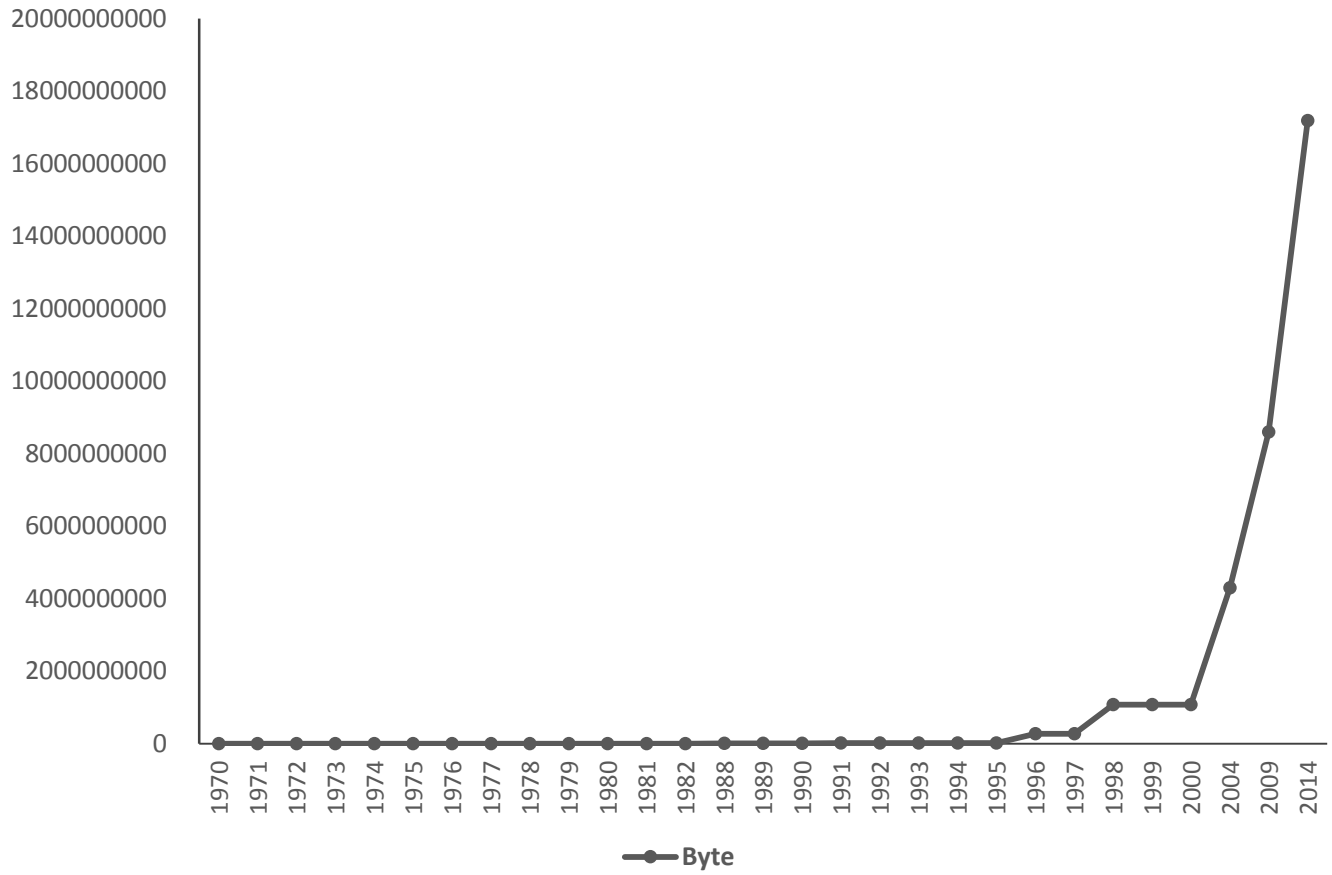
HW Unit 1

By: Qian Liu
15620161152269

1.

Year	Byte
1970	262144
1971	262144
1972	262144
1973	262144
1974	262144
1975	262144
1976	262144
1977	262144
1978	262144
1979	262144
1980	262144
1981	262144
1982	262144
1988	2097152
1989	2097152

Year	Byte
1990	2097152
1991	16777216
1992	16777216
1993	16777216
1994	16777216
1995	16777216
1996	268435456
1997	268435456
1998	1073741824
1999	1073741824
2000	1073741824
2004	4294967296
2009	8589934592
2014	17179869184


20000000000
18000000000
16000000000
14000000000
12000000000
10000000000
8000000000
6000000000
4000000000
2000000000
0
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

2.

