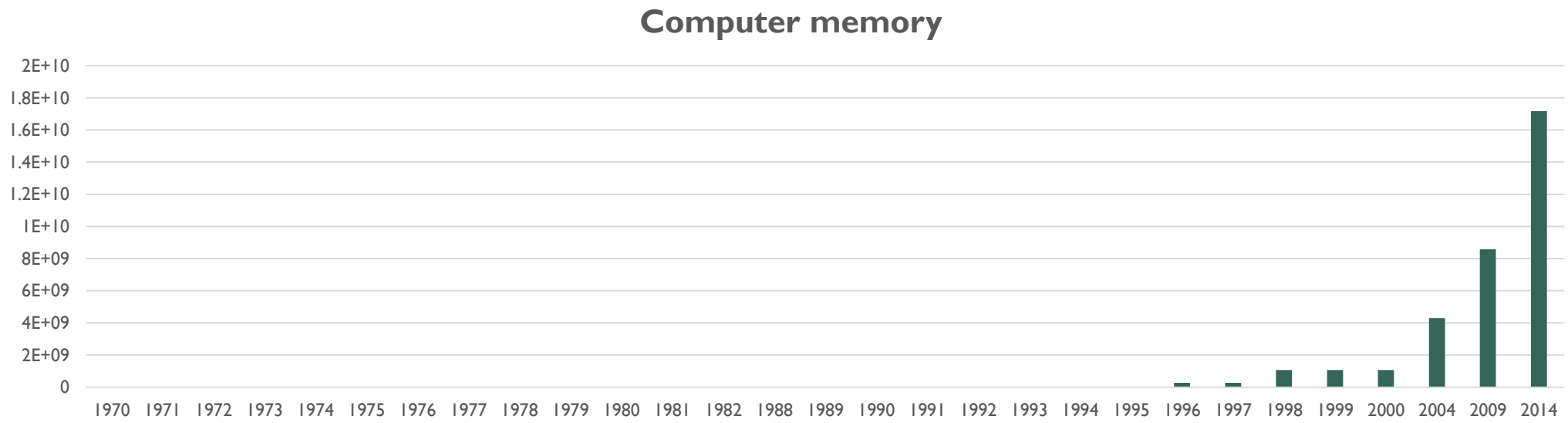
HW1: Q1

Original data of memory size(hand collected)

Year	type	memory size	ratio
1970s	DIP	256KB	1
1982	SIMM	256KB	1
1988-1990	FPM DRAM	2MB	8
1991-1995	EDO	16MB	64
1996-2000	SDR SDRAM	256MB	1024
1996	DDR SDRAM	1GB	4096
2004	DDR2	4GB	16384
2009	DDR3	8GB	32768
2014	DDR4	16GB	65536

HW1: Q1

year	1970	1971	1972	1973	1974	1975	1976	1977	1978	1979	1980	1981	1982	1988	1989	1990	1991	1992	1993	1994	1995	1996	1997	1998	1999	2000	2004	2009	2014
Byte	262144	262144	262144	262144	262144	262144	262144	262144	262144	262144	262144	262144	262144	2097152	2097152	2097152	16777216	16777216	16777216	16777216	16777216	2.68E+08	2.68E+08	1.07E+09	1.07E+09	1.07E+09	4.29E+09	8.59E+09	1.72E+10



HW1 : Q2

Logistic regression, also called **logit regression**, or **logit model,** was developed by statistician David Cox in 1958.
Logistic regression is a type of **generalized linear model**, where the **dependent variable is categorical**.
The most common one is **binary logistic model** —that is, where the output can take only two values, "0" and "1", which represent outcomes such as pass/fail, win/lose, alive/dead or healthy/sick.
Cases where the dependent variable has more than two outcome categories may be analysed in **multinomial logistic regression**, or, if the multiple categories are ordered, in **ordinal logistic regression**.
Logistic regression is used in various fields, including machine learning, most medical fields, and social sciences.
In the terminology of economics, logistic regression is an example of a qualitative/response/discrete choice model.

