ECOG314 – Starbucks returns

FRB

November 14, 2016

Contents

Download Starbux monthly return data

Let us get started by downloading the monthly return data from sbuxPrices.csv, and by using the read.csv()

```r
# Assign the URL to the CSV file
data_url <- "http://assets.datacamp.com/course/compfin/sbuxPrices.csv"

# Load the data frame using read.csv
#sbux_df <- read.csv(file = data_url, header = TRUE, stringsAsFactors = FALSE)

#saving file as rds -- to be able to run in Linux
#saveRDS(sbux_df, "sbux_df.rds")

#read RDS file
sbux_df <- readRDS("sbux_df.rds")
```

Take a look at the data

Before you analyze return data, it is a good idea to have (at least) a quick look at the data. R has a number of functions that help you do that:

- str – display data structure
- head – display first part of data
- tail – display last part of the data
- class – show tha class of the object

These expressions will all extract the first five closing prices. If you do not provide anything for the rows (or columns), all rows (or columns) will be selected (e.g. sbux_df[, "Adj.Close"]).

```
# The sbux_df data frame is already loaded in your work space

# Check the structure of sbux_df
str(sbux_df)
```

```
OUTPUT>  'data.frame':  181 obs. of  2 variables:
OUTPUT>   $ Date     : chr  "3/31/1993" "4/1/1993" "5/3/1993" "6/1/1993" ...
OUTPUT>   $ Adj.Close: num  1.13 1.15 1.43 1.46 1.41 1.44 1.63 1.59 1.32 1.32 ...
```

```
# Check the first and last part of sbux_df
head(sbux_df, n=3)
```

```
OUTPUT>         Date Adj.Close
OUTPUT>  1 3/31/1993      1.13
OUTPUT>  2  4/1/1993      1.15
OUTPUT>  3  5/3/1993      1.43
```

```
tail(sbux_df, n=4)
```

```
OUTPUT>           Date Adj.Close
OUTPUT>  178 12/3/2007     19.46
OUTPUT>  179  1/2/2008     17.98
OUTPUT>  180  2/1/2008     17.10
OUTPUT>  181  3/3/2008     16.64
```

```
# Get the class of the Date column of sbux_df
class(sbux_df$Date)
```

```
OUTPUT>  [1] "character"
```

Extract part of the data

You can use square brackets to extract data from the sbux_df data frame like this sbux_df[rows, columns]. To specify which rows or columns to extract, you have several options:

- sbux_df[1:5, "Adj.Close"]
- sbux_df[1:5, 2]
- sbux_df$Adj.Close[1:5].

```
# The sbux_df data frame is already loaded in your work space
closing_prices <- sbux_df[, "Adj.Close", drop = FALSE]
head(closing_prices, n=5)
```

```
OUTPUT>    Adj.Close
OUTPUT>  1      1.13
OUTPUT>  2      1.15
OUTPUT>   3       1.43
OUTPUT>   4       1.46
OUTPUT>   5       1.41
```

Find indices associated with certain dates

It will often be useful to select stock data between certain dates. Advanced users are advised to look at the xts package.

However, base R also provides sufficient functionality to do this, including:

- which() function – returns the indices for which a condition is TRUE. For example: which(sbux_df$Date == “3/1/1994”) returns the position of the date 3/1/1994, which indicates in this case the row number in the sbux_df data frame.

```
# The sbux_df data frame is already loaded in your work space

# Find indices associated with the dates 3/1/1994 and 3/1/1995
index_1 <- which(sbux_df$Date == "3/1/1994")
index_2 <- which(sbux_df$Date == "3/1/1995")

# Extract prices between 3/1/1994 and 3/1/1995
some_prices <- sbux_df[index_1:index_2, "Adj.Close"]

cbind( head(some_prices, n=4) )
```

```
OUTPUT>        [,1]
OUTPUT>   [1,] 1.45
OUTPUT>   [2,] 1.77
OUTPUT>   [3,] 1.69
OUTPUT>   [4,] 1.50
```

