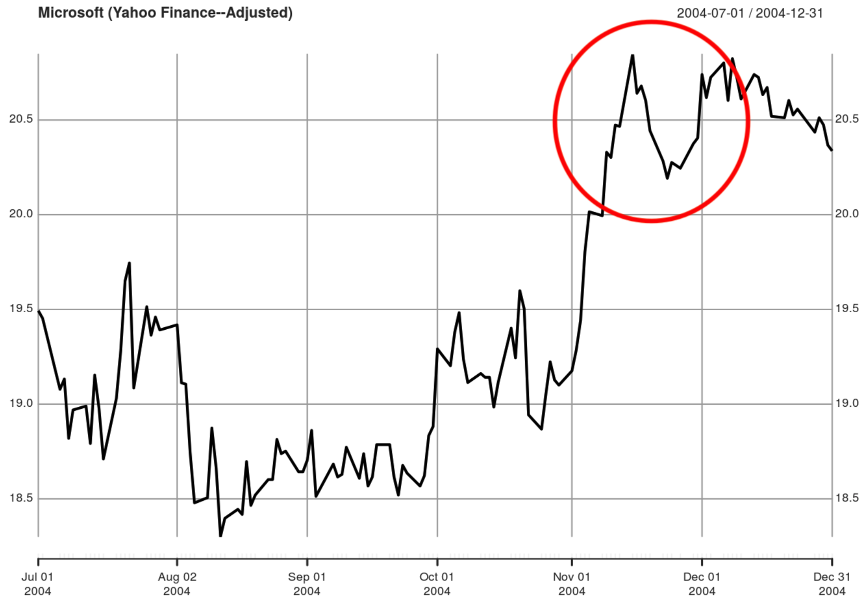


Cross-reference sources

```
getSymbols("MSFT", from = "2004-07-01", to = "2004-12-31")
```

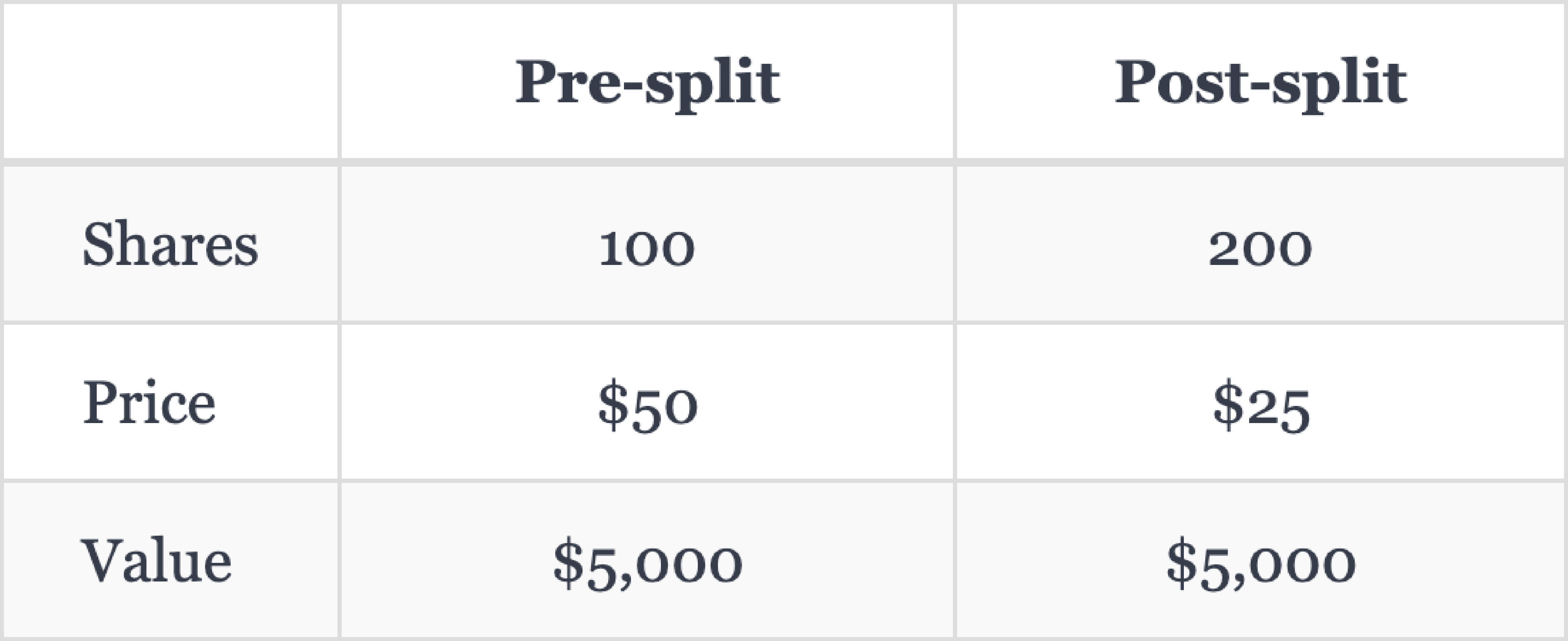
```
"MSFT"
```

```
plot(Ad(MSFT), main = "Microsoft (Yahoo Finance–Adjusted)")
```