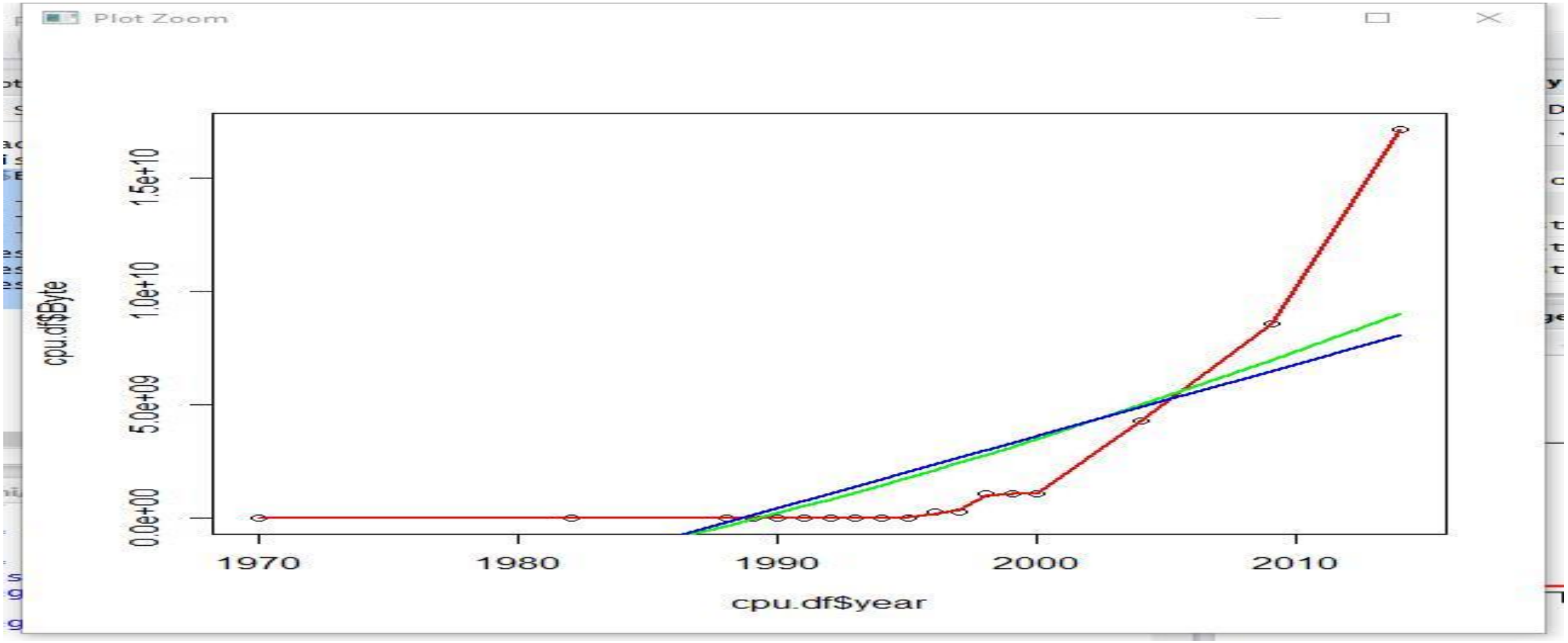
HW1 : Q2

Like other forms of regression analysis, logistic regression makes use of **one or more predictor variables** that may be **either continuous or categorical**. Unlike ordinary linear regression, however, logistic regression is used for predicting **dependent variables that take membership in one of a limited number of categories** (treating the dependent variable in the binomial case as the outcome of a Bernoulli trial) rather than a continuous outcome. Given this difference, **the assumptions of linear regression are violated**. In particular, **the residuals cannot be normally distributed**. In addition, linear regression may make nonsensical predictions for a binary dependent variable. What is needed is a way to convert a binary variable into a continuous one that can take on any real value (negative or positive). To do that, binomial logistic regression first takes the odds of the event happening for different levels of each independent variable, then takes the ratio of those odds (which is continuous but cannot be negative) and then takes the logarithm of that ratio (this is referred to as logit or log-odds) to create a continuous criterion as a transformed version of the dependent variable.

HW2: Q1&2

Year	type	memory size	ratio
1970s	DIP	256KB	1
1982	SIMM	256KB	1
1988-1990	FPM DRAM	2MB	8
1991-1995	EDO	16MB	64
1996-2000	SDR SDRAM	256MB	1024
1996	DDR SDRAM	1GB	4096
2004	DDR2	4GB	16384
2009	DDR3	8GB	32768
2014	DDR4	16GB	65536

HW2: Q1&2


Plot Zoom
cpu.df$Byte
0.0e+00
5.0e+09
1.0e+10
1.5e+10
1970
1980
1990
2000
2010
cpu.df$year

HW2：Q3

- (1)18.04%
- (2)1%

- lambda=2
- x=3
- probex1=exp(-lambda)*lambda^x/factorial(x)
- probex1

- lambda=5
- x=0
- probex2=exp(-lambda)*lambda^x/factorial(x)
- probex2

HW3: Q1

```
install.packages("digest")
library(digest)
digest("I learn a lot from this class when I am proper listening to the professor", "sha256")
digest("I do not learn a lot from this class when I am absent and playing on my Iphone", "sha256")
```

```
digest("I learn a lot from this class when I am proper listening to the professor", "sha256") [1]
"c16700de5a5c1961e279135f2be7dcf9c187cb6b21ac8032308c715e1ce9964c" > digest("I do not learn a lot from this class when I am absent and playing on my Iphone", "sha256") [1]
"2533d529768409d1c09d50451d9125fdbaa6e5fd4efdeb45c04e3c68bcb3a63e"
```

