Extracting columns from financial time series

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   20.84   20.85  20.83    20.84       157
 2016-01-16 02:00:00   20.85   20.85  20.83    20.85       214
 2016-01-16 03:00:00   20.85   20.85  20.84    20.85       103
 2016-01-16 04:00:00   20.85   20.85  20.84    20.85       180
 2016-01-16 05:00:00   20.85   20.85  20.84    20.85       211
 2016-01-16 06:00:00   20.84   20.85  20.84    20.85        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`

Extract other columns using `getPrice()`

```
> 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 transforming multiple instruments

Aggregating with `Quandl()`

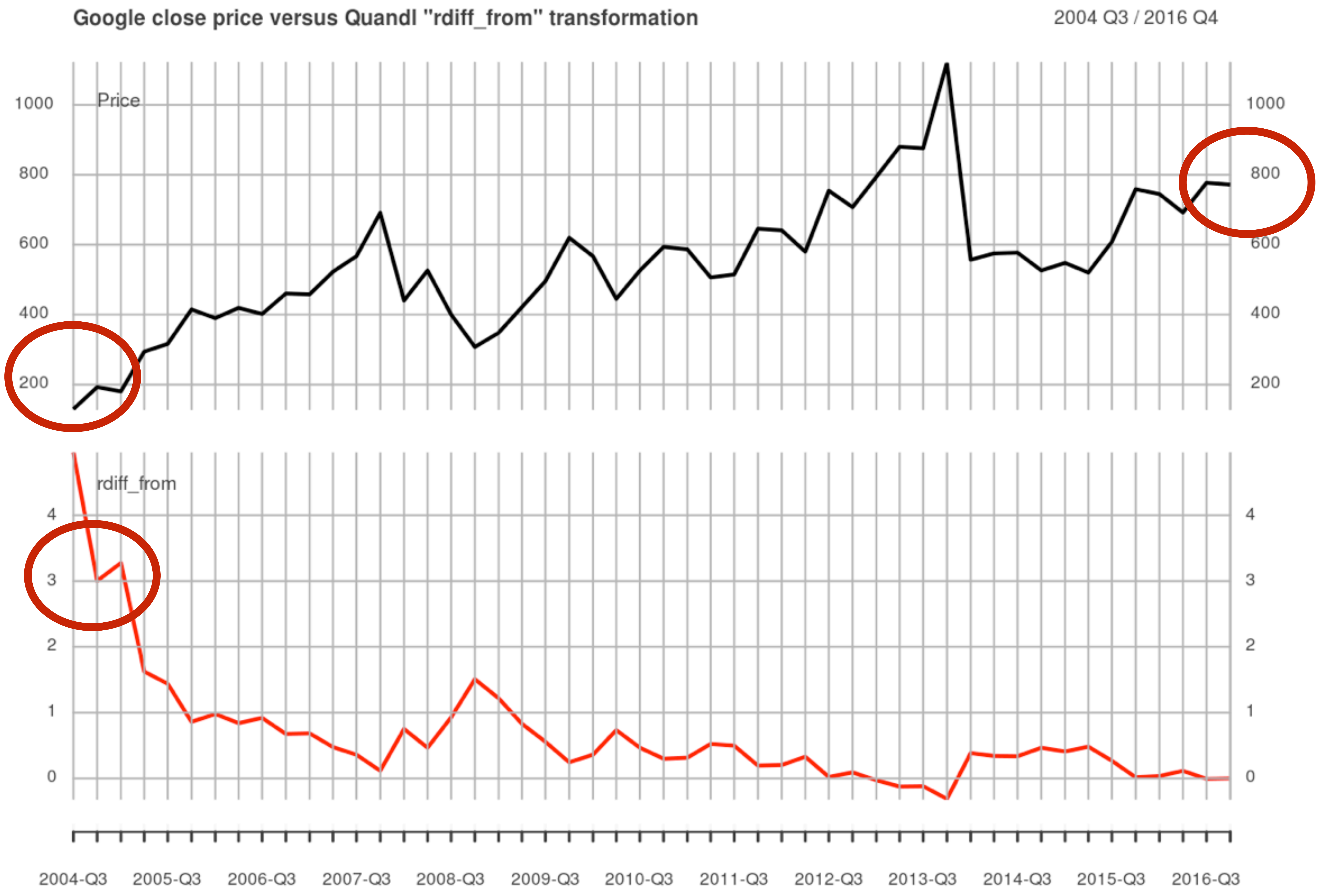
- Use `collapse` argument to aggregate
  - daily
  - weekly
  - monthly
  - quarterly
  - annual
- Always returns last observation for given time period
  - Can cause issues for some columns (e.g. opening price)