Stock split example

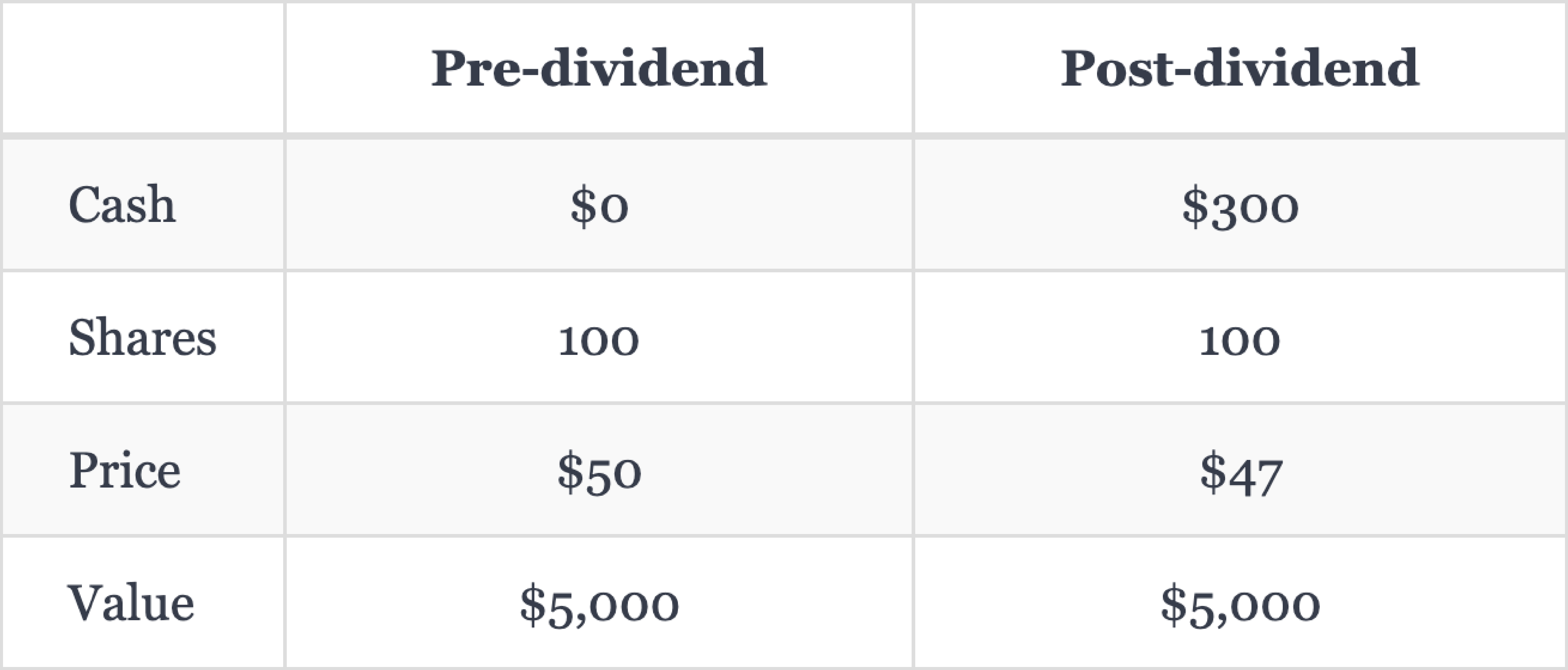
- MSFT stock splits 2-for-1



	Pre-split	Post-split
Shares	100	200
Price	$50	$25
Value	$5,000	$5,000

Stock dividend example

- MSFT issues a $3 per share dividend



	Pre-dividend	Post-dividend
Cash	$0	$300
Shares	100	100
Price	$50	$47
Value	$5,000	$5,000

Data source differences

- Yahoo Finance:
  - Raw OHLC prices
  - Split- and dividend-adjusted close
- Google Finance:
  - Split-adjusted OHLC prices

Let's practice!

IMPORTING AND MANAGING FINANCIAL DATA IN R