HW3: Q2

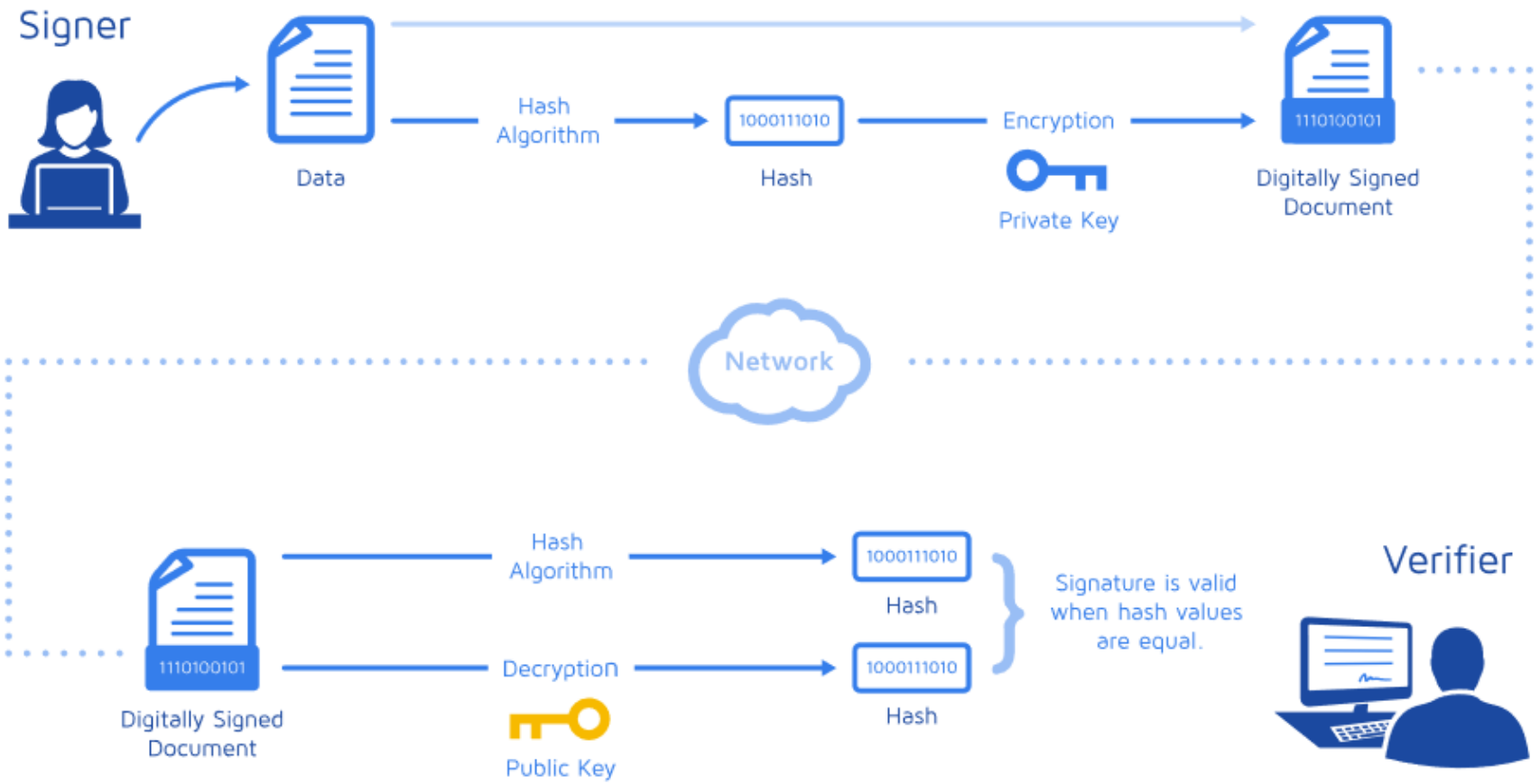
The Digital Signature Algorithm (DSA) is a **United States Federal Government standard for digital signatures.**

DSA was proposed by the National Institute of Standards and Technology (NIST) in August 1991 for use in their Digital Signature Standard (DSS), specified in FIPS 186.

DSA can be used by the recipient of a message to **verify that the message has not been altered during transit** as well as as **certain the originator's identity**

HW3: Q2

How do digital signatures work?



HW3: Q2

The Digital Signature is usually performed in several steps:

1. **Calculate the Message Digest**(hash-value of the message)

In the first step of the process, a hash-value of the message (often called the message digest) is calculated by applying some cryptographic hashing algorithm

2. Calculate the Digital Signature

In the second step of digitally signing a message, the information obtained in the first step hash-value of the message (the message digest) is encrypted with the private key of the person who signs the message and thus an encrypted hash-value, also called digital signature, is obtained. For this purpose, some mathematical cryptographic encrypting algorithm for calculating digital signatures from given message digest is used, which includes **DSA, TSA, ECDSA and so on**.

