Rishikesh\_Yadav\_HW1

Rishikesh Yadav

02-14-2023

# Part I

## 1.1 Vector

1. Create 2 vector, each containing 10 random numbers.

v1 <- sample(0:100, 10, replace=TRUE)  
v2 <- sample(0:100, 10, replace=TRUE)  
is.vector(v1)

## [1] TRUE

is.vector(v2)

## [1] TRUE

1. Appending Vector 2 to Vector 1.

appended.vector <- append(v1,v2)  
appended.vector

## [1] 65 6 77 84 57 60 69 86 1 62 49 52 65 31 55 77 37 13 51 92

1. Calculate the mean of the new combined vector.

mean.vector <- mean(appended.vector)  
mean.vector

## [1] 54.45

1. If element is lager than the mean, print ’True’, else print ’False’.

for (x in appended.vector) {  
 if (x > mean.vector) {  
 print(TRUE)  
 } else {  
 print(FALSE)  
 }  
}

## [1] TRUE  
## [1] FALSE  
## [1] TRUE  
## [1] TRUE  
## [1] TRUE  
## [1] TRUE  
## [1] TRUE  
## [1] TRUE  
## [1] FALSE  
## [1] TRUE  
## [1] FALSE  
## [1] FALSE  
## [1] TRUE  
## [1] FALSE  
## [1] TRUE  
## [1] TRUE  
## [1] FALSE  
## [1] FALSE  
## [1] FALSE  
## [1] TRUE

## 1.2 Matrix

1. Create a vector with 100 random numbers.

vector <- round(runif(n = 100, min = 1, max = 100), 0)

1. Transfer the above vector into a 10 by 10 matrix M.

matrix <- matrix(vector, nrow = 10)  
matrix

## [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]  
## [1,] 78 68 57 21 91 98 64 54 84 71  
## [2,] 91 40 57 83 5 80 19 88 3 72  
## [3,] 15 78 76 74 12 81 83 47 73 30  
## [4,] 98 81 33 60 24 78 99 35 95 35  
## [5,] 13 96 16 37 67 70 8 52 42 45  
## [6,] 7 76 59 63 62 88 93 2 33 80  
## [7,] 30 65 53 24 61 81 76 99 7 34  
## [8,] 85 62 71 94 92 61 88 83 50 7  
## [9,] 70 45 24 72 22 1 46 71 38 58  
## [10,] 22 34 36 14 9 59 24 33 40 61

1. Find the transposed matrix

transposed.matrix <- t(matrix)  
print(transposed.matrix)

## [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]  
## [1,] 78 91 15 98 13 7 30 85 70 22  
## [2,] 68 40 78 81 96 76 65 62 45 34  
## [3,] 57 57 76 33 16 59 53 71 24 36  
## [4,] 21 83 74 60 37 63 24 94 72 14  
## [5,] 91 5 12 24 67 62 61 92 22 9  
## [6,] 98 80 81 78 70 88 81 61 1 59  
## [7,] 64 19 83 99 8 93 76 88 46 24  
## [8,] 54 88 47 35 52 2 99 83 71 33  
## [9,] 84 3 73 95 42 33 7 50 38 40  
## [10,] 71 72 30 35 45 80 34 7 58 61

print(transposed.matrix[2,1])

## [1] 68

1. Nested loop to calculate the inner product between M.T and M.

InnerProduct <- function(a, b){  
 if(ncol(a) != nrow(b)){  
 return("can't multiply")  
 }  
 else{  
 c = matrix(rep(0, nrow(a) \* ncol(b)), nrow = nrow(a))  
 for(i in 1:nrow(a)){  
 for(j in 1:ncol(b)){  
 for(k in 1:nrow(b)){  
 c[i,j] <- c[i,j] + a[i,k] \* b[k, j]  
 }  
 }  
 }  
 }  
 return(c)  
}  
  
matrix.product <- InnerProduct(transposed.matrix, matrix)  
print(matrix.product)