Adjusting for corporate actions

IMPORTING AND MANAGING FINANCIAL DATA IN R

Joshua Ulrich
Instructor

Adjust for stock splits and dividends (1)

```r
getSymbols("MSFT", from = "2004-07-01", to = "2004-12-31")
```

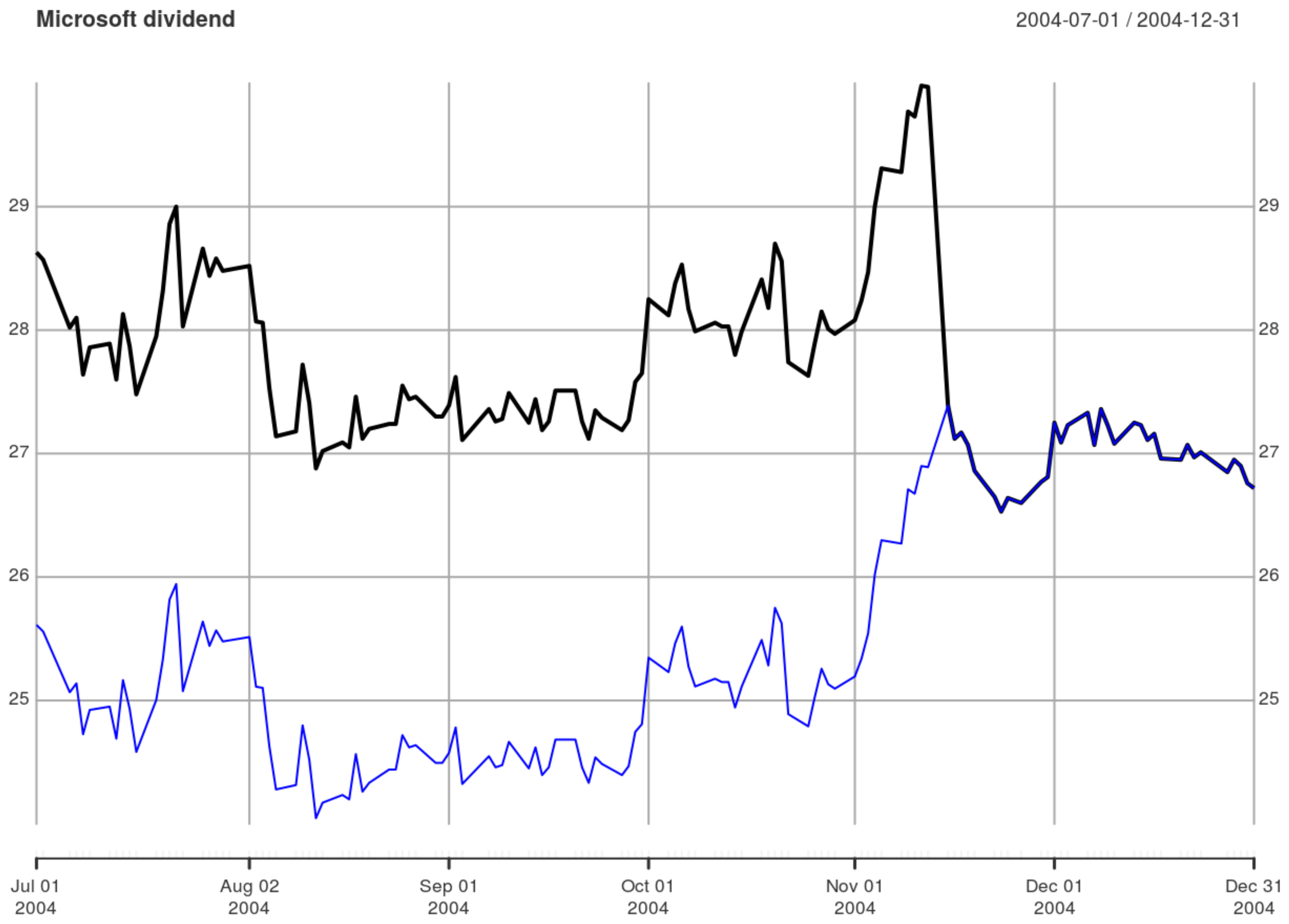
```
"MSFT"
```

```r
# Adjust data for splits and dividends
msft_adjusted <- adjustOHLC(MSFT)
```

```r
# Object name is not ticker symbol
my_data <- MSFT

# Use symbol.name argument
my_data_adjusted <- adjustOHLC(my_data, symbol.name = "MSFT")
```

