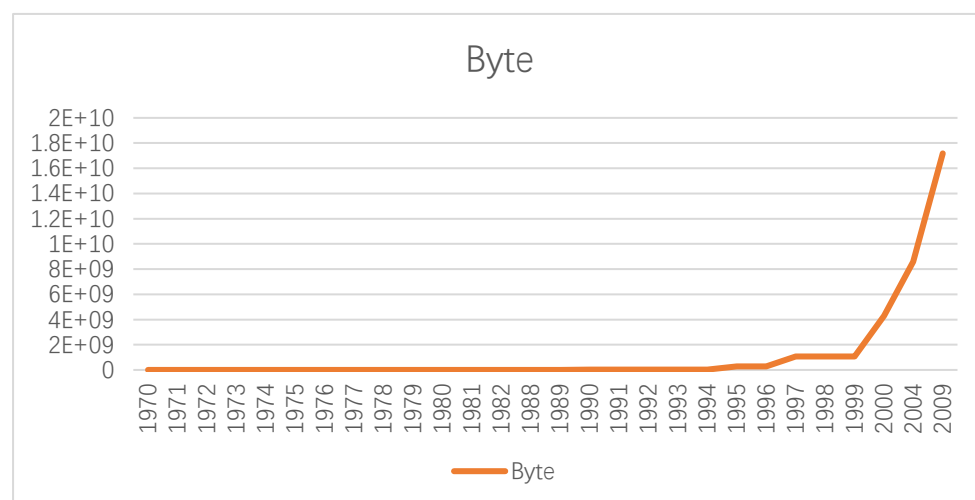


## HW Unit 1

Q1. Calculate the **increase** of memory of PCs over the last 30 years and check whether the

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
1990	2097152
1991	16777216
1992	16777216
1993	16777216
1994	16777216
1995	16777216
1996	2.68E+08
1997	2.68E+08
1998	1.07E+09
1999	1.07E+09
2000	1.07E+09
2004	4.29E+09
2009	8.59E+09
2014	1.72E+10

FMRI analysis could have been done 20 years ago.



2. prepare 2-5 slides explaining logistic regression

## Logistic regression

Logistic regression is a regression model where the dependent variable (DV) is categorical, where a categorical variable is a variable that can take on one of a limited, and usually fixed, number of possible values,

## Logistic regression VS multiple linear regression

- There are many similarities between logistic regression and multiple linear regression, the biggest difference is that their dependent variables are different, and the others are almost the same.
- If the dependent variable is continuous, that is, multiple linear regression, if it is the two distribution, that is the logistic regression.

## Application

- To explore the risk factors of a disease and predict the probability of a disease according to the risk factors.
- If we have established the logistic regression model, we can predict the probability of a disease or a certain situation under different independent variables according to the model.

3. I have done that and my account is Bingren Sun026.

### HW Unit 2

1. `memory.df = read.csv("byte.csv",header = TRUE)`

`plot(memory.df$Byte~memory.df$year,type="o",main="The development of internal memory")`

2. `splines.reg.l1 = smooth.spline(x = memory.df$year, y = memory.df$Byte, spar = 0.2)`

`splines.reg.l2 = smooth.spline(x = memory.df$year, y = memory.df$Byte, spar = 1)`

`splines.reg.l3= smooth.spline(x = memory.df$year, y = memory.df$Byte, spar = 2)`

`lines(splines.reg.l1, col = "green", lwd = 2)`

`lines(splines.reg.l2, col = "pink", lwd = 2)`

`lines(splines.reg.l3, col = "blue", lwd = 2)`

3. `lambda=4`

`x=6`

`dpois(x,lambda)`

`lambda=5`

`x=0`

`dpois(x,lambda)`

## HW Unit 3

1. `#install.packages("digest",repos='http://cran.us.r-project.org')`

`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")`

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



# Digital Signature Algorithms

DSA-Digital Signature Algorithm is a variant of Schnorr and ElGamal signature algorithms, and it is DSS (Digital Signature Standard) by NIST in the United States

## Digital Signature



Digital signature is a common physical signature similar to that written on paper, but it is implemented in the field of public key cryptography, and is used to identify digital information.



Digital signature is a digital string which can not be forged by others only. It is also an effective proof of the authenticity of information sent by the sender of the information.



A set of digital signatures usually defines two complementary operations, one for signature and the other for verification.



## Digital Signature Algorithms



- ✓ DSA is based on RSA algorithm.
- ✓ The private key  $X$  and the public key  $y$  are called a pair of keys  $(x, y)$ . The private key can only be held solely by the signer himself, and the public key can be published publicly. Key pairs can be used continuously over a period of time.
- ✓ It follows the principle of "to signature with the private key and to verify with the public key"



## Function of DSA



The integrity of the digital signature file is easy to verify, and the digital signature has non repudiation

Guarantee the integrity of information transmission, the identity authentication of sender, and the repudiation of transactions

Digital signature is an encryption process, and digital signature verification is a decryption process.