## [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]  
## [1,] 37921 30950 24725 31161 22778 34292 31931 32766 26007 24132  
## [2,] 30950 45111 30928 34824 31377 47006 40723 34735 32386 30611  
## [3,] 24725 30928 26822 28019 22523 35960 32165 27803 22534 22913  
## [4,] 31161 34824 28019 36536 22861 36082 34902 31818 24912 24996  
## [5,] 22778 31377 22523 22861 30109 33414 29553 25900 21898 20539  
## [6,] 34292 47006 35960 36082 33414 55057 43967 37785 33654 34906  
## [7,] 31931 40723 32165 34902 29553 43967 46072 31982 31942 26999  
## [8,] 32766 34735 27803 31818 25900 37785 31982 39622 22667 25383  
## [9,] 26007 32386 22534 24912 21898 33654 31942 22667 29865 21457  
## [10,] 24132 30611 22913 24996 20539 34906 26999 25383 21457 29065

1. Calculate the same inner product using operator %∗%.

matrix.operator <- transposed.matrix %\*% matrix  
print(matrix.operator)

## [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]  
## [1,] 37921 30950 24725 31161 22778 34292 31931 32766 26007 24132  
## [2,] 30950 45111 30928 34824 31377 47006 40723 34735 32386 30611  
## [3,] 24725 30928 26822 28019 22523 35960 32165 27803 22534 22913  
## [4,] 31161 34824 28019 36536 22861 36082 34902 31818 24912 24996  
## [5,] 22778 31377 22523 22861 30109 33414 29553 25900 21898 20539  
## [6,] 34292 47006 35960 36082 33414 55057 43967 37785 33654 34906  
## [7,] 31931 40723 32165 34902 29553 43967 46072 31982 31942 26999  
## [8,] 32766 34735 27803 31818 25900 37785 31982 39622 22667 25383  
## [9,] 26007 32386 22534 24912 21898 33654 31942 22667 29865 21457  
## [10,] 24132 30611 22913 24996 20539 34906 26999 25383 21457 29065

## 1.3 Function

1. Load the given CSV file

df <- read.csv("stock\_data-1.csv", head = TRUE)  
df$X <- as.Date(df$X, format = "%Y-%m-%d")  
head(df)

## X AAPL AMGN AXP BA CAT CRM CSCO CVX  
## 1 1996-01-02 0.286830 14.56250 12.10832 39.93750 14.87500 NA 4.243055 26.43750  
## 2 1996-01-03 0.286830 14.40625 12.10832 39.56250 15.12500 NA 4.076389 26.50000  
## 3 1996-01-04 0.281808 13.78125 11.99890 38.56250 15.00000 NA 3.923611 27.25000  
## 4 1996-01-05 0.305804 14.09375 11.96243 39.25000 15.25000 NA 3.972222 27.68750  
## 5 1996-01-08 0.309152 13.85938 11.96243 40.12500 15.18750 NA 3.934028 27.81250  
## 6 1996-01-09 0.292411 13.53125 11.78008 39.67969 14.78125 NA 3.631944 27.92188  
## DIS DOW GS HD IBM INTC JNJ JPM KO  
## 1 20.01773 NA NA 10.527778 22.71875 7.328125 21.06250 19.58333 18.75000  
## 2 20.14104 NA NA 10.333333 22.31250 7.218750 21.90625 19.58333 18.90625  
## 3 19.89442 NA NA 10.305555 21.71875 7.187500 21.68750 18.75000 18.75000  
## 4 20.26435 NA NA 10.055555 22.15625 7.187500 21.68750 18.66667 18.65625  
## 5 20.38767 NA NA 9.777778 22.28125 7.203125 21.96875 18.66667 18.78125  
## 6 20.41850 NA NA 9.666667 21.68750 6.875000 22.09375 18.20833 18.53125  
## MCD MMM MRK MSFT NKE PG TRV UNH V  
## 1 22.7500 33.87500 32.1250 5.609375 4.445313 20.78125 28.2500 8.078125 NA  
## 2 22.7500 33.81250 31.6875 5.429688 4.312500 21.40625 28.6250 8.109375 NA  
## 3 22.8750 33.68750 31.8750 5.460938 4.265625 21.75000 29.0000 8.187500 NA  
## 4 22.5000 33.75000 31.5000 5.398438 4.132813 21.84375 29.0625 7.859375 NA  
## 5 22.5625 33.50000 31.9375 5.390625 4.203125 21.93750 29.1875 7.703125 NA  
## 6 22.1875 33.01562 31.7500 5.011719 4.117188 21.90625 28.9375 7.265625 NA  
## VZ WBA WMT  
## 1 30.46456 7.53125 11.6250  
## 2 31.42009 7.50000 11.7500  
## 3 30.85801 7.40625 11.8750  
## 4 31.19526 7.68750 11.6875  
## 5 30.97043 7.62500 11.6875  
## 6 30.93530 7.56250 11.5000