Logistic regression

Logistic regression is used in various fields, including machine learning, most medical fields, and social sciences.

Logistic regression may be used to predict whether a patient has a given disease (e.g. diabetes; coronary heart disease), based on observed characteristics of the patient (age, sex, body mass index, results of various blood tests, etc.).

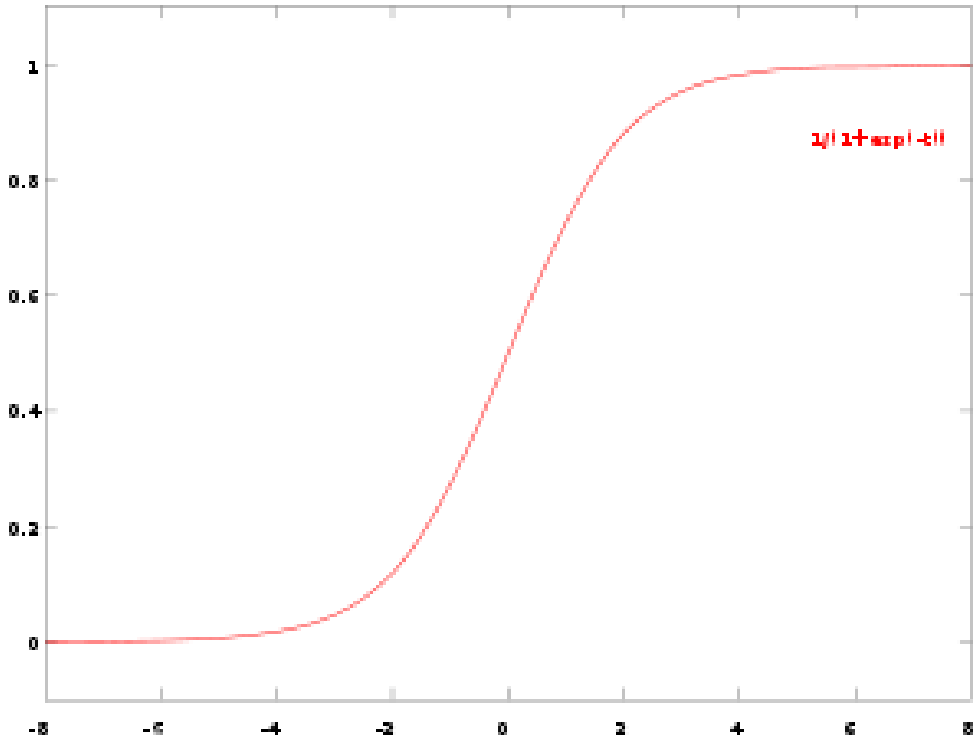
Another example might be to predict whether an American voter will vote Democratic or Republican, based on age, income, sex, race, state of residence, votes in previous elections, etc.

Stata

- logistic hcv age marry sex

Different from GLM

- Dependent variable: “0” or “1”, not cardinal numbers.