3. **Verifying Digital Signatures**

The public key is used in the signature verification process to verify the authenticity of the signature

HW3: Q2

Compared with other encrypting algorithms:

- DSA is based on the theory of the discrete logarithms.

- Cryptographic hash function H used in DSA was always SHA-1, but the stronger SHA-2 hash functions are approved for use in the current DSS

- key length *L* has to be a multiple of 64 between 512 and 1,024 (inclusive)

HW3: Q3

```
install.packages("rjson")
library(rjson)
json_file = "http://crix.hu-berlin.de/data/crix.json"
json_data = fromJSON(file=json_file)
x = as.data.frame(json_data)

date1=c(json_data[[1]]$date)
for (i in 1:50){

  date1[i]=c(json_data[[i]]$date)
}

price1=c(json_data[[1]]$price)
for (i in 1:50){

  price1[i]=c(json_data[[i]]$price)
}
```

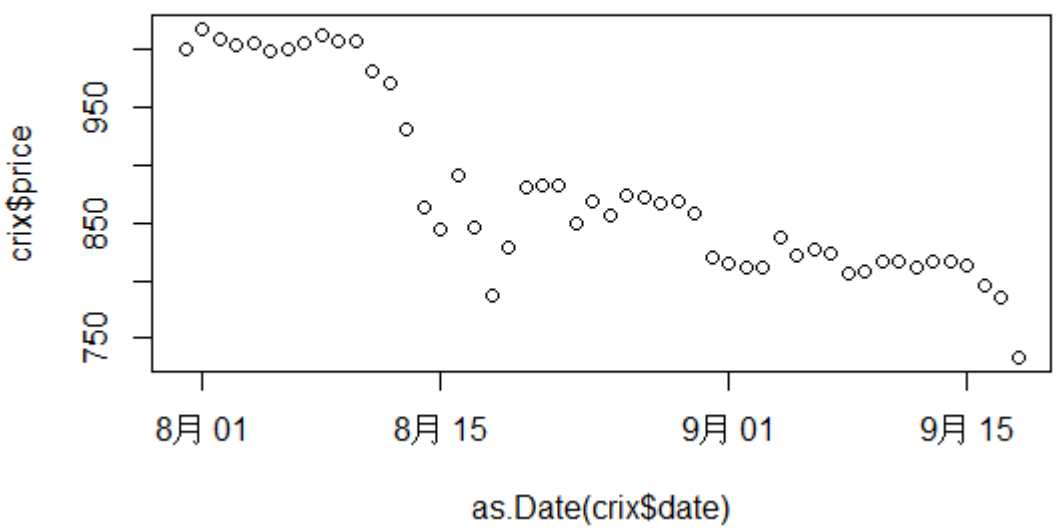
HW3：Q3

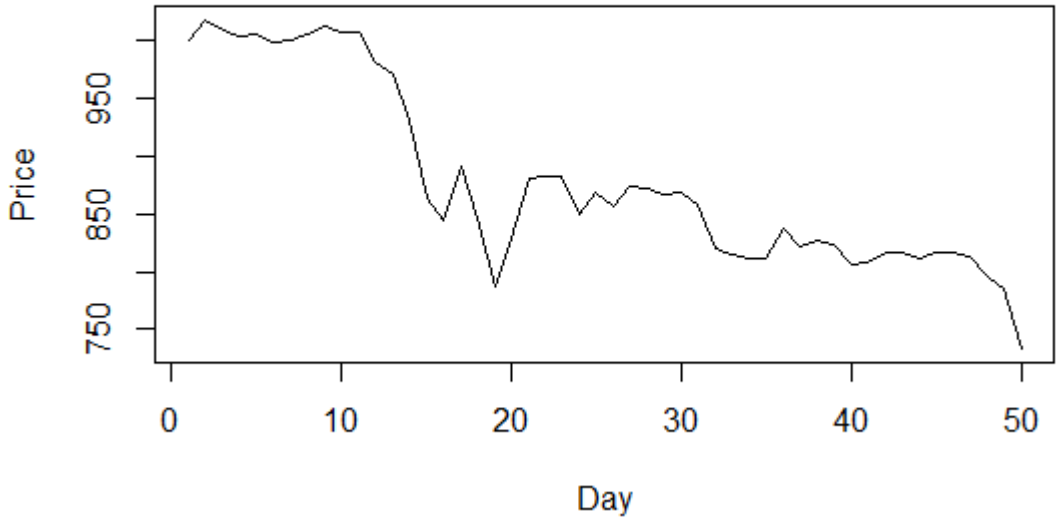
```
date=date1
price=price1
crix=data.frame(date,price)

plot(crix$price~as.Date(crix$date))
plot(crix$price~crix$date,type="b")
plot(ts(crix$price,freq=1),type='l',xlab='Day',ylab='Price')
```