Subsetting directly

A convinient way is to select the price on 3/1/1994 simply with sbux_prices_df[“3/1/1994”, 1]

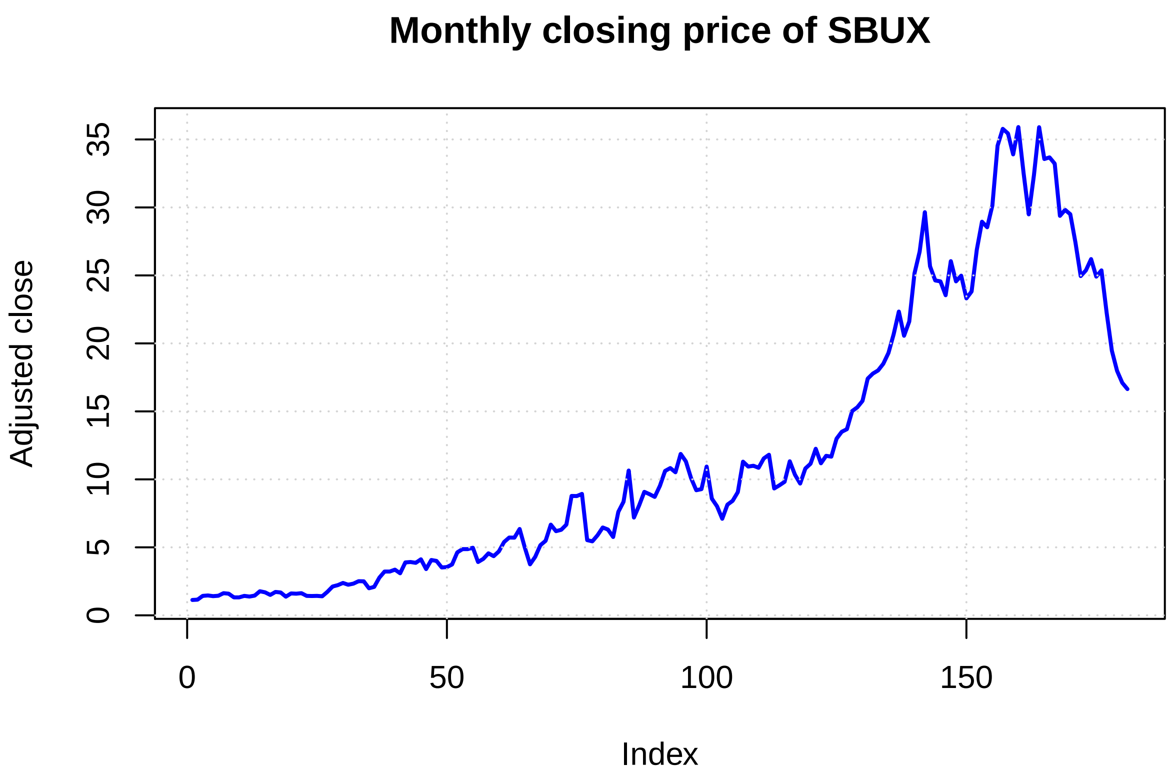
```
# Create a new data frame that contains the price data with the dates as the row names
sbux_prices_df <- sbux_df[, "Adj.Close", drop = FALSE]
rownames(sbux_prices_df) <- sbux_df$Date
#head(sbux_prices_df, n=3)

# With Dates as rownames, you can subset directly on the dates.
# Find indices associated with the dates 3/1/1994 and 3/1/1995.
price_1 <- sbux_prices_df["3/1/1994", ]
price_2 <- sbux_prices_df["3/1/1995", ]

#--
#formated output
cat(sprintf("\n ==> Price 1 = %.2f  and price 2 = %.1f\n", price_1, price_2))
```

```
OUTPUT>
OUTPUT>   ==> Price 1 = 1.45  and price 2 = 1.4
```

Plot the price data

This plot was generated with plot(sbux_df$Adj.Close)

```r
# We use base R plot function below to get a nicer plot
plot(sbux_df$Adj.Close, type = "l", col = "blue",
     lwd = 2, ylab = "Adjusted close",
     main = "Monthly closing price of SBUX")
grid()
```



Calculate simple returns

If you denote by the stock price at the end of month t, the simple return is given by:

- R[t] = (P[t] - P[t-1]) / P[t-1] - the percentage price difference.