HW Unit 2

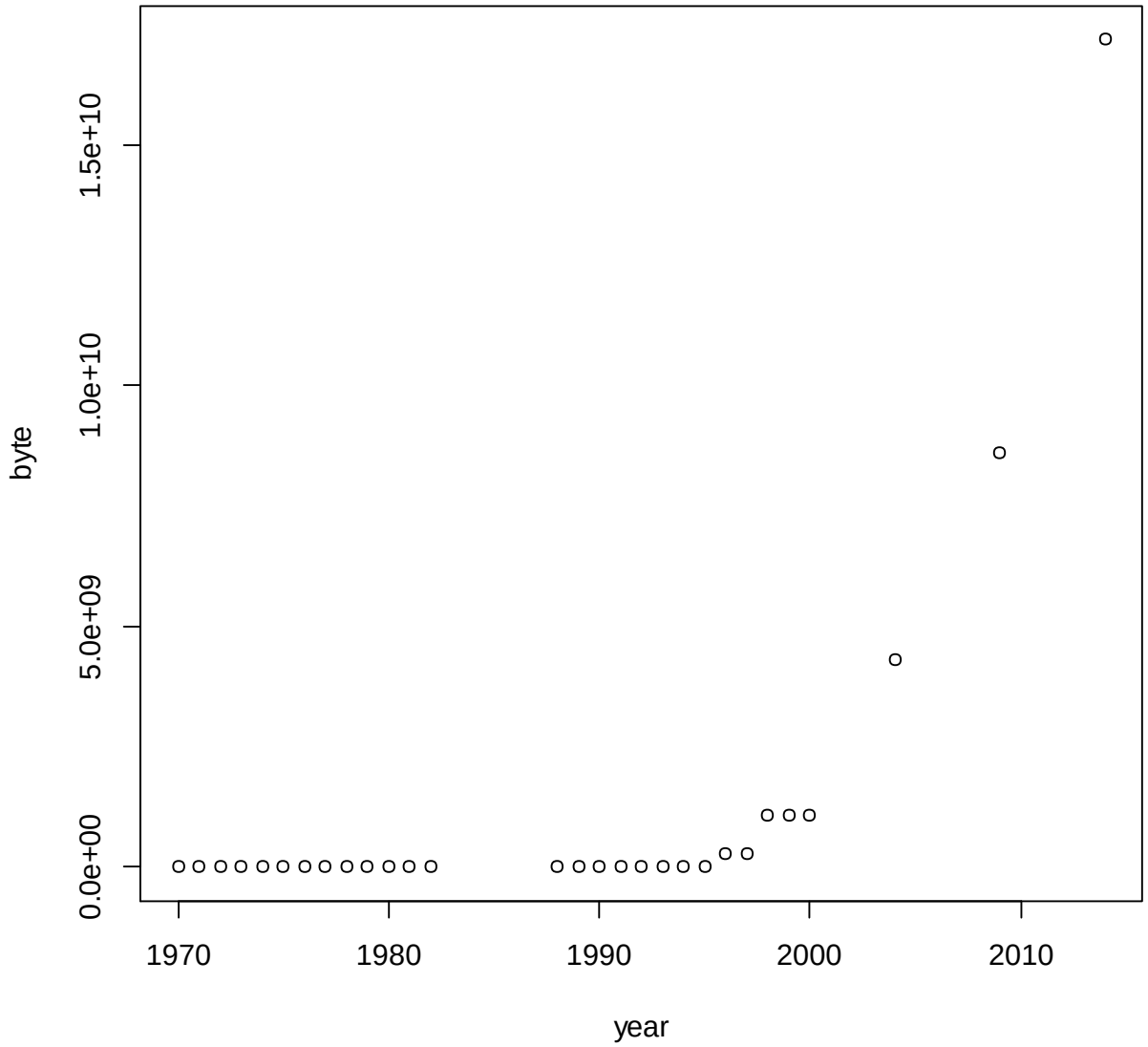
By: Qian Liu
15620161152269

1.

```
>year=c(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=c(262144,262144,262144,262144,262144,262144,262144,262144,262144,262144,26
2144,262144,262144,2097152,2097152,2097152,16777216,16777216,16777216,16777216,
16777216,268435456,268435456,1073741824,1073741824,1073741824,4294967296,85899
34592,17179869184)

> plot(year,byte)
```


byte
0.0e+00
5.0e+09
1.0e+10
1.5e+10
1970
1980
1990
2000
2010
year

2.

