AIML for Finance - Using Quantmod to Import Financial Data

Brian Clark

Fall 2020

Contents

---

1 Overview

This document is broken down into three parts. The first describes how to use R's "quantmod" package to download stock data. This part is based on a vignette put together and maintained by Majeed Simaan who is a Lally School (RPI) PhD and is now an Assisant Professor of Finance at Stevens Institute of Technology. A link to his github page with the details of his vignette is **here**. The second part includes examples of how to bring in other relevant financial data. The third part shows some useful time-series commands.

2 Downloading Stock Data into R using "quantmod""

This section is based on Majeed Simaan's R vignette. Please refer to his original file as his vignette also contains instructions and examples of how to download and manage options data along with macroeconomic data hosted by the **Federal Reserve Bank of St. Louis**.

Prior to starting, you need to install the "quantmod" package. For this example, we download daily equity data for six stocks starting on January 1, 1980.

```
rm(list=ls())

# install.packages("quantmod")
library("quantmod")

# Define some companies:
tics <- c("AAPL","GE","TWTR","BAC","RAD","PFE")
P.list <- lapply(tics, function(tic)
  get(getSymbols(tic, from = "1980-01-01")))

sapply(P.list,nrow)
```

```
## [1]  9862 10102  1562 10102 10050 10102
```

A few comments are in order for the above block of code. The tics object obviouly constains the ticker symbols you want to use. They correspond to the tickers on Yahoo Finance so if you want other companies you can search for the tickers **here**. We will stick to Majeed's example for now.

The "lapply" function is another way to write a for loop in R. For example, P.list could also be defined (equivalently) as follows:

```
# The following block is equivalent to the lapply command::
P.list2 <- list(NA)
n <- length(tics)
for (i in 1:n){
  P.list2[[i]] <- get(getSymbols(tics[i], from = "1980-01-01"))
}

# ...is equivalent to...

# P.list <- lapply(tics, function(tic)
#   get(getSymbols(tic, from = "1980-01-01")))

# Printing the number of rows could also be accomplished in a loop:
for (i in 1:n){
  print(nrow(P.list2[[i]]))
}
```

```
## [1] 9862
## [1] 10102
```

```
## [1] 1562
## [1] 10102
## [1] 10050
## [1] 10102
```

```
sapply(P.list,nrow)
```

```
## [1]  9862 10102  1562 10102 10050 10102
```