Task

Our task in this exercise is to compute the simple returns for every time point n.

Hint

Use vectorization instead of loop to calculate the price difference over time. For example:

- sbux_prices_df[2:n,1] - sbux_prices_df[1:(n - 1),1]

Note:

- The first vector contains all prices, except the price on the first day.
- The second vector contains all prices except the price on the last day.

Given the fact that R takes the element-wise difference of these vectors, you get:

- P[t] - P[t-1]

for every t.

```
sbux_prices_df <- sbux_df[, "Adj.Close", drop = FALSE]

# Denote n the number of time periods:
n <- nrow(sbux_prices_df)
sbux_ret <- (sbux_prices_df[2:n, 1] - sbux_prices_df[1:(n - 1), 1]) / sbux_prices_df[1:(n - 1), 1]

# Notice that sbux_ret is not a data frame object
class(sbux_ret)
```

```
OUTPUT>  [1] "numeric"
```

```
#--
# library stringr
library(stringr)
data.frame(value_t=sbux_prices_df[, 1], value_t_minus_1=c("NA", sbux_prices_df[2:n, 1]), returns=c("NA"
```

```
OUTPUT>     value_t value_t_minus_1  returns
OUTPUT>  1     1.13              NA       NA
OUTPUT>  2     1.15            1.15   0.0177
OUTPUT>  3     1.43            1.43  0.24348
OUTPUT>  4     1.46            1.46  0.02098
OUTPUT>  5     1.41            1.41 -0.03425
OUTPUT>  6     1.44            1.44  0.02128
OUTPUT>  7     1.63            1.63  0.13194
OUTPUT>  8     1.59            1.59 -0.02454
OUTPUT>  9     1.32            1.32 -0.16981
OUTPUT>  10    1.32            1.32        0
```

Add dates to simple return vector

It would be nice to have the dates as names for the elements of that vector.

Remember that the trading dates were in the first column of the sbux_df data frame.

To set the names of a vector, you can use names(vector) <- some_names

Remember we are dealing with closing prices.

The first return in sbux_df is thus realized on the second day, or sbux_prices_df[2,1].

```
# we now add dates as names to the vector and print the first elements of sbux_ret to the console to ch
names(sbux_ret) <- sbux_df[2:n,1]
head(sbux_ret, n=7)
```

```
OUTPUT>     4/1/1993    5/3/1993    6/1/1993    7/1/1993    8/2/1993    9/1/1993   10/1/1993
OUTPUT>   0.01769912  0.24347826  0.02097902 -0.03424658  0.02127660  0.13194444 -0.02453988
```

Compute continuously compounded 1-month returns

Recall:

As you might remember from class, the relation between single-period and multi-period returns is multiplicative for single returns.

That is not very convenient.

The yearly return is for example the geometric average of the monthly returns.

Therefore, in practice you will often use continuously compounded returns. T

hese returns have an additive relationship between single and multi-period returns and are defined as:

- r[t] = ln(1 + R[t])

with R[t] the simple return and r[t] the continuously compounded return at moment t.

Continuously compounded returns can be computed easily in R by realizing that

In R, the log price can be easily computed through log(price)

```
# Recall-- we alredy did this
n <- nrow(sbux_prices_df)
sbux_ret <- (sbux_prices_df[2:n, 1] - sbux_prices_df[1:(n - 1), 1]) / sbux_prices_df[1:(n - 1), 1]

# Compute continuously compounded 1-month returns
sbux_ccret <- log(sbux_prices_df[2:n,1]) - log(sbux_prices_df[1:(n - 1),1])

# Assign names to the continuously compounded 1-month returns
names(sbux_ccret) <- sbux_df[2:n,1]

# Show sbux_ccret
head(sbux_ccret)
```

```
OUTPUT>     4/1/1993    5/3/1993    6/1/1993    7/1/1993    8/2/1993    9/1/1993
OUTPUT>   0.01754431  0.21791250  0.02076199 -0.03484673  0.02105341  0.12393690
```

Compare simple and continuously compounded returns

We would like to compare the simple and the continuously compounded returns.