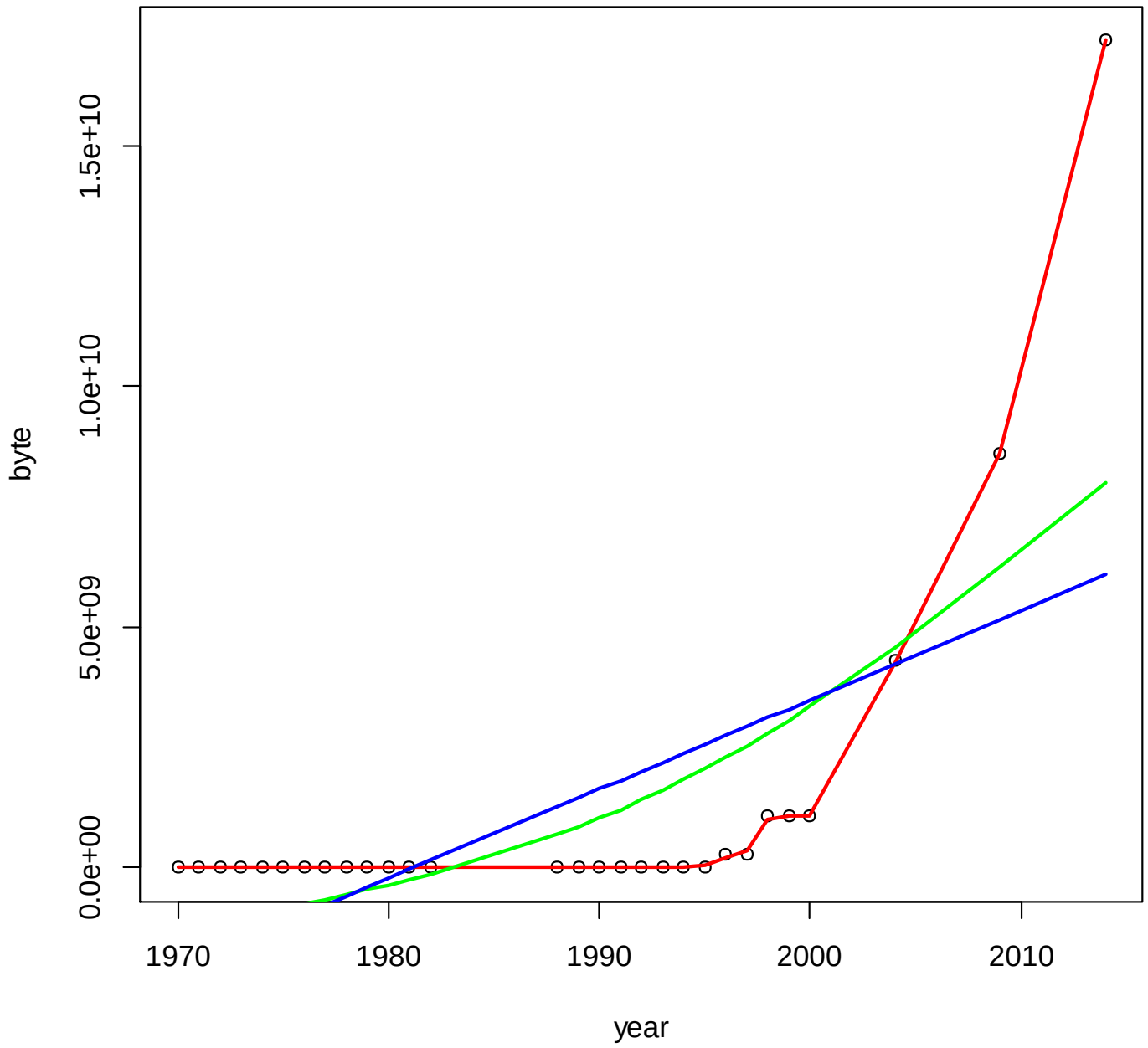
```
>year=c(1970,1971,1972,1973,1974,1975,1976,1977,1978,1979,1980,1981,1982,1988,1989,
1990,1991,1992,1993,1994,1995,1996,1997,1998,1999,2000,2004,2009,2014)

>byte=c(262144,262144,262144,262144,262144,262144,262144,262144,262144,262144,26
2144,262144,262144,2097152,2097152,2097152,16777216,16777216,16777216,16777216,
16777216,268435456,268435456,1073741824,1073741824,1073741824,4294967296,85899
34592,17179869184)

> plot(year,byte)

> splines.reg.1= smooth.spline(x = year, y = byte, spar = 0.2)
> splines.reg.2 = smooth.spline(x = year, y = byte, spar = 1)
> splines.reg.3= smooth.spline(x = year, y = byte, spar = 2)

> plot(year,byte)
> lines(splines.reg.1, col = "red", lwd = 2)
> lines(splines.reg.2, col = "green", lwd = 2)
> lines(splines.reg.3, col = "blue", lwd = 2)
```


byte
0.0e+00
5.0e+09
1.0e+10
1.5e+10
1970
1980
1990
2000
2010
year

3.

3.1Suppose you observe that in n=1000 mails (in 1 week) you have about 2 scams. Use the LvB /Possion pdf to calculate that you have 6 scam emails in 2 weeks.

3.2In Scammyland you have 5 scams on average, what is the probability to have no scam mail.

3.1 R code:

```
> lambda=2
> x=3
> dpois(x,lambda)
```

3.2 R code:

```
> lambda=5
> x=0
> dpois(x,lambda)
```

HW Unit 3

By: Qian Liu
15620161152269

1.

```
> install.packages("digest", repos='http://cran.us.r-project.org')
> library("digest")
> sentence1=digest("I learn a lot from this class when I am proper listening to the professor","sha256")
> sentence2=digest("I do not learn a lot from this class when I am absent and playing on my Iphone","sha256")
> sentence1
[1] "c16700de5a5c1961e279135f2be7dcf9c187cb6b21ac8032308c715e1ce9964c"
> sentence2
[1] "2533d529768409d1c09d50451d9125fdbaa6e5fd4efdeb45c04e3c68bcb3a63e"
```

2. Make 3-5 slides (in PPTX) on the DSA (Digital Signature Algorithms)

A **digital signature** is a mathematical scheme for demonstrating the authenticity of digital messages or documents. A valid digital signature gives a recipient reason to believe that the message was created by a known sender (authentication), that the sender cannot deny having sent the message (non-repudiation), and that the message was not altered in transit (integrity).