1. Delete the columns containing NA(empty values).

df<- df[ , colSums(is.na(df))==0]  
head(df)

## X AAPL AMGN AXP BA CAT CSCO CVX  
## 1 1996-01-02 0.286830 14.56250 12.10832 39.93750 14.87500 4.243055 26.43750  
## 2 1996-01-03 0.286830 14.40625 12.10832 39.56250 15.12500 4.076389 26.50000  
## 3 1996-01-04 0.281808 13.78125 11.99890 38.56250 15.00000 3.923611 27.25000  
## 4 1996-01-05 0.305804 14.09375 11.96243 39.25000 15.25000 3.972222 27.68750  
## 5 1996-01-08 0.309152 13.85938 11.96243 40.12500 15.18750 3.934028 27.81250  
## 6 1996-01-09 0.292411 13.53125 11.78008 39.67969 14.78125 3.631944 27.92188  
## DIS HD IBM INTC JNJ JPM KO MCD  
## 1 20.01773 10.527778 22.71875 7.328125 21.06250 19.58333 18.75000 22.7500  
## 2 20.14104 10.333333 22.31250 7.218750 21.90625 19.58333 18.90625 22.7500  
## 3 19.89442 10.305555 21.71875 7.187500 21.68750 18.75000 18.75000 22.8750  
## 4 20.26435 10.055555 22.15625 7.187500 21.68750 18.66667 18.65625 22.5000  
## 5 20.38767 9.777778 22.28125 7.203125 21.96875 18.66667 18.78125 22.5625  
## 6 20.41850 9.666667 21.68750 6.875000 22.09375 18.20833 18.53125 22.1875  
## MMM MRK MSFT NKE PG TRV UNH VZ WBA  
## 1 33.87500 32.1250 5.609375 4.445313 20.78125 28.2500 8.078125 30.46456 7.53125  
## 2 33.81250 31.6875 5.429688 4.312500 21.40625 28.6250 8.109375 31.42009 7.50000  
## 3 33.68750 31.8750 5.460938 4.265625 21.75000 29.0000 8.187500 30.85801 7.40625  
## 4 33.75000 31.5000 5.398438 4.132813 21.84375 29.0625 7.859375 31.19526 7.68750  
## 5 33.50000 31.9375 5.390625 4.203125 21.93750 29.1875 7.703125 30.97043 7.62500  
## 6 33.01562 31.7500 5.011719 4.117188 21.90625 28.9375 7.265625 30.93530 7.56250  
## WMT  
## 1 11.6250  
## 2 11.7500  
## 3 11.8750  
## 4 11.6875  
## 5 11.6875  
## 6 11.5000

1. Calculate daily log return for each stock.

daily.log.return <- as.data.frame(sapply(df[2:26], function(x) diff(log(x))))  
head(daily.log.return)