3. `library(rjson)`

`library(readr)`

`data <- read_csv("~/R/Home-Work-for-BDIF/data.csv")`

`json_ratedata <- toJSON(data, method = "C")`

`data`

`json_ratedata`

# JSON Data Generation

INITIAL DATA: DATA.CSV

T	rate
0.25	3.495
0.5	3.4836
0.75	3.5101
1	3.5232
2	3.6204
3	3.6833
4	3.7906
5	3.8189
7	3.9
10	3.97

JSON DATA

```
{\"T\": [0.25, 0.5, 0.75, 1, 2, 3, 4, 5, 7, 10],  
  \"rate\": [3.495, 3.4836, 3.5101, 3.5232, 3.6204, 3.6833, 3.7906, 3.8189, 3.9, 3.97]}
```

# JSON Data Generation

R CODE:

```
library(rjson)  
  
library(readr)  
  
data <- read_csv("~/R/Home-Work-for-BDIF/data.csv")  
  
json_ratedata <- toJSON(data, method = "C")  
  
data  
  
json_ratedata
```

4. `#install.packages("rjson", repos="http://cran.us.r-project.org")`

```
library(rjson)
```

```
json_file = "http://crix.hu-berlin.de/data/crix.json"
```

```
json_data = fromJSON(file=json_file)
```

```
crix <- Reduce(rbind,json_data)
```

```
crix_data_frame <- as.data.frame(crix)
```

```
lst <- lapply(json_data,function(x){
```

```
df<-data.frame(date=x$date,price=x$price)
```

```

return(df)
  })

crix_data_frame <- Reduce(rbind,lst)

plot(crix_data_frame$date,crix_data_frame$price)

#library(forecast)

#library(tseries)

plot(crix_data_frame)

```

HW Unit 4



## Q1

```

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)
w=crix_data_frame
dim(w)
n=dim(w)
a=seq(1,n[2],2)
b=seq(2,n[2],2)
data=t(w[1,a])
price=t(w[1,b])
#figure3
ts.plot(price)
#figure4
ret=diff(log(price))
ts.plot(ret)

```

```

#figure5
hist(ret, col = "grey", breaks = 40, freq = FALSE)
lines(density(ret), lwd = 2)
par(mfrow = c(1, 2))
# 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 = "red",
      lwd = 2)
# qq-plot
qqnorm(ret)
qqline(ret, col = "blue", lwd = 3)
#figure6
library(forecast)
library(tseries)
Acf(ret)

```

## Q1-Figures

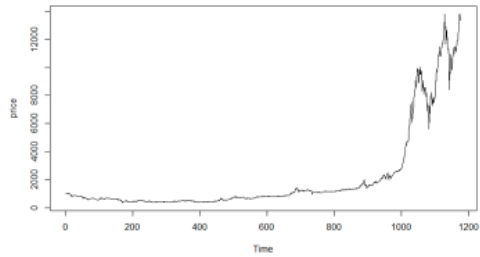


Figure 3

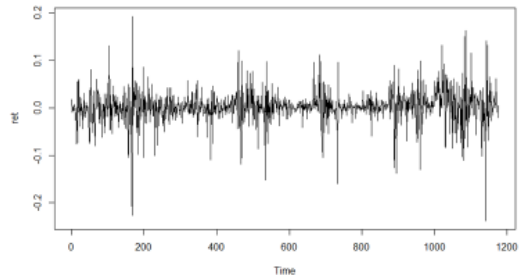


Figure 4



## Q1-Figures

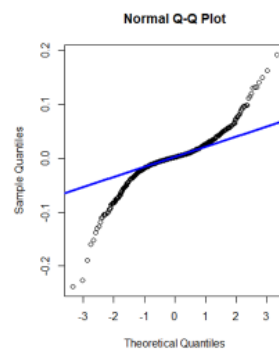
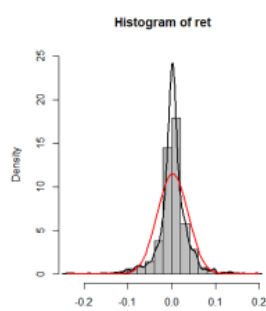