Digital signatures are often used to implement electronic signatures, a broader term that refers to any electronic data that carries the intent of a signature, but not all electronic signatures use digital signatures.In some countries, including the United States, Turkey, India,Brazil, Indonesia, Mexico, Saudi Arabia, Switzerland and the countries of theEuropean Union, electronic signatures have legal significance.

1.

鲍勃

鲍勃的公钥

鲍勃的私钥

2.

帕蒂

道格

苏珊

每人一把

3.

苏珊

"Hey Bob, how about lunch at Taco Bell. I hear they have free refills!"

公钥加密

HNFmsEm6Un
BejhhyCGKO
KJUxhiygSBC
EiC0QYIh/Hn
3xgiKBcyLK1
UcYiYlxx2lCF
HDC/A

4.

鲍勃

HNFmsEm6Un
BejhhyCGKO
KJUxhiygSBC
EiC0QYIh/Hn
3xgiKBcyLK1
UcYiYlxx2lCF
HDC/A

私钥解密

"Hey Bob, how about lunch at Taco Bell. I hear they have free refills!"

5.

Hash

Digest

Digital Signature

6.

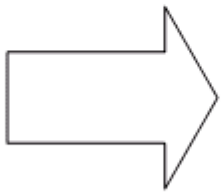
Digest

私钥加密

Signature

7.

Signature



Signature

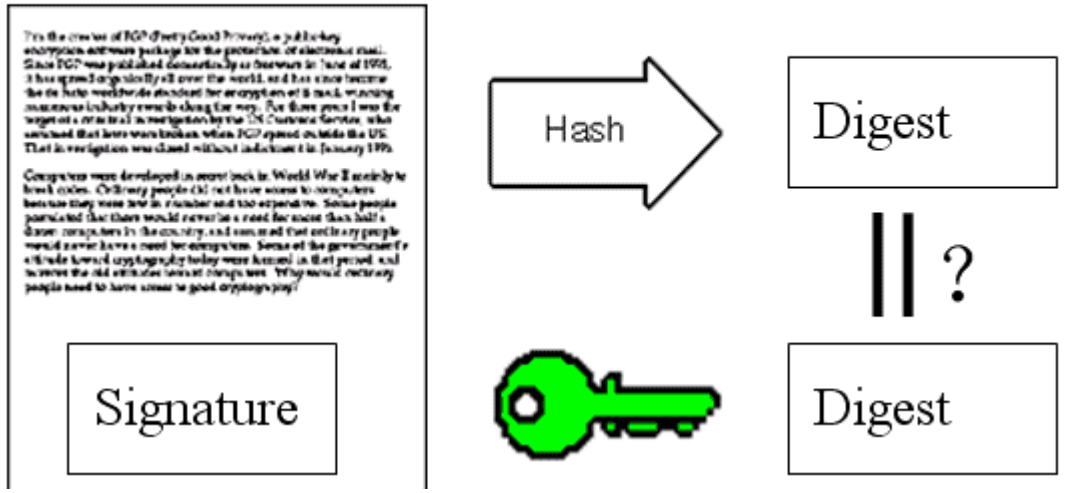
8.