## AAPL AMGN AXP BA CAT CSCO  
## 1 0.00000000 -0.01078759 0.000000000 -0.009434032 0.016667052 -0.040071981  
## 2 -0.01766372 -0.04435317 -0.009077177 -0.025601398 -0.008298803 -0.038199144  
## 3 0.08171839 0.02242246 -0.003044156 0.017671143 0.016529302 0.012313233  
## 4 0.01088869 -0.01676954 0.000000000 0.022048137 -0.004106782 -0.009661798  
## 5 -0.05567272 -0.02396007 -0.015361271 -0.011160162 -0.027113235 -0.079895795  
## 6 0.04478403 -0.02573241 -0.034649121 -0.033433214 -0.010627093 0.020814407  
## CVX DIS HD IBM INTC JNJ  
## 1 0.002361276 0.006141243 -0.018642404 -0.018043515 -0.015037877 0.039277776  
## 2 0.027908788 -0.012320435 -0.002691813 -0.026971117 -0.004338402 -0.010035927  
## 3 0.015927527 0.018424292 -0.024557852 0.019943681 0.000000000 0.000000000  
## 4 0.004504512 0.006066728 -0.028012958 0.005625894 0.002171554 0.012884931  
## 5 0.003924872 0.001510949 -0.011428684 -0.027009460 -0.046623316 0.005673774  
## 6 -0.052265699 -0.040570375 0.011428684 0.005747142 -0.016036999 -0.033072748  
## JPM KO MCD MMM MRK MSFT  
## 1 0.000000000 0.008298803 0.000000000 -0.001846723 -0.013712262 -0.032557631  
## 2 -0.043485146 -0.008298803 0.005479466 -0.003703708 0.005899722 0.005738896  
## 3 -0.004454386 -0.005012542 -0.016529302 0.001853569 -0.011834458 -0.011510917  
## 4 0.000000000 0.006677821 0.002773927 -0.007434978 0.013793322 -0.001448319  
## 5 -0.024859965 -0.013400536 -0.016760169 -0.014564505 -0.005888143 -0.072882364  
## 6 -0.009195541 -0.017007213 -0.017045867 -0.008077972 -0.036076056 0.026914398  
## NKE PG TRV UNH VZ WBA  
## 1 -0.030332500 0.029631798 0.013187004 0.003861009 0.030883522 -0.004158010  
## 2 -0.010929071 0.015930822 0.013015368 0.009587801 -0.018051105 -0.012578782  
## 3 -0.031630423 0.004301082 0.002152853 -0.040901514 0.010869672 0.037271395  
## 4 0.016870007 0.004282662 0.004291852 -0.020080996 -0.007233283 -0.008163311  
## 5 -0.020657890 -0.001425517 -0.008602204 -0.058471768 -0.001134952 -0.008230499  
## 6 -0.003802285 -0.012921931 -0.010857870 -0.002152853 -0.007980903 -0.008298803  
## WMT  
## 1 0.01069529  
## 2 0.01058211  
## 3 -0.01591546  
## 4 0.00000000  
## 5 -0.01617286  
## 6 -0.01643873

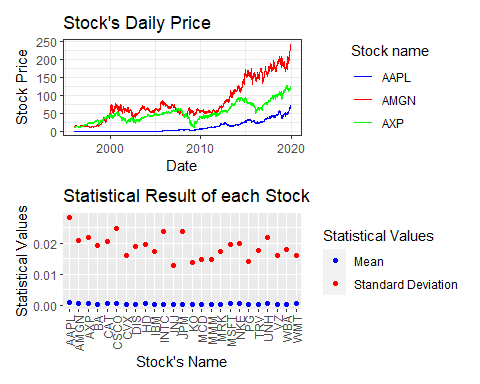
1. Calculate the mean and standard deviation of log return for each stock

mean.and.std <- as.data.frame(sapply(daily.log.return, function(x)   
 c("Mean" = mean(x, na.rm=TRUE),  
 "Standard.deviation" = sd(x)  
 )  
))  
mean.and.std

## AAPL AMGN AXP BA  
## Mean 0.0009168344 0.0004641248 0.0003855638 0.0003478159  
## Standard.deviation 0.0284047796 0.0208557858 0.0219789482 0.0193786437  
## CAT CSCO CVX DIS  
## Mean 0.000379848 0.0004002217 0.0002502413 0.0003264233  
## Standard.deviation 0.020493476 0.0247623992 0.0159682591 0.0188428129  
## HD IBM INTC JNJ  
## Mean 0.0005012099 0.0002923392 0.0003470648 0.0003197527  
## Standard.deviation 0.0195658648 0.0173720510 0.0237095985 0.0129114075  
## JPM KO MCD MMM  
## Mean 0.0003240281 0.0001789796 0.0003573148 0.0002726557  
## Standard.deviation 0.0238587249 0.0138393263 0.0149395005 0.0149365129  
## MRK MSFT NKE PG  
## Mean 0.0001724428 0.0005522446 0.0005167696 0.0002963599  
## Standard.deviation 0.0173385301 0.0195438394 0.0199581843 0.0142370700  
## TRV UNH VZ WBA  
## Mean 0.0002607877 0.0005950182 0.000115521 0.0003405546  
## Standard.deviation 0.0177976490 0.0219630363 0.015967949 0.0181026340  
## WMT  
## Mean 0.0003856492  
## Standard.deviation 0.0160858239