Adjust for stock splits and dividends (2)



```r
# Download split data from Yahoo Finance
splits <- getSplits("GE")
head(splits, n = 4)
```

```
           GE.spl
1971-06-08    0.5
1983-06-02    0.5
1987-05-26    0.5
1994-05-16    0.5
```

```r
# Download split-adjusted dividend data from Yahoo Finance
dividends <- getDividends("GE")
head(dividends, n = 4)
```

```
            GE.div?1970-03-03 0.00677
1970-06-11 0.00677
1970-09-21 0.00677
1970-12-07 0.00677
```

Download unadjusted dividends

```r
# Download unadjusted dividend data from Yahoo Finance
dividends_raw <- getDividends("GE", split.adjust = FALSE)

# Compare adjusted and unadjusted dividends
head(merge(dividends, dividends_raw))
```

```
            GE.div GE.div.1
1970-03-03 0.00677  0.64992
1970-06-11 0.00677  0.64992
1970-09-21 0.00677  0.64992
1970-12-07 0.00677  0.64992
1971-03-03 0.00677  0.64992
1971-06-17 0.00729  0.34992
```

adjRatios()

- Back-adjust any series for splits, dividends, or both
- Has 3 arguments:
  - `splits`
  - `dividends`
  - `close`
- Returns `xts` object with 2 columns: `Split` and Div

Adjust univariate series for splits and dividends

```
getSymbols("GE", from = "2000-01-01")
```

```
"GE"
```

```
close <- Cl(GE)
splits <- getSplits("GE")
dividends_raw <- getDividends("GE", split.adjust = FALSE)
```

```
# Pass splits, unadjusted dividends, and unadjusted close
ratios <- adjRatios(splits = splits,
                    dividends = dividends_raw,
                    close = close)
```

Adjust univariate series for splits and dividends

```r
# Multiply unadjusted close by split and dividend ratios
close_adjusted <- close * ratios[, "Split"] * ratios[, "Div"]

head(merge(close, close_adjusted, Ad(GE)), n = 4)
```

```
           GE.Close GE.Close.1 GE.Adjusted
2000-01-03 150.0000   29.50422    29.44630
2000-01-04 144.0000   28.32405    28.26845
2000-01-05 143.7500   28.27488    28.21937
2000-01-06 145.6718   28.65289    28.59664
```

Let's practice!

Congratulations!

IMPORTING AND MANAGING FINANCIAL DATA IN R

Joshua Ulrich
Instructor

Congratulations!

IMPORTING AND MANAGING FINANCIAL DATA IN R