Signature

公钥解密

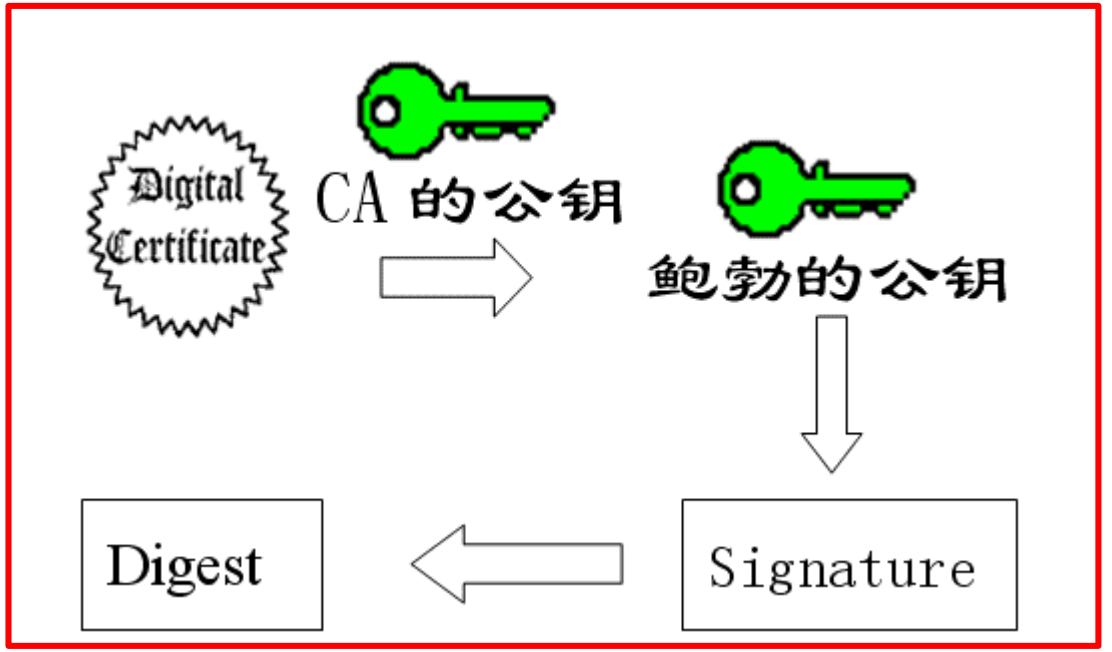
Digest

9.



Certificate Authority(CA)

Digital Certificate(DC)



Let H be the hashing function and m the message:

- Generate a random per-message value k where $1 < k < q$
- Calculate $r = (g^k \bmod p) \bmod q$
- In the unlikely case that $r = 0$, start again with a different random k
- Calculate $s = k^{-1} (H(m) + xr) \bmod q$
- In the unlikely case that $s = 0$, start again with a different random k
- The signature is (r, s)

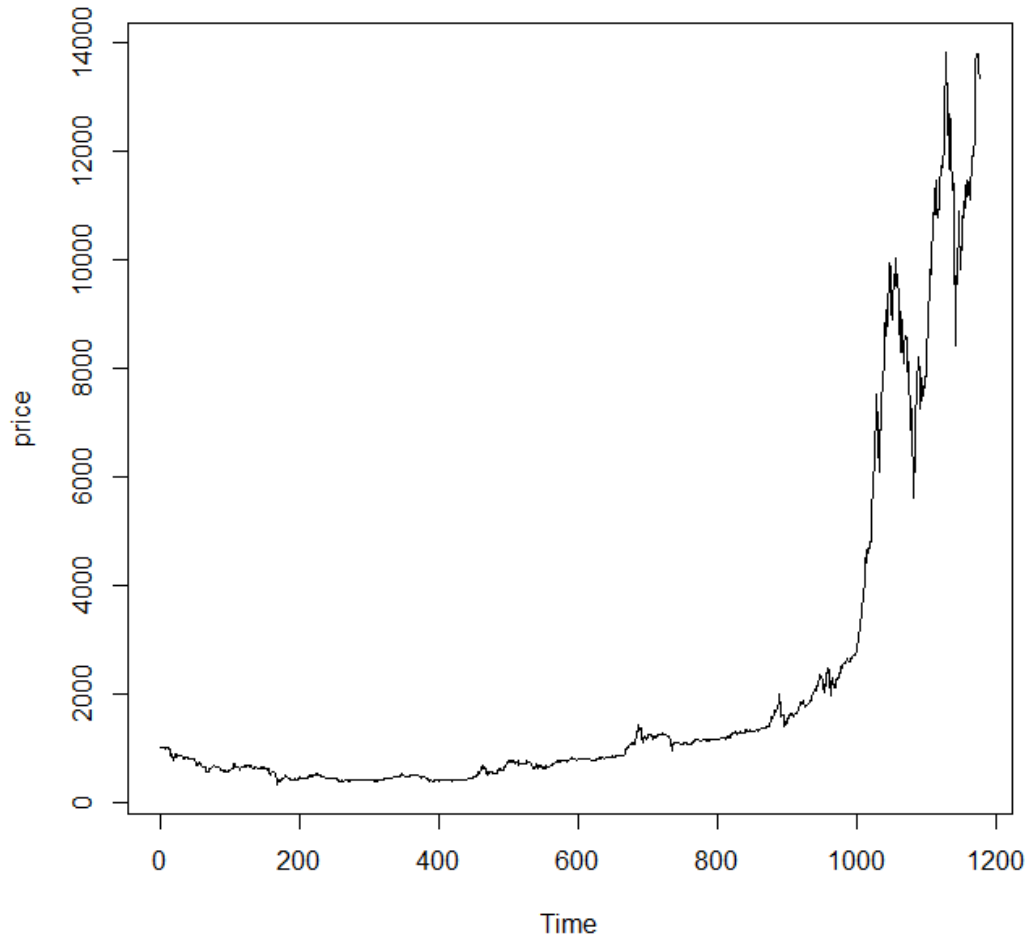
- Reject the signature if $0 < r < q$ or $0 < s < q$ is not satisfied.
- Calculate $w = s^{-1} \bmod q$
- Calculate $u_1 = H(m) \cdot w \bmod q$
- Calculate $u_2 = r \cdot w \bmod q$
- Calculate $v = (g^{u_1} y^{u_2} \bmod p) \bmod q$
- The signature is invalid unless $v = r$