HW3：Q3

Plot of time series





HW3: Q3

```
install.packages("caschrono")
install.packages("TTR")
install.packages("fGarch")
install.packages("rugarch")
install.packages("forecast")
install.packages("TSA")

library(caschrono)
library(TTR)
library(fGarch)
library(rugarch)
library(forecast)
library(TSA)
```

HW3：Q3

```
#*******************ARIMA model******************
xy.acfb(crix$price,numer=FALSE)
adf.test(crix$price)
#Augmented Dickey-Fuller Test:not stationary

##******1)return
r=diff(log(crix$price))*100
plot(r,type="b")
abline(h = 0)
plot(r,type="l")
```

HW3：Q3

```
#******2)Model Specification ARIMA(p,d,q)
#ADF test-H0:unit root H1:no unit root(test for stationarity)
adf.test(r)
#p-value=0.27,not stationary.
dr=diff(r)
plot(dr,type="b")
abline(h = 0)
adf.test(dr)
#p-value=0.01,stationary.(d=1)
```

HW3: Q3

```
#******3)Parameter Estimation
#estimation of p and q
a.fin1=auto.arima(dr)
summary(a.fin1)
#ARMA(0,0) therefore r fits ARIMA(0,1,0)
a.fin2=arima(r,order=c(0,1,0))
summary(a.fin2)
f=forecast(a.fin2,h=3,level=c(99.5))
acf(f$residuals,lag.max = 20)
Box.test(f$residuals,lag=20,type='Ljung-Box')
#the residuals follow Gaussian distribution
plot.ts(f$residuals)
```

HW3：Q3

```
#****4)some evidence to GARCH model
#get ACF and PACF of the residuals
xy.acfb(residuals(a.fin2),numer=FALSE)
xy.acfb((residuals(a.fin2))^2,numer=FALSE)+
  xy.acfb(abs(residuals(a.fin2)),numer=FALSE)

#get the Conditional heteroskedasticity test
McLeod.Li.test(y=residuals(a.fin2))
#p-values are all included in the test, it formally shows strong evidence for ARCH in this data.

#**Normality of the Residuals
qqnorm(residuals(a.fin2))
qqline(residuals(a.fin2))
shapiro.test(residuals(a.fin2))
#The QQ plot suggest that the distribution of returns may have a tail thicker that of a
#normal distribution and maybe somewhat skewed to the right
#p-value<0.05 reject the normality hypothesis
```

HW3：Q3

```
g1=garchFit(~garch(1,1),data=residuals(a.fin2),trace=FALSE,include.mean=TRUE, na.action=na.pass)
summary(g1)
g2=garchFit(~garch(1,2),data=residuals(a.fin2),trace=FALSE,include.mean=TRUE, na.action=na.pass)
summary(g2)
g3=garchFit(~garch(2,1),data=residuals(a.fin2),trace=FALSE,include.mean=TRUE, na.action=na.pass)
summary(g3)
g4=garchFit(~garch(2,2),data=residuals(a.fin2),trace=FALSE,include.mean=TRUE, na.action=na.pass)
summary(g4)
#The best one is Garch(1,1) model which has the smallest AIC.
```