Note that each of the stocks has a different number of rows. This is because we are using real data and as such there will be missing values (for example, Twitter was not around in 1980!). Let's print the first and last few observations for each stock:

```
for (i in 1:n){
  print(tics[i])
  print(head(P.list[[i]],4))
  print("----------------------------------------")
}
```

```
## [1] "AAPL"
##            AAPL.Open AAPL.High AAPL.Low AAPL.Close AAPL.Volume
## 1980-12-12  0.513393  0.515625 0.513393   0.513393   117258400
## 1980-12-15  0.488839  0.488839 0.486607   0.486607    43971200
## 1980-12-16  0.453125  0.453125 0.450893   0.450893    26432000
## 1980-12-17  0.462054  0.464286 0.462054   0.462054    21610400
##            AAPL.Adjusted
## 1980-12-12      0.407747
## 1980-12-15      0.386473
## 1980-12-16      0.358108
## 1980-12-17      0.366972
## [1] "----------------------------------------"
## [1] "GE"
##             GE.Open  GE.High   GE.Low GE.Close GE.Volume GE.Adjusted
## 1980-01-02 1.014123 1.016627 0.976563 0.976563   7433000    0.004071
## 1980-01-03 0.976563 0.991587 0.959034 0.989083   9185200    0.004124
## 1980-01-04 0.999099 1.024139 0.999099 1.021635   8556200    0.004259
## 1980-01-07 1.021635 1.071715 1.014123 1.056691  10518100    0.004405
## [1] "----------------------------------------"
## [1] "TWTR"
##            TWTR.Open TWTR.High TWTR.Low TWTR.Close TWTR.Volume
## 2013-11-07     45.10     50.09    44.00      44.90   117701600
## 2013-11-08     45.93     46.94    40.69      41.65    27925300
## 2013-11-11     40.50     43.00    39.40      42.90    16113900
## 2013-11-12     43.66     43.78    41.83      41.90     6316700
##            TWTR.Adjusted
## 2013-11-07         44.90
## 2013-11-08         41.65
## 2013-11-11         42.90
## 2013-11-12         41.90
## [1] "----------------------------------------"
## [1] "BAC"
##            BAC.Open BAC.High  BAC.Low BAC.Close BAC.Volume BAC.Adjusted
## 1980-01-02        0 1.718750 1.687500  1.687500      36000     0.130777
## 1980-01-03        0 1.687500 1.671875  1.671875      31200     0.129566
## 1980-01-04        0 1.671875 1.640625  1.671875      35200     0.129566
## 1980-01-07        0 1.703125 1.656250  1.687500     260000     0.130777
```

```
## [1] "----------------------------------------"
## [1] "RAD"
##            RAD.Open RAD.High  RAD.Low RAD.Close RAD.Volume RAD.Adjusted
## 1980-03-17        0 28.28125 27.03125   27.1875      11400     19.13496
## 1980-03-18        0 26.87500 26.25000   26.5625      27400     18.69507
## 1980-03-19        0 26.87500 25.93750   25.9375      12000     18.25519
## 1980-03-20        0 26.25000 25.62500   25.6250       2900     18.03524
## [1] "----------------------------------------"
## [1] "PFE"
##            PFE.Open PFE.High  PFE.Low PFE.Close PFE.Volume PFE.Adjusted
## 1980-01-02 0.809896 0.809896 0.781250  0.781250    3216000     0.000681
## 1980-01-03 0.781250 0.789063 0.770833  0.781250    2846400     0.000681
## 1980-01-04 0.789063 0.809896 0.789063  0.809896    3316800     0.000706
## 1980-01-07 0.809896 0.820313 0.799479  0.809896    2184000     0.000706
## [1] "----------------------------------------"
```

```
for (i in 1:n){
  print(tics[i])
  print(tail(P.list[[i]],4))
  print("----------------------------------------")
}
```

```
## [1] "AAPL"
##            AAPL.Open AAPL.High AAPL.Low AAPL.Close AAPL.Volume
## 2020-01-17    316.27    318.74   315.00     318.73    34454100
## 2020-01-21    317.19    319.02   316.00     316.57    27710800
## 2020-01-22    318.58    319.99   317.31     317.70    25458100
## 2020-01-23    317.92    319.56   315.65     319.23    26071600
##            AAPL.Adjusted
## 2020-01-17        318.73
## 2020-01-21        316.57
## 2020-01-22        317.70
## 2020-01-23        319.23
## [1] "----------------------------------------"
## [1] "GE"
##            GE.Open GE.High GE.Low GE.Close GE.Volume GE.Adjusted
## 2020-01-17   11.85   11.93  11.76    11.81  46580700       11.81
## 2020-01-21   11.74   11.80  11.61    11.66  43738200       11.66
## 2020-01-22   11.71   11.71  11.35    11.37  61947800       11.37
## 2020-01-23   11.73   11.85  11.60    11.77  73125200       11.77
## [1] "----------------------------------------"
## [1] "TWTR"
##            TWTR.Open TWTR.High TWTR.Low TWTR.Close TWTR.Volume
## 2020-01-17     33.82     34.39    33.58      34.22    19303500
## 2020-01-21     34.08     34.39    33.87      34.09    17055600
## 2020-01-22     34.29     34.54    33.96      34.02    12851800
## 2020-01-23     33.99     34.36    33.72      33.89    14879100
##            TWTR.Adjusted
## 2020-01-17         34.22
## 2020-01-21         34.09
## 2020-01-22         34.02
## 2020-01-23         33.89
## [1] "----------------------------------------"
## [1] "BAC"
##            BAC.Open BAC.High BAC.Low BAC.Close BAC.Volume BAC.Adjusted
```

```
## 2020-01-17    34.92    34.95   34.60     34.71   54156100        34.71
## 2020-01-21    34.42    34.52   34.22     34.26   50811000        34.26
## 2020-01-22    34.37    34.45   34.23     34.36   38879700        34.36
## 2020-01-23    34.09    34.26   33.73     34.12   45696200        34.12
## [1] "----------------------------------------"
## [1] "RAD"
##            RAD.Open RAD.High RAD.Low RAD.Close RAD.Volume RAD.Adjusted
## 2020-01-17    12.96    13.48   12.45     12.60    4672500        12.60
## 2020-01-21    12.49    13.45   12.35     12.98    4854900        12.98
## 2020-01-22    12.97    13.28   12.72     12.86    2808500        12.86
## 2020-01-23    12.68    12.92   12.36     12.58    2453400        12.58
## [1] "----------------------------------------"
## [1] "PFE"
##            PFE.Open PFE.High PFE.Low PFE.Close PFE.Volume PFE.Adjusted
## 2020-01-17    40.57    40.81   40.42     40.51   21901300        40.51
## 2020-01-21    40.38    40.66   40.19     40.34   21931400        40.34
## 2020-01-22    40.32    40.41   39.97     40.19   17170600        40.19
## 2020-01-23    40.19    40.83   40.13     40.71   25757600        40.71
## [1] "----------------------------------------"
```