3.

```
> install.packages("rjson",repos='htpp://cran.us.r-project.org')
> library("rjson")
> name<-c("Bob","Jon","Linda","Kim","Susan","Amy")
> gender<-c("M","M","F","M","F","F" )
> individualfeature<-data.frame(name,gender)
> data<-as.matrix( individualfeature)
> cat(toJSON(data))
["Bob","Jon","Linda","Kim","Susan","Amy","M","M","F","M","F","F"]
```

4.

```
> install.packages("rjson",repos='htpp://cran.us.r-project.org')
> library("rjson")
> json_file="http://crix.hu-berlin.de/data/crix.json"
> json_data=fromJSON(file=json_file)
> crix_data_frame=as.data.frame(json_data)
> x=crix_data_frame
> n=dim(x)
> a=seq(1,n[2],2)
> b=seq(2,n[2],2)
> date=t(x[1,a])
> price=t(x[1,b])
> ts.plot(price)
```



HW Unit 4

By: Qian Liu
15620161152269

1.

Fig 3

```
>libraries = c("ccgarch", "rmgarch", "xts", "zoo")
>lapply(libraries, function(x) if (!(x %in% installed.packages())) { install.packages(x) })
>lapply(libraries, library, quietly = TRUE, character.only = TRUE)

>load(file = "crix.RData")
>load(file = "ecrix.RData")
>load(file = "efcrix.RData")

>ecrix1 = zoo(ecrix, order.by = index(crix1))
>efcrix1 = zoo(efcrix, order.by = index(crix1))

>my.panel <- function(x, ...) {
  lines(x, ...)
  lines(ecrix1, col = "blue")
  lines(efcrix1, col = "red") }

>plot.zoo(crix1, plot.type = "multiple", type = "l", lwd = 1.5, panel = my.panel, main = "Indices in the CRIX family")
```

Fig 4

```
> install.packages("rjson",repos='htpp://cran.us.r-project.org')
> library("rjson")
> json_file="http://crix.hu-berlin.de/data/crix.json"
> json_data=fromJSON(file=json_file)
> crix_data_frame=as.data.frame(json_data)
> x=crix_data_frame
> n=dim(x)
> a=seq(1,n[2],2)
> b=seq(2,n[2],2)
> date=t(x[1,a])
> price=t(x[1,b])
> ts.plot(price)
> ret=diff(log(price))
> ts.plot(ret)
```