This could be done by generating two graphs.

It would be nice to have the simple and continuously compounded return next to each other in a matrix, with n rows and two columns.

This can be done by using the cbind() function to paste the two vectors that contain both types of returns next to each other in a matrix.

```
# Compare the simple and cc returns
head(cbind(sbux_ret, sbux_ccret, diff=abs(sbux_ret-sbux_ccret)))
```

```
OUTPUT>               sbux_ret  sbux_ccret         diff
OUTPUT>  4/1/1993  0.01769912  0.01754431 0.0001548054
OUTPUT>  5/3/1993  0.24347826  0.21791250 0.0255657590
OUTPUT>  6/1/1993  0.02097902  0.02076199 0.0002170295
OUTPUT>  7/1/1993 -0.03424658 -0.03484673 0.0006001560
OUTPUT>  8/2/1993  0.02127660  0.02105341 0.0002231865
OUTPUT>  9/1/1993  0.13194444  0.12393690 0.0080075432
```

Graphically compare the simple and continuously compounded returns

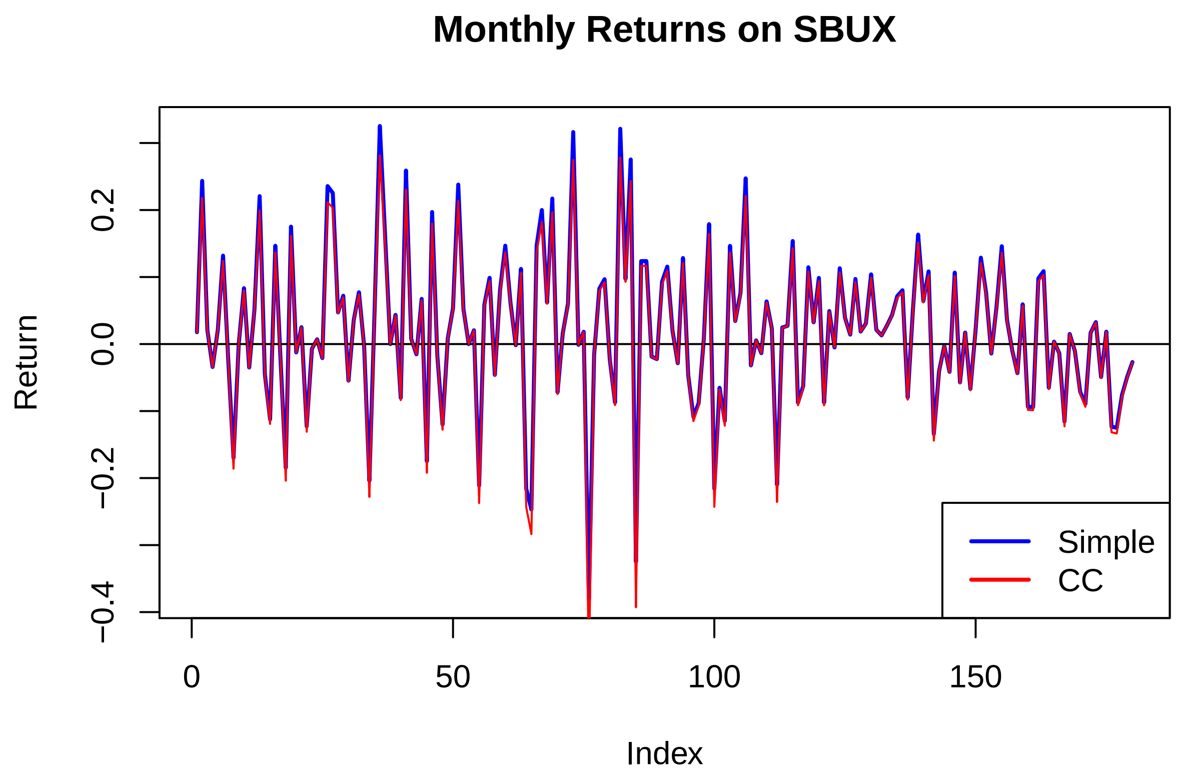
In this section we will create a plot that contains both the simple and continuously compounded returns.