While all of our stocks have different starting dates, they all have the same end date which is the most recent closing date (it will change depending on when you run the code).

Not all data is relevant for our puposes. What we are really interested in are the adjusted prices which are given in the last column of each list object. The adjusted prices are adjusted for stock splits and dividends. The idea is that the adjusted prices represent the true return on an investment in a given stock.

Therefore, the next step is to get all the adjusted prices into a single object:

```
# Get the adjusted prices into a single object
P.adj <- lapply(P.list, function(p) p[,6])

# Merge the elements of the list
P <- Reduce(merge,P.adj)
names(P) <- tics

head(P,10)
```

```
##             AAPL       GE TWTR       BAC RAD       PFE
## 1980-01-02   NA 0.004071   NA 0.130777  NA 0.000681
## 1980-01-03   NA 0.004124   NA 0.129566  NA 0.000681
## 1980-01-04   NA 0.004259   NA 0.129566  NA 0.000706
## 1980-01-07   NA 0.004405   NA 0.130777  NA 0.000706
## 1980-01-08   NA 0.004562   NA 0.129566  NA 0.000740
## 1980-01-09   NA 0.004510   NA 0.135620  NA 0.000724
## 1980-01-10   NA 0.004531   NA 0.133198  NA 0.000715
## 1980-01-11   NA 0.004520   NA 0.133198  NA 0.000724
## 1980-01-14   NA 0.004520   NA 0.133198  NA 0.000724
## 1980-01-15   NA 0.004416   NA 0.134409  NA 0.000706
```

```
tail(P,10)
```

```
##              AAPL    GE  TWTR   BAC   RAD   PFE
## 2020-01-09 309.63 11.91 33.22 35.03 12.69 38.89
## 2020-01-10 310.33 11.67 32.78 34.74 12.04 39.49
## 2020-01-13 316.96 12.12 32.69 35.06 11.66 39.41
## 2020-01-14 312.68 12.03 32.82 35.32 12.28 40.07
```

```
## 2020-01-15 311.34 11.87 33.23 34.67 12.83 40.67
## 2020-01-16 315.24 11.84 34.19 34.72 13.00 40.61
## 2020-01-17 318.73 11.81 34.22 34.71 12.60 40.51
## 2020-01-21 316.57 11.66 34.09 34.26 12.98 40.34
## 2020-01-22 317.70 11.37 34.02 34.36 12.86 40.19
## 2020-01-23 319.23 11.77 33.89 34.12 12.58 40.71
```

Again, the P.adj "lapply" funciton could be written equivalently in a loop as follows:

```
# Get the adjusted prices into a single object
P.adj2 <- list(NA)
for (i in 1:n){
  P.adj2[[i]] <- P.list[[i]][,6]
}

P2 <- Reduce(merge,P.adj2)
names(P2) <- tics

head(P2,10)
```

```
##             AAPL       GE TWTR      BAC RAD      PFE
## 1980-01-02   NA 0.004071   NA 0.130777  NA 0.000681
## 1980-01-03   NA 0.004124   NA 0.129566  NA 0.000681
## 1980-01-04   NA 0.004259   NA 0.129566  NA 0.000706
## 1980-01-07   NA 0.004405   NA 0.130777  NA 0.000706
## 1980-01-08   NA 0.004562   NA 0.129566  NA 0.000740
## 1980-01-09   NA 0.004510   NA 0.135620  NA 0.000724
## 1980-01-10   NA 0.004531   NA 0.133198  NA 0.000715
## 1980-01-11   NA 0.004520   NA 0.133198  NA 0.000724
## 1980-01-14   NA 0.004520   NA 0.133198  NA 0.000724
## 1980-01-15   NA 0.004416   NA 0.134409  NA 0.000706
```

```
tail(P2,10)
```