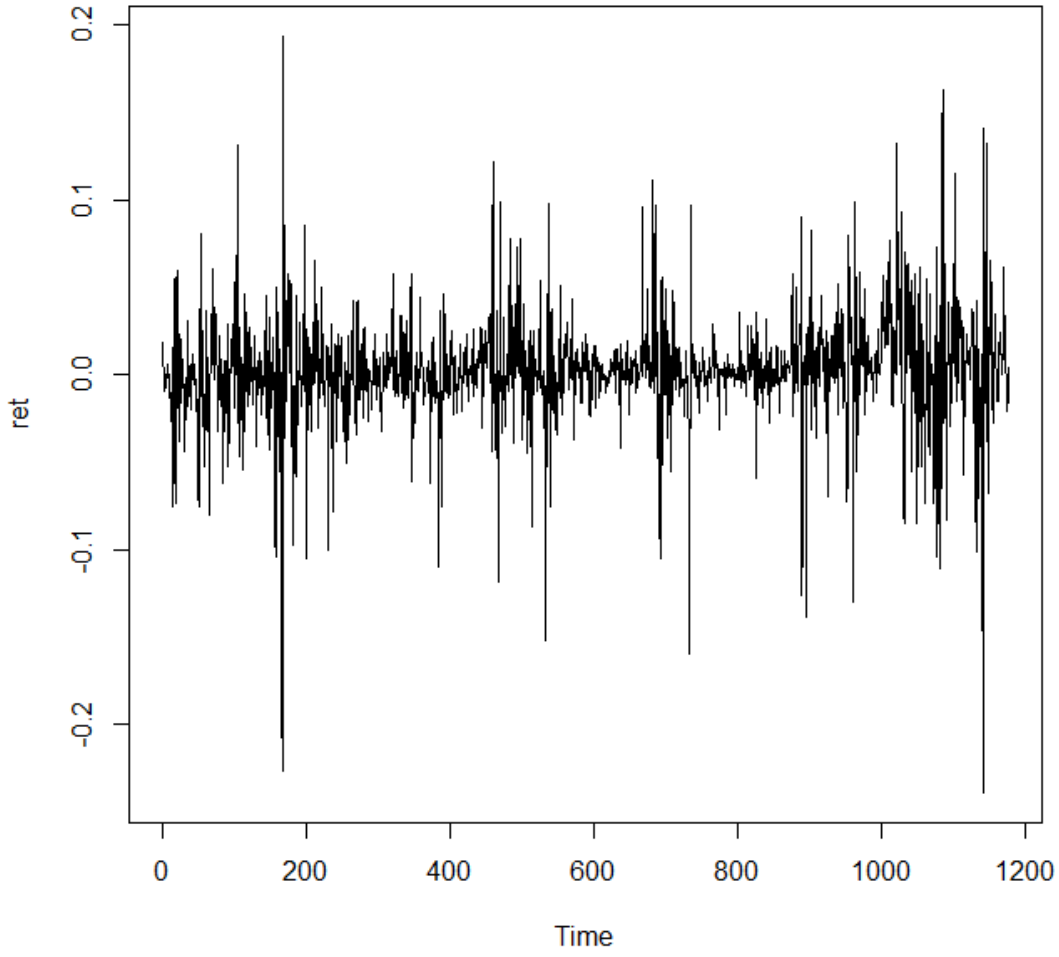


Fig 5

```
# install and load packages
>libraries = c("zoo", "tseries", "xts")
>lapply(libraries, function(x) if (!(x %in% installed.packages())) { install.packages(x) })
>lapply(libraries, library, quietly = TRUE, character.only = TRUE)

# load dataset
>load(file = "crix.RData")

# histogram of returns
>hist(ret, col = "grey", breaks = 20, freq = FALSE, ylim = c(0, 25), xlab = NA)
>lines(density(ret), lwd = 2)
>mu = mean(ret)
>sigma = sd(ret)
>x = seq(-4, 4, length = 100)
>curve(dnorm(x, mean = mean(ret), sd = sd(ret)), add = TRUE, col = "darkblue", lwd = 2)

# qq-plot
>qqnorm(ret)
>qqline(ret, col = "blue", lwd = 3)
```

Fig 6

```
> install.packages("rjson",repos='htpp://cran.us.r-project.org')
> library("rjson")
> json_file="http://crix.hu-berlin.de/data/crix.json"
> json_data=fromJSON(file=json_file)
> crix_data_frame=as.data.frame(json_data)
> x=crix_data_frame
> n=dim(x)
> a=seq(1,n[2],2)
> b=seq(2,n[2],2)
> date=t(x[1,a])
> price=t(x[1,b])
> ts.plot(price)
> ret=diff(log(price))
> ts.plot(ret)

# plot of pacf
>autopcorr=pacf(ret,lag.max=20,ylab="Sample Partial Autocorrelation",main=NA,ylim=c(-0.3.0.3),lwd=2)
>print(cbind(autopcorr$lag, autopcorr$acf))
```

2.

Fig 7

```
# arima202 predict

>fit202 = arima(ret, order = c(2, 0, 2))
>crpre = predict(fit202, n.ahead = 30)

>dates = seq(as.Date("02/08/2014", format = "%d/%m/%Y"), by = "days", length = length(ret))

>plot(ret, type = "l", xlim = c(0, 644), ylab = "log return", xlab = "days", lwd = 1.5)
>lines(crpre$pred, col = "red", lwd = 3)
>lines(crpre$pred + 2 * crpre$se, col = "red", lty = 3, lwd = 3)
>lines(crpre$pred - 2 * crpre$se, col = "red", lty = 3, lwd = 3)
```

3.

Fig 8

```
# install and load packages
>libraries = c("FinTS", "tseries")
>lapply(libraries, function(x) if (!(x %in% installed.packages())) { install.packages(x) })
>lapply(libraries, library, quietly = TRUE, character.only = TRUE)

>setwd()

>load(file = "crix.RData")
>Pr = as.numeric(crix)
>Da = factor(date1)
>crx = data.frame(Da, Pr)

# plot of crix return
>ret = diff(log(crx$Pr))
>Dare = factor(date1[-1])
>retts = data.frame(Dare, ret)
```