Figure 5

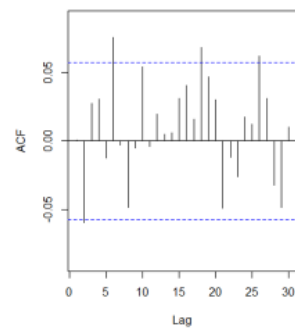


Figure 6





## Q2

```
# install and load packages
libraries = c("zoo", "tseries")
lapply(libraries, function(x) if (!x %in%
installed.packages())) {
  install.packages(x)
})
lapply(libraries, library, quietly = TRUE,
character.only = TRUE)
# d order
Box.test(ret, type = "Ljung-Box", lag = 20)
# stationary test
adf.test(ret, alternative = "stationary")
kpss.test(ret, null = "Trend")
par(mfrow = c(1, 2))
```

```
# acf plot
autocorr = acf(ret, lag.max = 20, ylab = "Sample
Autocorrelation", main = NA,
lwd = 2, ylim = c(-0.3, 1))
# LB test of linear dependence
print(cbind(autocorr$lag, autocorr$acf))
Box.test(ret, type = "Ljung-Box", lag = 1, fitdf = 0)
Box.test(autocorr$acf, type = "Ljung-Box")
# plot of pacf
autopcorr = pacf(ret, lag.max = 20, ylab = "Sample
Partial Autocorrelation",
main = NA, ylim = c(-0.3, 0.3), lwd = 2)
print(cbind(autopcorr$lag, autopcorr$acf))
```

## Q2

```
# arima model
par(mfrow = c(1, 1))
auto.arima(ret)
fit1 = arima(ret, order = c(1, 0, 1))
tsdiag(fit1)
Box.test(fit1$residuals, lag = 1)
# aic
aic = matrix(NA, 6, 6)
for (p in 0:4) {
  for (q in 0:3) {
    a.p.q = arima(ret, order = c(p, 0, q))
    aic.p.q = a.p.q$aic
    aic[p + 1, q + 1] = aic.p.q
  }
}
aic
```

```
# bic
bic = matrix(NA, 6, 6)
for (p in 0:4) {
  for (q in 0:3) {
    b.p.q = arima(ret, order = c(p, 0, q))
    bic.p.q = AIC(b.p.q, k = log(length(ret)))
    bic[p + 1, q + 1] = bic.p.q
  }
}
bic
# select p and q order of ARIMA model
fit4 = arima(ret, order = c(2, 0, 3))
tsdiag(fit4)
Box.test(fit4$residuals, lag = 1)

fitr4 = arima(ret, order = c(2, 1, 3))
tsdiag(fitr4)
Box.test(fitr4$residuals, lag = 1)
```

## Q2

```
# to conclude, 202 is better than 213
fit202 = arima(ret, order = c(2, 0, 2))
tsdiag(fit202)
tsdiag(fit4)
tsdiag(fitr4)
```

```
AIC(fit202, k = log(length(ret)))
AIC(fit4, k = log(length(ret)))
AIC(fitr4, k = log(length(ret)))
fit202$aic
fit4$aic
fitr4$aic
```

```
# 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, 1200), 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)
```

### Q2-figure

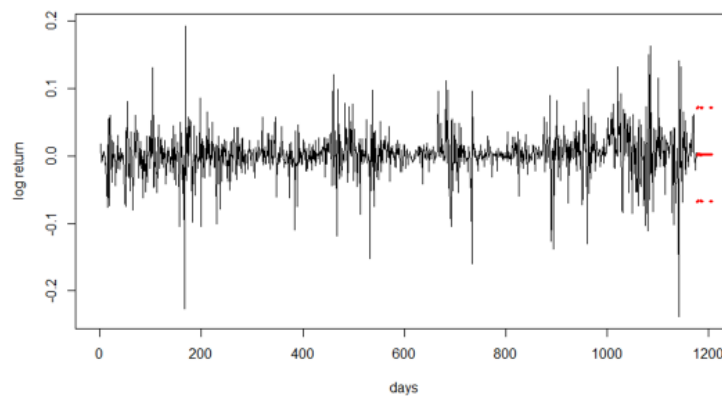
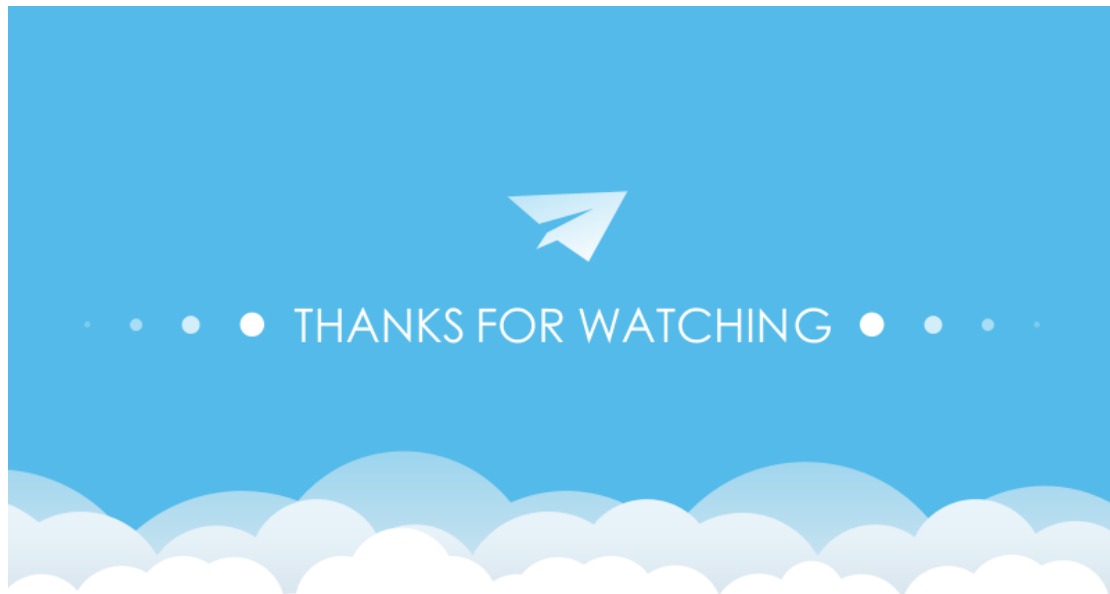