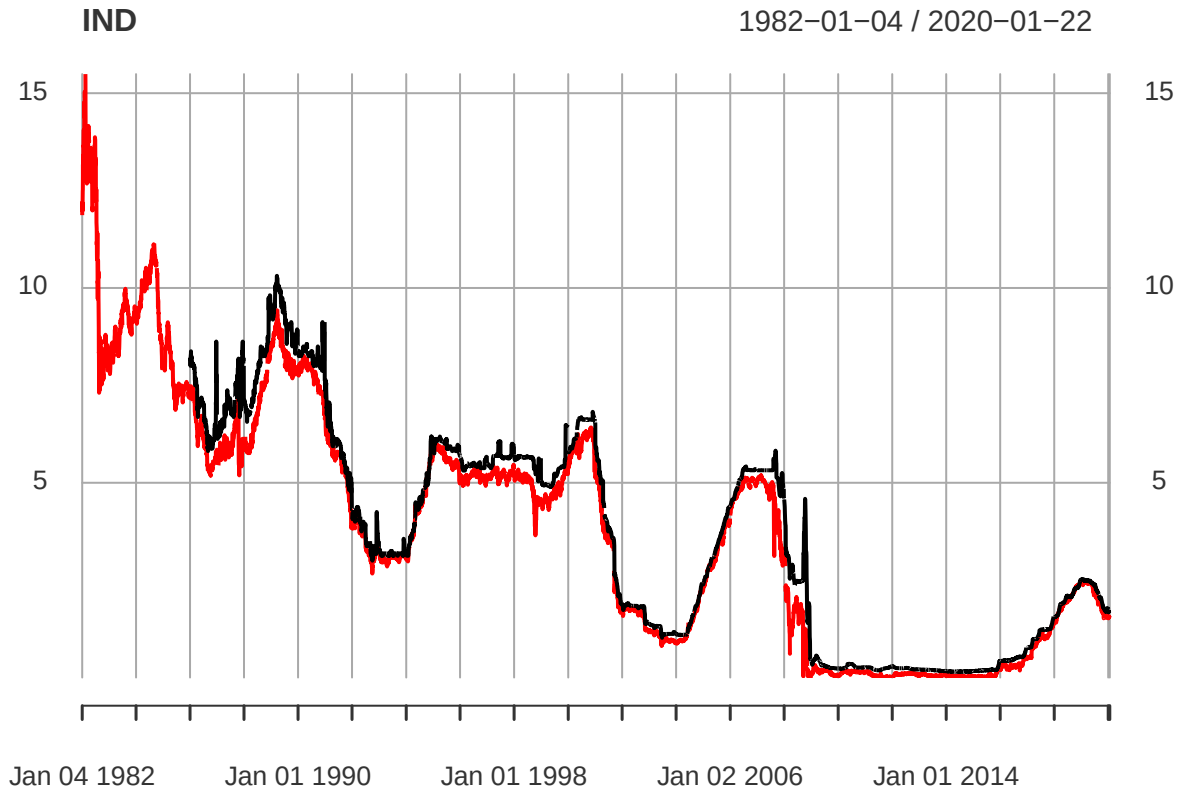
```
##              AAPL    GE  TWTR   BAC   RAD   PFE
## 2020-01-09 309.63 11.91 33.22 35.03 12.69 38.89
## 2020-01-10 310.33 11.67 32.78 34.74 12.04 39.49
## 2020-01-13 316.96 12.12 32.69 35.06 11.66 39.41
## 2020-01-14 312.68 12.03 32.82 35.32 12.28 40.07
## 2020-01-15 311.34 11.87 33.23 34.67 12.83 40.67
## 2020-01-16 315.24 11.84 34.19 34.72 13.00 40.61
## 2020-01-17 318.73 11.81 34.22 34.71 12.60 40.51
## 2020-01-21 316.57 11.66 34.09 34.26 12.98 40.34
## 2020-01-22 317.70 11.37 34.02 34.36 12.86 40.19
## 2020-01-23 319.23 11.77 33.89 34.12 12.58 40.71
```

Now we have a data set of equity prices which contains time series observations in along the rows and the stocks across the columns. In the next section, we will prepare the data to form a set of efficient portfolios.