Transforming with `Quandl()`

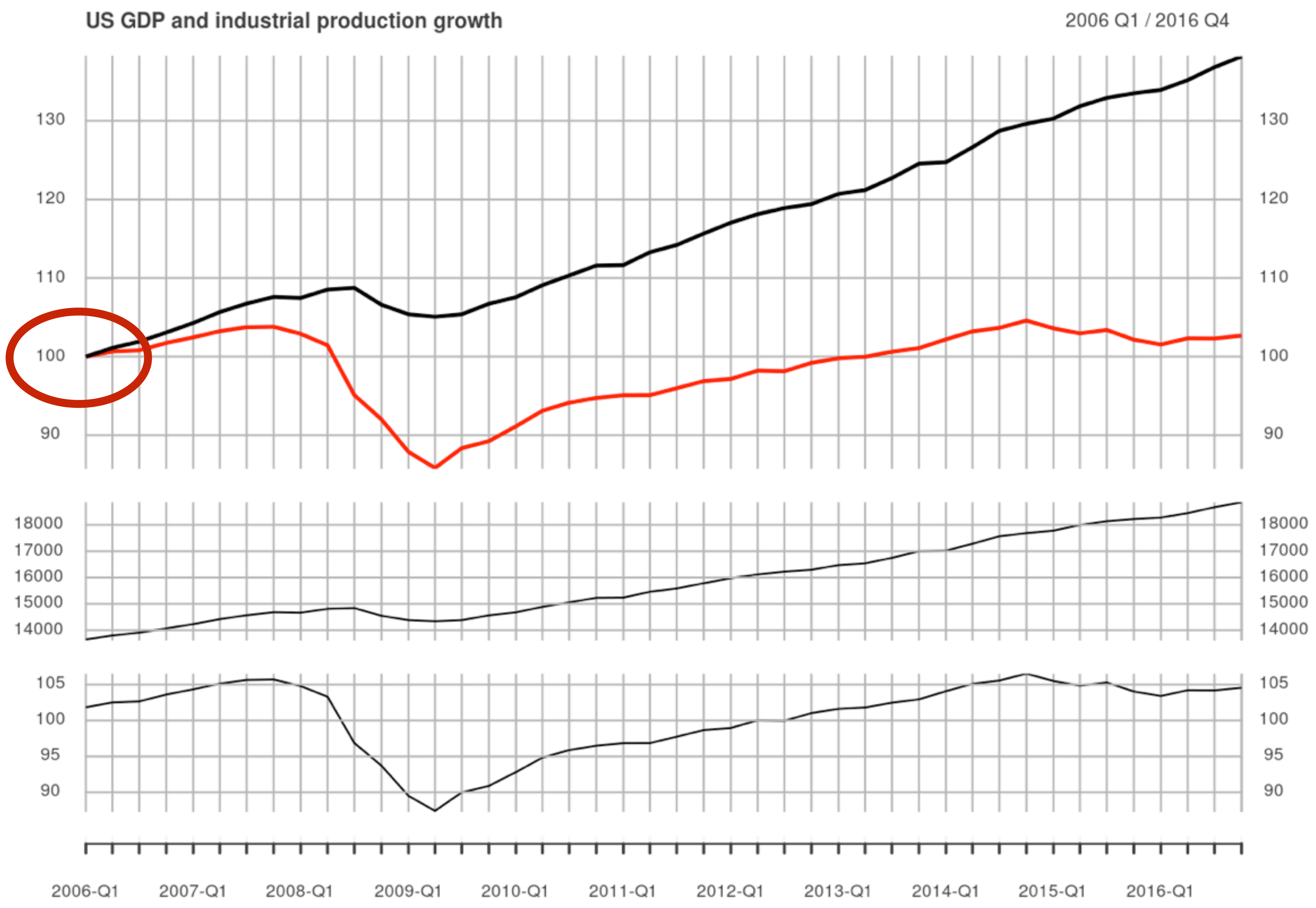
- Use `transform` argument to perform simple calculations before downloading

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 `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)
[1] "SPY" "QQQ"

> # Look at a few rows of the SPY data
> head(data_env$SPY)
           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
 2007-01-08   140.82   141.41  140.25    141.19   71655000     114.6632
 2007-01-09   141.31   141.60  140.40    141.07   75680100     114.5658
 2007-01-10   140.58   141.57  140.30    141.54   72428000     114.9475
```

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

```
> # 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)
> 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

> # The above is equivalent to:
> more_typing <- merge(adjusted_list[[1]], adjusted_list[[2]])
```

Let’s practice!