HW4: Q1

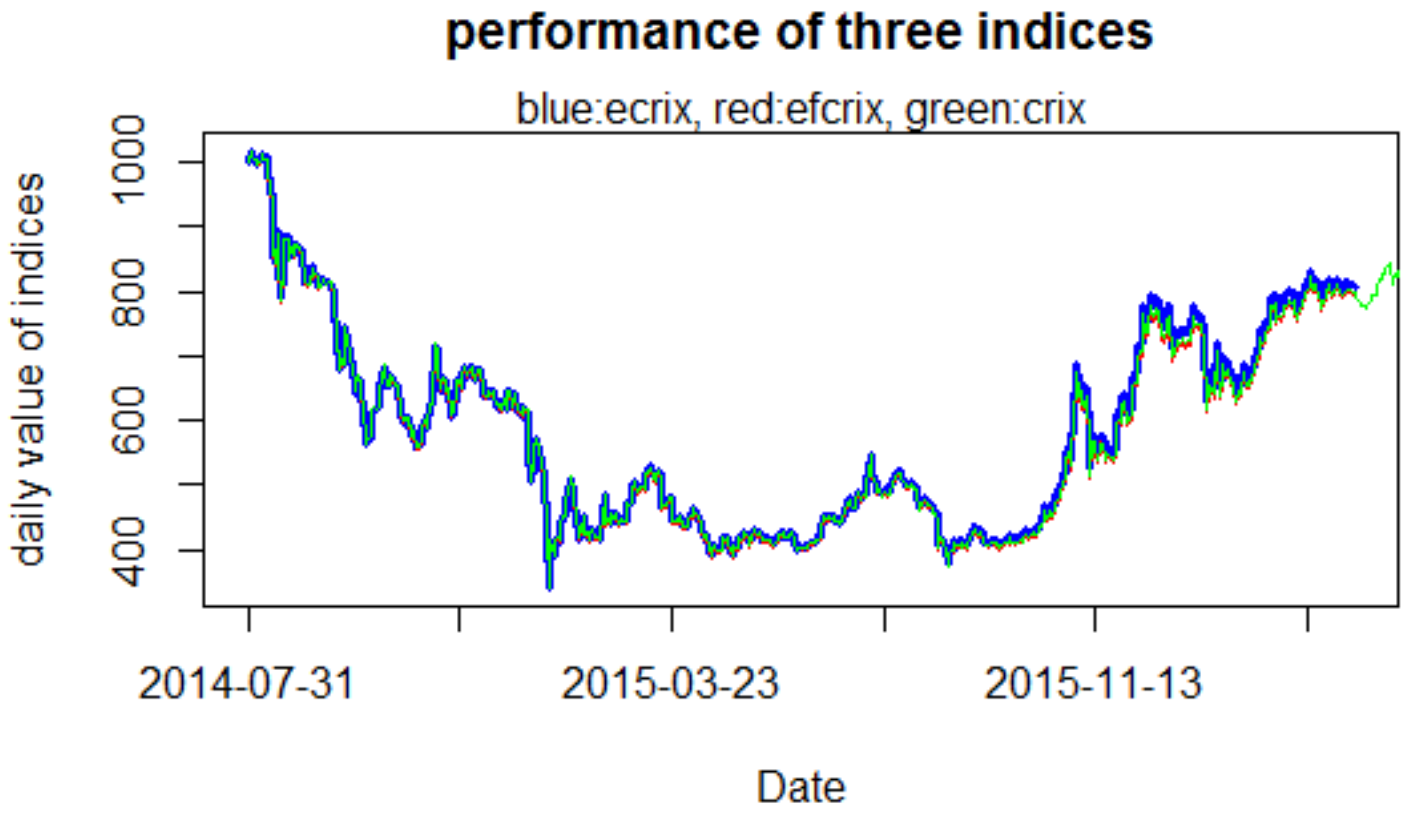


Figure 3

HW4: Q1

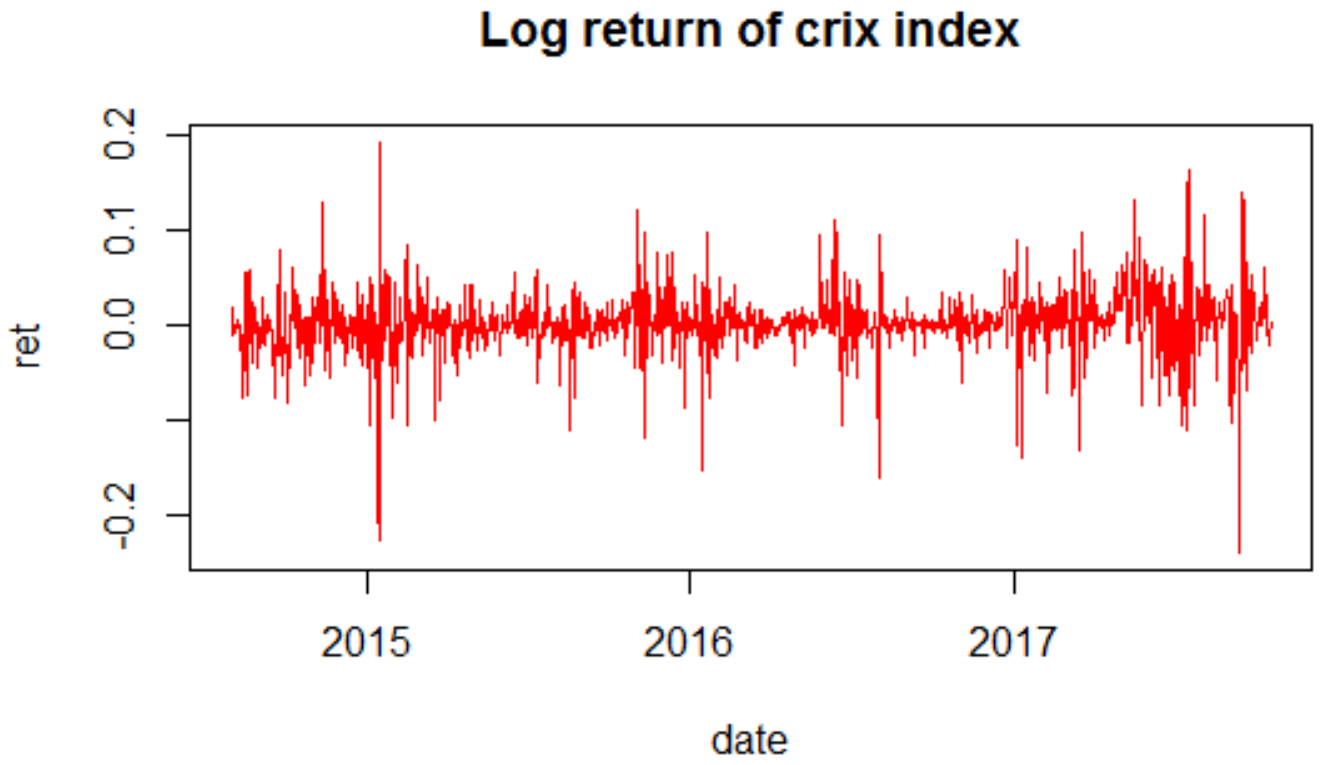