3 Macro Data

This section gives an example of how to get some interest rate or other macroeconomic data. The example we will use is the FRED database which is maintained by the Federal Reserve Bank of St. Louis and is available **here**. Simply search for the data you want and the data code will be returned with the search.

```
#-----------------------------------------------
#         PART 3: Getting interest rate data:
indices <- c("USD1MTD156N","DGS3MO")
ind.list <- lapply(indices, function(tic) get(getSymbols(tic, from = "1980-01-01", src = "FRED")) )
IND <- Reduce(merge,ind.list)
plot(IND)
```




```
write.csv(data.frame(IND),file = "IND.csv")
```

4 Time-Series Data in R

This section provides some useful commands for handling time-series data in R. Note that there are other ways to handle time-series data.

Let's start by importing daily S&P 500 returns using the `quantmod` package. We'll start from January 1, 1990. We will use the 'SPDR S&P 500 ETF Trust ETF to proxy the actual index.

```
## ----warning=FALSE,message=FALSE-------------------------------------
library(quantmod)
library(lubridate)
```

```
##
## Attaching package: 'lubridate'
```

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

```
library(plyr)
```

```
##
## Attaching package: 'plyr'
## The following object is masked from 'package:lubridate':
##
##     here
```

```
library(glmnet)
```

```
## Loading required package: Matrix
## Loading required package: foreach
## Loaded glmnet 2.0-18
```

```
library(PerformanceAnalytics)
```

```
##
## Attaching package: 'PerformanceAnalytics'
## The following object is masked from 'package:graphics':
##
##     legend
```

```
library(ggplot2)
library(plotly)
```

```
##
## Attaching package: 'plotly'
## The following object is masked from 'package:ggplot2':
##
##     last_plot
## The following objects are masked from 'package:plyr':
##
##     arrange, mutate, rename, summarise
## The following object is masked from 'package:stats':
##
##     filter
## The following object is masked from 'package:graphics':
##
##     layout
```

```
library(parallel)

rm(list = ls())

t1 <- "1990-01-01"
v <- c("SPY")
P.list <- lapply(v, function(sym) get(getSymbols(sym,from = t1)) )
```

Summarize the data.

```
head(P.list[[1]],10)
```

```
##            SPY.Open SPY.High  SPY.Low SPY.Close SPY.Volume SPY.Adjusted
```

```
## 1993-01-29 43.96875 43.96875 43.75000  43.93750    1003200     26.45393
## 1993-02-01 43.96875 44.25000 43.96875  44.25000     480500     26.64206
## 1993-02-02 44.21875 44.37500 44.12500  44.34375     201300     26.69851
## 1993-02-03 44.40625 44.84375 44.37500  44.81250     529400     26.98074
## 1993-02-04 44.96875 45.09375 44.46875  45.00000     531500     27.09362
## 1993-02-05 44.96875 45.06250 44.71875  44.96875     492100     27.07482
## 1993-02-08 44.96875 45.12500 44.90625  44.96875     596100     27.07482
## 1993-02-09 44.81250 44.81250 44.56250  44.65625     122100     26.88667
## 1993-02-10 44.65625 44.75000 44.53125  44.71875     379600     26.92429
## 1993-02-11 44.78125 45.12500 44.78125  44.93750      19500     27.05599
```

```
## ------------------------------------------------------------------------
sapply(P.list,dim)
```

```
##      [,1]
## [1,] 6795
## [2,]    6
```

```
## ------------------------------------------------------------------------
lapply(P.list, function(x)  first(date(x)) )
```

```
## [[1]]
## [1] "1993-01-29"
```

```
## ------------------------------------------------------------------------
P.list5 <- lapply(P.list, function(x) x[,5])
P.list6 <- lapply(P.list, function(x) x[,6])

## ------------------------------------------------------------------------
P5 <- na.omit(Reduce(function(...) merge(...),P.list5 ))
P6 <- na.omit(Reduce(function(...) merge(...),P.list6 ))

head(P5, 5)
```

```
##            SPY.Volume
## 1993-01-29    1003200
## 1993-02-01     480500
## 1993-02-02     201300
## 1993-02-03     529400
## 1993-02-04     531500
```

```
head(P6, 5)
```

```
##            SPY.Adjusted
## 1993-01-29     26.45393
## 1993-02-01     26.64206
## 1993-02-02     26.69851
## 1993-02-03     26.98074
## 1993-02-04     27.09362
```