This makes it easy to compare both types of returns visually.

Hints

First of all, we have to plot the simple returns as a function of time. * Use the argument type = l to specify a line plot * col = blue to specify that the simple returns line is blue * lwd = 2 to specify that the line thickness * ylab = "Return" to specify that "Return" is the label of the y-axis * main to specify the plot's main title

```r
# Plot the returns on the same graph
plot(sbux_ret, type = "l", col = "blue", lwd = 2, ylab = "Return", main = "Monthly Returns on SBUX")

# Add horizontal line at zero
abline(h = 0)

# Add a legend
legend(x = "bottomright", legend = c("Simple", "CC"), lty = 1, lwd = 2, col = c("blue", "red"))

# Add the continuously compounded returns
lines(sbux_ccret, col = "red", lwd = 1)
```



Calculate growth of $1 invested in SBUX

Question

Would it have been a good idea to invest in the SBUX stock over the period in our data set?

Specifically: * In case you invested $1 in SBUX on 3/31/1993 (the first day in sbux_df), how much would that dollar be worth on 3/3/2008 (the last day in sbux_df)?

- What was the evolution of the value of that dollar over time?

What to do

R can help you to quickly come up with an answer to these questions.

Hint

Remember that when you use simple returns, the total return over a period can be obtained by taking the cumulative product of the gross returns.