Figure 4

HW4: Q1

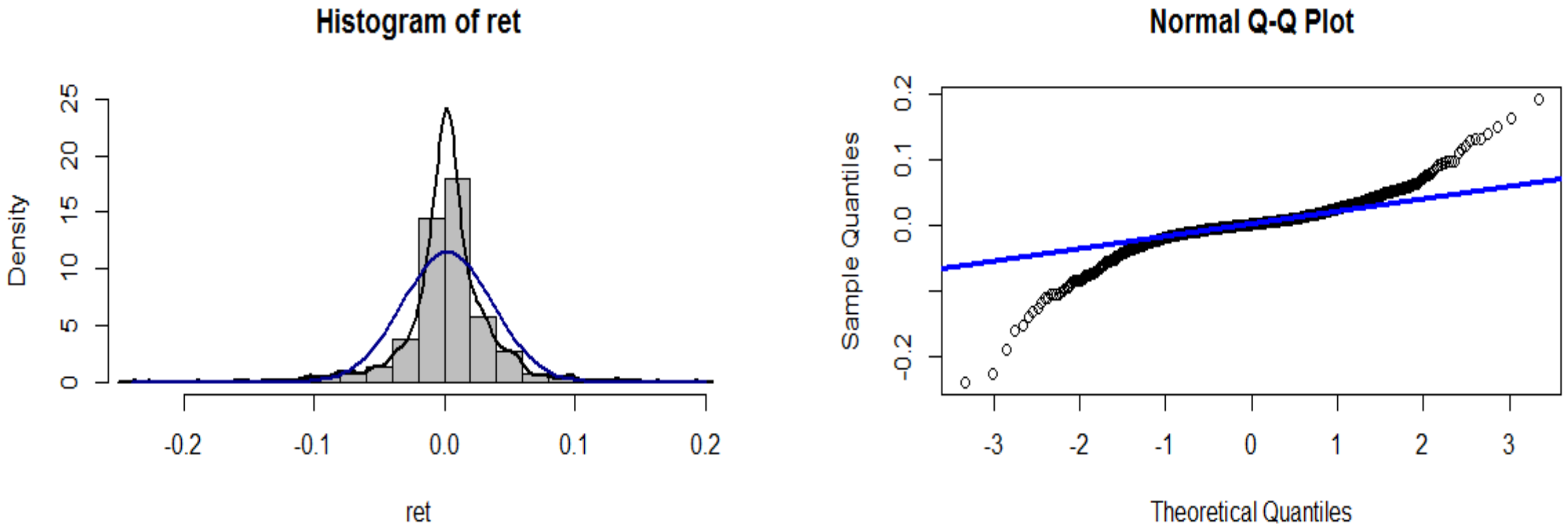


Figure 5

HW4: Q1