$P.5$ and $P.6$ are `xts zoo` objects containing the daily volume and adjusted price of SPY.

```
# adjust names
names(P5) <- names(P6) <- c("SPY")
names(P5) <- paste(names(P5),"vol",sep = "_")
head(P5,5)
```

```
##            SPY_vol
## 1993-01-29 1003200
```

```
## 1993-02-01  480500
## 1993-02-02  201300
## 1993-02-03  529400
## 1993-02-04  531500
```

```
head(P6,5)
```

```
##                 SPY
## 1993-01-29 26.45393
## 1993-02-01 26.64206
## 1993-02-02 26.69851
## 1993-02-03 26.98074
## 1993-02-04 27.09362
```

Next, let's compute the returns to forecast.

```
## ------------------------------------------------------------------------
R6 <- Return.calculate(P6)
head(R6, 5)
```

```
##                    SPY
## 1993-01-29          NA
## 1993-02-01 0.007111495
## 1993-02-02 0.002118830
## 1993-02-03 0.010571190
## 1993-02-04 0.004183799
```

Let's say we wanted a rolling average of the returns to implement a momentum strategy. We can use the rollapply() function.

```
# add 5-day rolling average:
n.days <- 5
R6_roll <- rollapply(R6,n.days,mean)
names(R6_roll) <- paste(names(R6_roll),"_roll_",n.days,sep="")

R <- na.omit(merge(R6,R6_roll,P5))

SPY_lag <- stats::lag(R$SPY,1)
names(SPY_lag) <- "SPY_lag"
R <- na.omit(merge(SPY_lag,R))

## ------------------------------------------------------------------------
range(date(R))
```

```
## [1] "1993-02-08" "2020-01-23"
```

```
## ------------------------------------------------------------------------
cor(R)[,"SPY"]
```

```
##     SPY_lag         SPY  SPY_roll_5     SPY_vol
## -0.06156836  1.00000000  0.42700574 -0.07683575
```

```
## ------------------------------------------------------------------------
cor(R)[,"SPY_lag"]
```

```
##     SPY_lag         SPY  SPY_roll_5     SPY_vol
##  1.00000000 -0.06156836  0.40392257 -0.10460493
```

```
# stack into a dataset rather than an xts object
ds <- data.frame(date = date(R),R)
```

```
rownames(ds) <- NULL

library("kableExtra")
```

```
## Warning: package 'kableExtra' was built under R version 3.6.2
```

```
kable(ds[c(1:20),], digits=4)
```

date	SPY_lag	SPY	SPY_roll_5	SPY_vol
1993-02-08	-0.0007	0.0000	0.0032	596100
1993-02-09	0.0000	-0.0069	0.0014	122100
1993-02-10	-0.0069	0.0014	-0.0004	379600
1993-02-11	0.0014	0.0049	-0.0003	19500
1993-02-12	0.0049	-0.0076	-0.0017	42500
1993-02-16	-0.0076	-0.0252	-0.0067	374800
1993-02-17	-0.0252	-0.0007	-0.0055	210900
1993-02-18	-0.0007	-0.0007	-0.0059	378100
1993-02-19	-0.0007	0.0036	-0.0061	34900
1993-02-22	0.0036	0.0036	-0.0039	513600
1993-02-23	0.0036	-0.0007	0.0010	373700
1993-02-24	-0.0007	0.0129	0.0037	26300
1993-02-25	0.0129	0.0021	0.0043	44500
1993-02-26	0.0021	0.0014	0.0039	66200
1993-03-01	0.0014	-0.0028	0.0026	66500
1993-03-02	-0.0028	0.0148	0.0057	182400
1993-03-03	0.0148	0.0042	0.0039	280100
1993-03-04	0.0042	-0.0055	0.0024	89500
1993-03-05	-0.0055	-0.0028	0.0016	40000
1993-03-08	-0.0028	0.0223	0.0066	50800