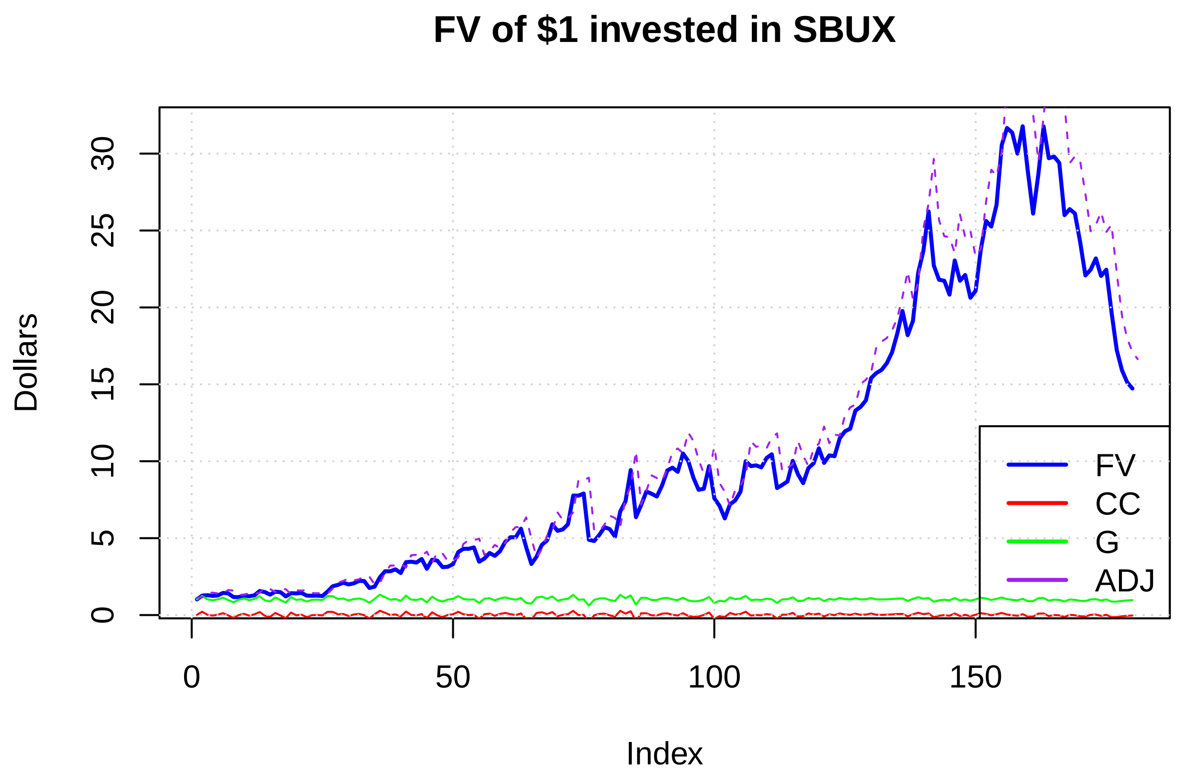
R has a handy cumprod() function that calculates that cumulative product.

```r
# Note:
#The simple returns (sbux_ret) and the continuously compounded returns (sbux_ccret) have been preloaded

# Compute gross returns
sbux_gret <- 1 + sbux_ret

# Compute future values
sbux_fv <- cumprod(sbux_gret)

# see head of data
head(cbind(sbux_gret, sbux_fv))
```

```
OUTPUT>       sbux_gret  sbux_fv
OUTPUT>  [1,] 1.0176991 1.017699
OUTPUT>  [2,] 1.2434783 1.265487
OUTPUT>  [3,] 1.0209790 1.292035
OUTPUT>  [4,] 0.9657534 1.247788
OUTPUT>  [5,] 1.0212766 1.274336
OUTPUT>  [6,] 1.1319444 1.442478
```

```r
# Plot the evolution of the $1 invested in SBUX as a function of time
plot(sbux_fv, type = "l", col = "blue", lwd = 2, ylab = "Dollars", main = "FV of $1 invested in SBUX")
# Add the continuously compounded returns
lines(sbux_ccret, col = "red", lwd = 1)

# Add the gross returns
lines(sbux_gret, col = "green", lwd = 1)

# The adjusted closing
lines(sbux_df$Adj.Close, col = "purple", lty = 2)

# Add a legend
legend(x = "bottomright", legend = c("FV", "CC", "G", "ADJ"), lty = 1, lwd = 2,
       col = c("blue", "red", "green", "purple"))
```

```
grid()
```



Using timeseries approach via the xts and lubridate packages

We will convert the sbux_df dataframe to an xts dataset

```
# first we take a peek at the data
head(sbux_df, n=4)
```

```
OUTPUT>         Date Adj.Close
OUTPUT>  1 3/31/1993      1.13
OUTPUT>  2  4/1/1993      1.15
OUTPUT>  3  5/3/1993      1.43
OUTPUT>  4  6/1/1993      1.46
```

```
#create an xts dataset
library(xts)                    #load the xts library
```

```
OUTPUT>  Loading required package: zoo

OUTPUT>
OUTPUT>  Attaching package: 'zoo'

OUTPUT>  The following objects are masked from 'package:base':
OUTPUT>
OUTPUT>      as.Date, as.Date.numeric
```

```
sbux_df.xts1 = xts( sbux_df[,-1],
                   order.by=as.POSIXct(sbux_df[,1], format = "%m/%d/%Y"))  #, tz = "UTC") )

class( sbux_df.xts1 )
```

```
OUTPUT>  [1] "xts" "zoo"
```

```
str( sbux_df.xts1 )
```

```
OUTPUT>  An 'xts' object on 1993-03-31/2008-03-03 containing:
OUTPUT>    Data: num [1:181, 1] 1.13 1.15 1.43 1.46 1.41 1.44 1.63 1.59 1.32 1.32 ...
OUTPUT>    Indexed by objects of class: [POSIXct,POSIXt] TZ:
OUTPUT>    xts Attributes:
OUTPUT>   NULL
```

```
head( sbux_df.xts1 )
```

```
OUTPUT>             [,1]
OUTPUT>  1993-03-31 1.13
OUTPUT>  1993-04-01 1.15
OUTPUT>  1993-05-03 1.43
OUTPUT>  1993-06-01 1.46
OUTPUT>  1993-07-01 1.41
OUTPUT>  1993-08-02 1.44
```