1. Build a graph with two sub-plots.

library(ggplot2)  
library(patchwork)  
knitr::opts\_chunk$set(fig.width=unit(18,"cm"), fig.height=unit(11,"cm"))  
p1 <- ggplot() + geom\_line(data=df, aes(x=X, y=AAPL, color = "AAPL")) + geom\_line(data=df, aes(x=X, y=AMGN, color = "AMGN")) + geom\_line(data=df, aes(x=X, y=AXP, color = "AXP")) + theme\_bw() + labs(y="Stock Price", x="Date", title="Stock's Daily Price") +   
 scale\_color\_manual(name = "Stock name", values = c("AAPL" = "blue", "AMGN" = "red", "AXP" = "green"))  
p2 <- ggplot() + geom\_point(data = stack(mean.and.std[1,]), aes(x = ind, y = values, color = "Mean")) +   
 geom\_point(data = stack(mean.and.std[2,]), aes(x = ind, y = values, color = "Standard Deviation")) + labs(x="Stock's Name", y="Statistical Values", title="Statistical Result of each Stock") + theme(axis.text.x = element\_text(angle = 90, vjust = 0.5, hjust=1)) +  
 scale\_color\_manual(name = "Statistical Values", values = c("Mean" = "blue", "Standard Deviation" = "red"))  
p1 / p2



# Part II

1. Download Amazon daily stock price data from 2021-01-01 to 2021-12-31 and save the data to a csv file.

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

## Loading required package: xts

## Loading required package: zoo

##   
## Attaching package: 'zoo'

## The following objects are masked from 'package:base':  
##   
## as.Date, as.Date.numeric

## Loading required package: TTR

## Registered S3 method overwritten by 'quantmod':  
## method from  
## as.zoo.data.frame zoo

start.date <- as.Date('2021-01-01')  
end.date <- as.Date('2021-12-31')  
getSymbols('AMZN', src = 'yahoo', from = start.date, to = end.date, warnings = FALSE, auto.assign = TRUE)

## [1] "AMZN"

amazon.data <- data.frame(AMZN)  
amazon.data$date <- rownames(amazon.data)  
rownames(amazon.data) <- NULL  
write.zoo(amazon.data, "amazon.csv", sep = ",")

1. Calculate weekly log returns based on adjusted close price.

log.return <- diff(log(AMZN$AMZN.Adjusted))[-1]  
weekly.log.return <- apply.weekly(log.return, FUN = sum)  
head(weekly.log.return)

## AMZN.Adjusted  
## 2021-01-08 -0.001234051  
## 2021-01-15 -0.024957760  
## 2021-01-22 0.058793021  
## 2021-01-29 -0.026478700  
## 2021-02-05 0.044515496  
## 2021-02-12 -0.022456923

1. Calculate median, mean, standard deviation of log returns.

mean.d <- mean(log.return)  
median.d <- median(log.return)  
standard.deviation <- sd(log.return)  
  
cat(" Mean:", round(mean.d, 6), "\n",  
 "Median:", round(median.d, 6), "\n",  
 "Standard Deviation:", round(standard.deviation, 6))