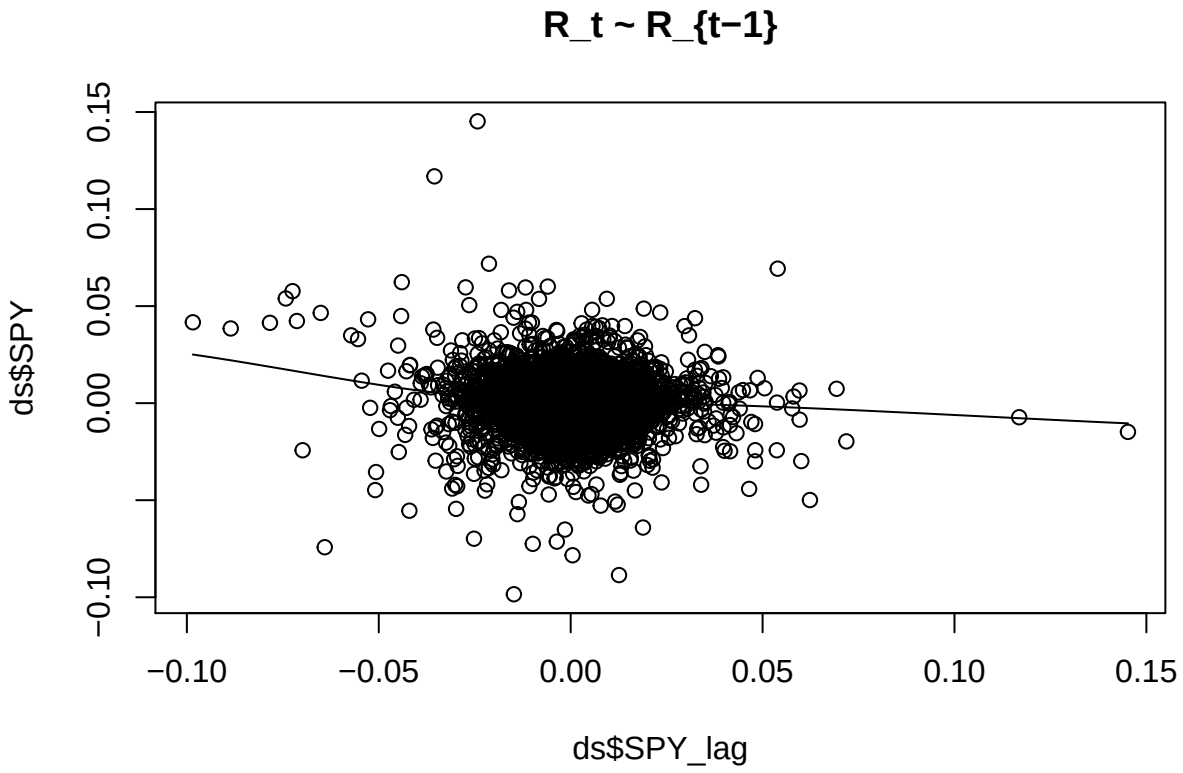
Now we have a dataset that we could use to predict returns, for example. If we wanted to extract a vector of the weeks of the data where each date corresponds to a Sunday, we could do the following. The same could be done for months.

```
## ----warning=FALSE-------------------------------------------------------
weeks <- date(unique(floor_date(ds$date,"week")))
weeks <- c(weeks, last(weeks) + weeks(1))

months <- date(unique(floor_date(ds$date,"month")))
months <- c(months, last(months) + months(1))
```

As a simple example, we could run a linear regression of the returns on the lagged returns. First, visualze a scatterplot. Note that we don't expect to see much here since returns are hard to predict!

```
scatter.smooth(x=ds$SPY_lag, y=ds$SPY, main="R_t ~ R_{t-1}")  # scatterplot
```



Now run the model and output a summary of the results.

```
sp.mod <- lm(SPY ~ SPY_lag,data=ds)
print(sp.mod)
```

```
##
## Call:
## lm(formula = SPY ~ SPY_lag, data = ds)
##
## Coefficients:
## (Intercept)      SPY_lag
##    0.000461    -0.061568
```

```
summary(sp.mod)
```

```
##
## Call:
## lm(formula = SPY ~ SPY_lag, data = ds)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.099820 -0.004687  0.000318  0.005284  0.143243
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.0004610  0.0001384   3.331 0.000869 ***
## SPY_lag     -0.0615683  0.0121154  -5.082 3.84e-07 ***
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.01139 on 6787 degrees of freedom
## Multiple R-squared:  0.003791,   Adjusted R-squared:  0.003644
## F-statistic: 25.83 on 1 and 6787 DF,  p-value: 3.838e-07
```

While the lagged returns are statistically significant, they explain virtually none of the errow as the $adj - R^2 = 0.0036$ is tiny.

5 References