Another form using lubridate

```
#install.packages("lubridate")
library(lubridate)               #Dates and Times Made Easy with lubridate
```

```
OUTPUT>
OUTPUT>  Attaching package: 'lubridate'

OUTPUT>  The following object is masked from 'package:base':
OUTPUT>
OUTPUT>      date
```

```
sbux_df.xts = xts( sbux_df[,-1], order.by=mdy(sbux_df[,1] ) )

class( sbux_df.xts )
```

```
OUTPUT>  [1] "xts" "zoo"
```

```
str( sbux_df.xts )
```

```
OUTPUT>  An 'xts' object on 1993-03-31/2008-03-03 containing:
OUTPUT>    Data: num [1:181, 1] 1.13 1.15 1.43 1.46 1.41 1.44 1.63 1.59 1.32 1.32 ...
OUTPUT>    Indexed by objects of class: [Date] TZ: UTC
OUTPUT>    xts Attributes:
OUTPUT>   NULL
```

```
head( sbux_df.xts )
```

```
OUTPUT>             [,1]
OUTPUT>  1993-03-31 1.13
OUTPUT>  1993-04-01 1.15
OUTPUT>  1993-05-03 1.43
OUTPUT>  1993-06-01 1.46
OUTPUT>  1993-07-01 1.41
OUTPUT>  1993-08-02 1.44
```

Are the two series identical?

```
head(sbux_df.xts1)
```

```
OUTPUT>              [,1]
OUTPUT>  1993-03-31 1.13
OUTPUT>  1993-04-01 1.15
OUTPUT>  1993-05-03 1.43
OUTPUT>  1993-06-01 1.46
OUTPUT>  1993-07-01 1.41
OUTPUT>  1993-08-02 1.44
```

```
head(sbux_df.xts)
```

```
OUTPUT>              [,1]
OUTPUT>  1993-03-31 1.13
OUTPUT>  1993-04-01 1.15
OUTPUT>  1993-05-03 1.43
OUTPUT>  1993-06-01 1.46
OUTPUT>  1993-07-01 1.41
OUTPUT>  1993-08-02 1.44
```

```
#--
dim(sbux_df.xts1)
```

```
OUTPUT>  [1] 181   1
```

```
dim(sbux_df.xts)
```

```
OUTPUT>  [1] 181   1
```

```
#--
identical( sbux_df.xts1, sbux_df.xts )
```

```
OUTPUT>  [1] FALSE
```

Using window() function

- Find indices associated with the dates 3/1/1994 and 3/1/1995

```
# Find indices associated with the dates 3/1/1994 and 3/1/1995
index_1 <- which( index(sbux_df.xts) == mdy("3/1/1994") )
index_2 <- which( index(sbux_df.xts) == mdy("3/1/1995") )

cat( sprintf("\n => Index1 = %d  and index2 = %d\n", index_1, index_2) )
```

```
OUTPUT>
OUTPUT>   => Index1 = 13  and index2 = 25
```