## Mean: 0.000227   
## Median: 0.001126   
## Standard Deviation: 0.015196

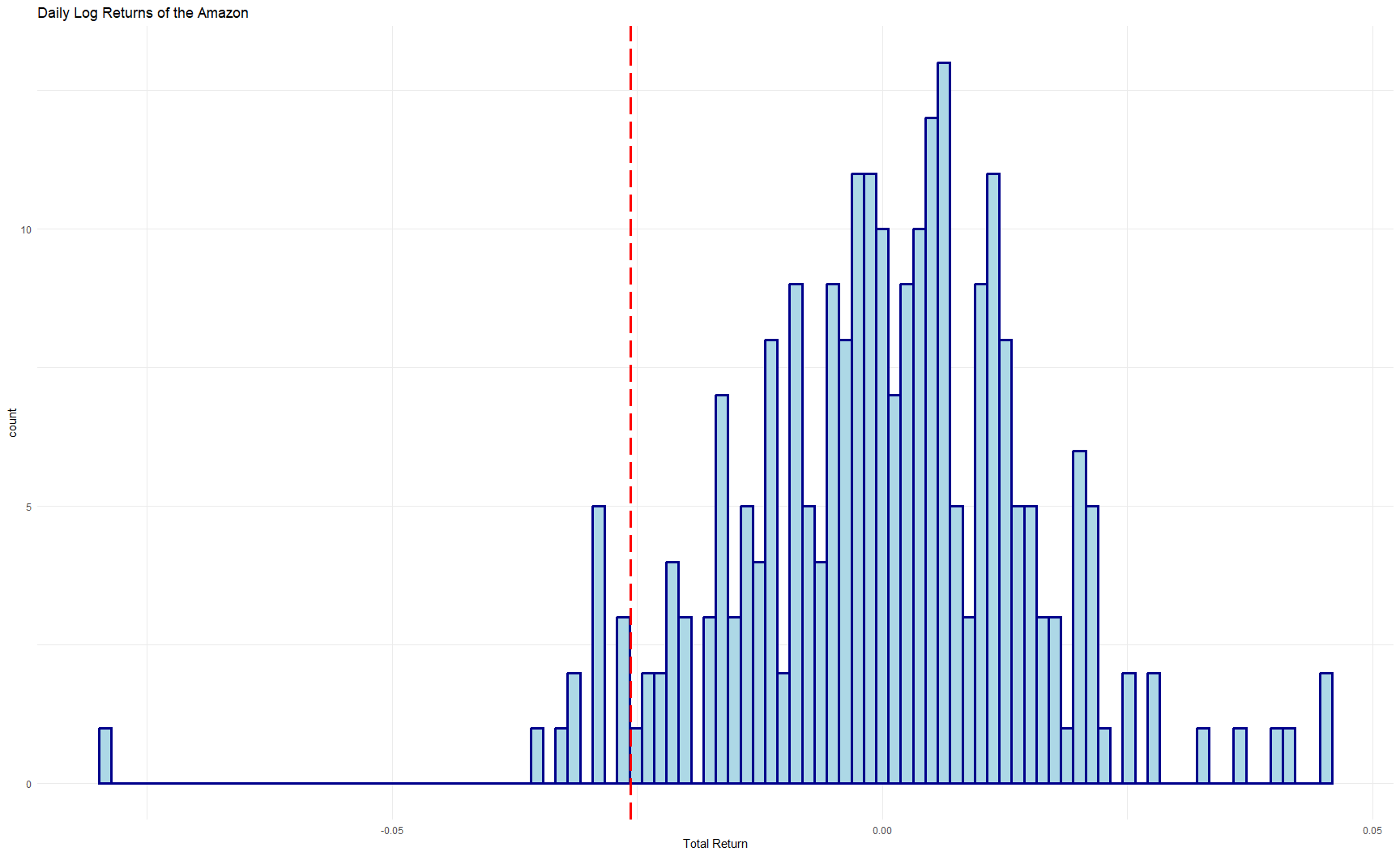
1. Plot the distribution of stock daily log returns

log.return.plot <- ggplot(data = log.return, aes(x = log.return$AMZN.Adjusted)) + geom\_histogram(color = "darkblue", fill = "lightblue", size = 1.2, bins = 100) + ggtitle("Daily Log Returns of the Amazon") + geom\_vline(xintercept = quantile(x = as.vector(log.return), probs = 0.05),   
 color = "red", size = 1.2, linetype = "longdash") + xlab("Total Return") + theme\_minimal()

## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.  
## ℹ Please use `linewidth` instead.

log.return.plot

## Don't know how to automatically pick scale for object of type <xts/zoo>.  
## Defaulting to continuous.



1. Observation in this series with log return is between 0.01 and 0.015

sum(log.return > 0.01 & log.return < 0.015)

## [1] 31