Figure 7



#### Question1

```
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)

w=crix_data_frame

dim(w)

n=dim(w)

a=seq(1,n[2],2)

b=seq(2,n[2],2)

data=t(w[1,a])

price=t(w[1,b])

#figure3

ts.plot(price)

#figure4

ret=diff(log(price))

ts.plot(ret)

#figure5

hist(ret, col = "grey", breaks = 40, freq = FALSE)
```

```

lines(density(ret), lwd = 2)

par(mfrow = c(1, 2))

# 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 = "red",
      lwd = 2)

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

#figure6

library(forecast)

library(tseries)

Acf(ret)

```

## Question2

```

#rm(list = ls(all = TRUE))

#graphics.off()

# install and load packages

libraries = c("zoo", "tseries")

lapply(libraries, function(x) if (!(x %in% installed.packages())) {
  install.packages(x)
})

lapply(libraries, library, quietly = TRUE, character.only = TRUE)

# d order

Box.test(ret, type = "Ljung-Box", lag = 20)

# stationary test

```

```

adf.test(ret, alternative = "stationary")

kpss.test(ret, null = "Trend")

par(mfrow = c(1, 2))

# acf plot

autocorr = acf(ret, lag.max = 20, ylab = "Sample Autocorrelation", main = NA, lwd = 2,
ylim = c(-0.3, 1))

# LB test of linear dependence

print(cbind(autocorr$lag, autocorr$acf))

Box.test(ret, type = "Ljung-Box", lag = 1, fitdf = 0)

Box.test(autocorr$acf, type = "Ljung-Box")

# 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))

# arima model

par(mfrow = c(1, 1))

auto.arima(ret)

fit1 = arima(ret, order = c(1, 0, 1))

tsdiag(fit1)

Box.test(fit1$residuals, lag = 1)

# aic

aic = matrix(NA, 6, 6)

for (p in 0:4) {
  for (q in 0:3) {
    a.p.q = arima(ret, order = c(p, 0, q))
    aic.p.q = a.p.q$aic
    aic[p + 1, q + 1] = aic.p.q
  }
}

aic

```

```

# bic
bic = matrix(NA, 6, 6)
for (p in 0:4) {
  for (q in 0:3) {
    b.p.q = arima(ret, order = c(p, 0, q))
    bic.p.q = AIC(b.p.q, k = log(length(ret)))
    bic[p + 1, q + 1] = bic.p.q
  }
}

bic

# select p and q order of ARIMA model
fit4 = arima(ret, order = c(2, 0, 3))
tsdiag(fit4)
Box.test(fit4$residuals, lag = 1)

fitr4 = arima(ret, order = c(2, 1, 3))
tsdiag(fitr4)
Box.test(fitr4$residuals, lag = 1)

# to conclude, 202 is better than 213
fit202 = arima(ret, order = c(2, 0, 2))
tsdiag(fit202)
tsdiag(fit4)
tsdiag(fitr4)

AIC(fit202, k = log(length(ret)))
AIC(fit4, k = log(length(ret)))
AIC(fitr4, k = log(length(ret)))

fit202$aic
fit4$aic
fitr4$aic

```

```
# 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, 1200), 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)
```