Extract prices between 3/1/1994 and 3/1/1995

```
window( sbux_df.xts, start=mdy("3/1/1994"), end=mdy("3/1/1995") )
```

```
OUTPUT>              [,1]
OUTPUT>  1994-03-01 1.45
OUTPUT>  1994-04-04 1.77
OUTPUT>  1994-05-02 1.69
OUTPUT>  1994-06-01 1.50
OUTPUT>  1994-07-01 1.72
OUTPUT>  1994-08-01 1.68
OUTPUT>  1994-09-01 1.37
OUTPUT>  1994-10-03 1.61
OUTPUT>  1994-11-01 1.59
OUTPUT>  1994-12-01 1.63
OUTPUT>  1995-01-03 1.43
OUTPUT>  1995-02-01 1.42
OUTPUT>  1995-03-01 1.43
```

```r
# multiple times
dates <- c("3/1/1994", "11/1/1994", "3/1/1995") %>% mdy
class(dates)
```

```
OUTPUT>  [1] "Date"
```

```r
#
sbux_df.xts[dates,]
```

```
OUTPUT>              [,1]
OUTPUT>  1994-03-01 1.45
OUTPUT>  1994-11-01 1.59
OUTPUT>  1995-03-01 1.43
```

```r
#all the dates
head( time(sbux_df.xts), n=3 )
```

```
OUTPUT>  [1] "1993-03-31" "1993-04-01" "1993-05-03"
```

```r
#only data
head( coredata(sbux_df.xts), n=3)
```

```
OUTPUT>        [,1]
OUTPUT>  [1,] 1.13
OUTPUT>  [2,] 1.15
OUTPUT>  [3,] 1.43
```

Dealing with missing data

- Interpolate
- Extrapolate
- Will work on this tomorrow

Aggregating and diasggregating data

- Will work on this tomorrow

Ploting using ggplot

- Will work on this tomorrow

Homework exercises

Use the sbux dataset in your workspace to answe the following question. Submit a copy of your R code justifying your answer

Question 1: Compute one simple Starbucks return

1: What is the simple monthly return between the end of December 2004 and the end of January 2005?

Possible answers

- A: 13.55%
- B: -12.82%
- C: -14.39%
- D: -13.41%
- E: 15.48%

Hint

- Remember that you can access the first element of the sbux vector with sbux[1].
- The simple return is the difference between the first price and the second Starbucks price, divided by the first price.

Question 2: Compute one continuously compounded Starbucks return

2: What is the continuously compounded monthly return between December 2004 and January 2005?

Possible answers

- A: 15.48%
- B: -13.41%
- C: -12.82%
- D: -14.39%
- E: 13.55%

Hint * Do you still remember how you calculated the simple return in the previous exercise?

- The continuously compounded return is just the natural logarithm of the simple return plus one.

Question 3: Monthly compounding

3: Assume that all twelve months have the same return as the simple monthly return between the end of December 2004 and the end of January 2005. What would be the annual return with monthly compounding in that case?

Possible answers

- A: 172.73%
- B: -160.92%
- C: -82.22%
- D: -80.72%

- E: -84.50%

Hint * In the first exercise you calculated the simple return between December 2004 and January 2005.

- Have a look a the wikipedia article on compound interest and think about how that applies to this situation.

Question 4: Simple annual Starbucks return

4: Use the data in sbux and compute the actual simple annual return between December 2004 and December 2005.

Your workspace still contains the vector sbux with the adjusted closing price data for Starbucks stock over the period December 2004 through December 2005.

Possible answers

- A: -2.15%
- B: -8.44%
- C: -12.34%
- D: -2.17%
- E: -6.20%

Hint * Use sbux[1] to extract the first price and sbux[length(sbux)] to extract the last price.

*To get the simple annual return, calculate the price difference and divide by the initial price.

Question 5: Annual continuously compounded return

5: Use the data sbux and compute the actual annual continuously compounded return between December 2004 and December 2005.

Possible answers

- A: 6.20%
- B: -2.17%
- C: -12.34%
- D: -2.15%
- E: 8.44%

Hint * Do you still remember how you calculated the annual Starbucks return in the previous exercise? * Well, the continuously compounded annual return is just the natural logarithm of that return plus one.

Solutions for Instructors only –

This will be deleted later

1: D: -13.41% ((sbux[2] - sbux[1]) / sbux[1]) 2: D -14.39% (natural logarithm of the simple return plus one) 3: C -82.22% 4: A -2.15% 5: B: -2.17% (natural logarithm of that return plus one)