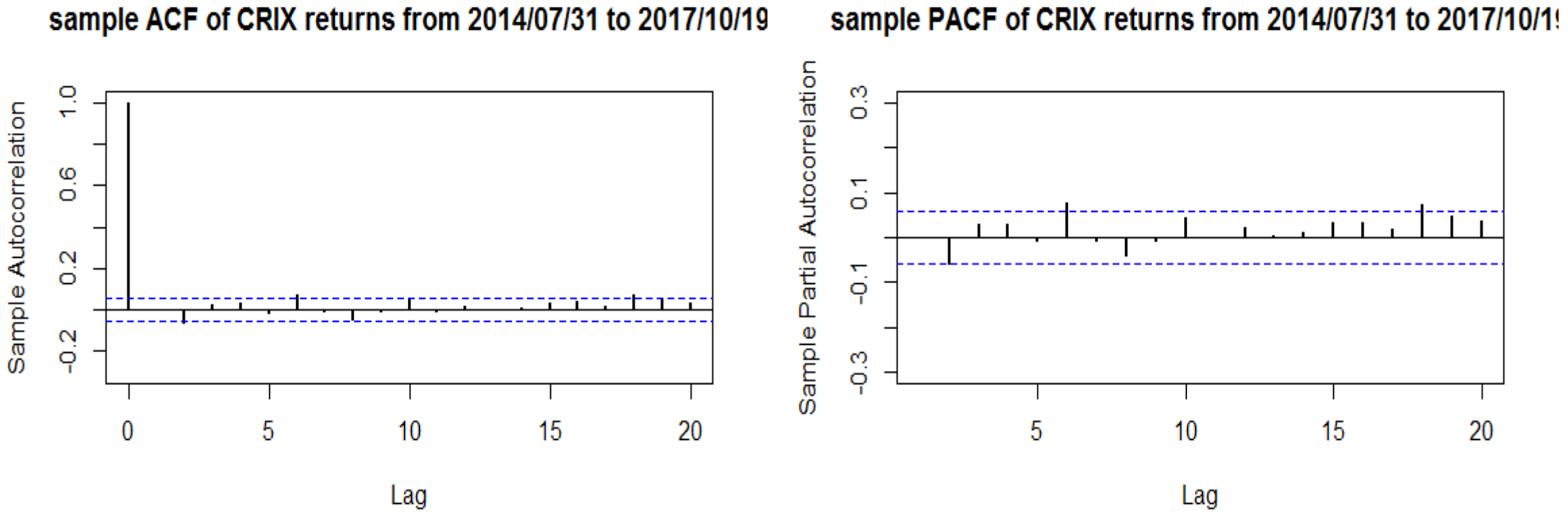


Figure 6

HW4: Q2

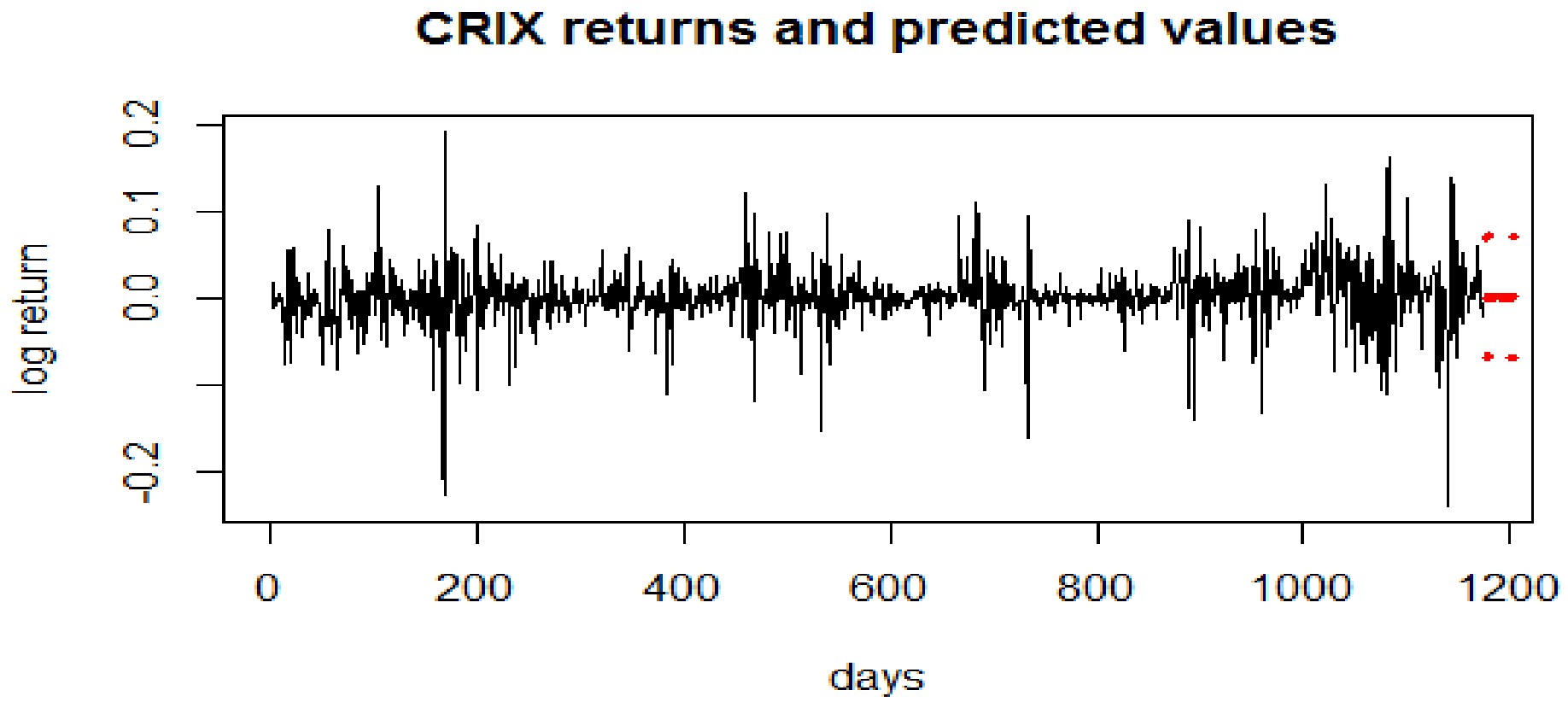


Figure 7