Big Data Homework

Lichao Zheng
27720161153035

HW1

HW1-1 Memory of personal computer

- **First stage:**
- 1982, Compaq created the first IBM laptop, the memory is 128KB RAM
- Taiwan promoted 72pin SO DIMM laptop
- In Pentium MMX period, 144pin 3.3V EDO **SO DIMM** appeared,
- **Second stage:**
- Synchronous Dynamic Random Access Memory(**SDRAM**), standardized 144pin
- **Third stage:**
- Double Data Rate(**DDR**): 1GB

HW1-2 Logistic Regression

- A regression model where the dependent variable is categorical.
- Logistic regression was developed by statistician David Cox in 1958.
- The binary logistic model is used to estimate the probability of a binary response based on one or more predictor (or independent) variables (features). It allows one to say that the presence of a risk factor increases the odds of a given outcome by a specific factor

HW1-2 Logistic Regression

- Logistic regression can be used in various fields, including machine learning, most medical fields, and social sciences.
- Compared with multiple linear regression, they all belong to generalized linear model, but they have different dependent variables: if DV is continuous, then it’s multiple linear regression; if DV is binomial distribution, then it’s logistic regression.

HW1-2 Example:

- Probability of passing an exam versus hours of study
- A group of 20 students spend between 0 and 6 hours studying for an exam. How does the number of hours spent studying affect the probability that the student will pass the exam?
- The dependent variable pass/fail represented by "1" and "0" are not cardinal numbers

HW2

HW 2

HW 2-1 & 2-2

- year<-c(1988,1991,1996,2000,2003,2007,2014)
- RAM<c(0.002,0.004,0.5,1,2,8,16)
- plot(year,RAM)
- lines(spline(year,RAM))
- lines(spline(year,RAM, n = 201), col = 2)

HW 2-3

- x = 6
- n = 1000
- lambda = 2
- p = lambda / n
- dbinom (x,2*n,p) # binomial probability mass function
- dpois (x, 2*lambda) # Poisson probability mass function
- dpois (0, 5)

HW3

HW3-1

#HW3-1#

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")

HW3-2

- Digital Signature Algorithms

The Digital Signature Algorithm (DSA) is a Federal Information Processing Standard for digital signatures.

For the key generation, it has two phases. The first phase is a choice of algorithm parameters which may be shared between different users of the system, while the second phase computes public and private keys for a single user.

For the Parameter generation, the steps are: Choose an approved cryptographic hash function *H*; Decide on a key length L and N which is the primary measure of the cryptographic strength of the key; Choose an N-bit prime q; Choose an L-bit prime p such that p – 1 is a multiple of q; Choose g, a number whose multiplicative order modulo p is q.

The algorithm parameters (p, q, g) may be shared between different users of the system.

Per-user keys: Given a set of parameters, the second phase computes private and public keys for a single user.

Apart from these, we also need signing and verifying process, then check the Correctness of the algorithm

HW3-3

```
# Create a JSON data set#
install.packages(“rjson”)
library(rjson)
Num <-c(1:5)
Name <-c(“Jack”,“Bob”,“Jobbs”,“Dell”,“apple”)
data <-as.matrix(data.frame(Num,Name))
cat(toJSON(data))
# Read the JSON data set #
library(“rjson”)
json_data = fromJSON(file=data)
```

HW3-4

```
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_data_frame = as.data.frame(json_data)
a<-seq(1,2348,2)
b<-seq(2,2348,2)
date<-t(crix_data_frame[1,a])
price<-t(crix_data_frame[1,b])
```

#to be continued

```
return<-1:1174
for(i in 1:1174)
{return[i+1]<-log(price[i+1]/price[i])}
new<-data.frame(date,price,return[1:1174])
names(new)<-c("date","price","return")
plot(new$date,new$return)
arima(new$return, order = c(2,0,1))
#These results suggest that the CRIX return series can be modeled by some ARIMA process,
 for example ARIMA(2, 0, 2).
library(timeDate)
library(timeSeries)
library(fBasics)
library(fGarch)
garchFit(new$return ~ garch(1, 1))
```

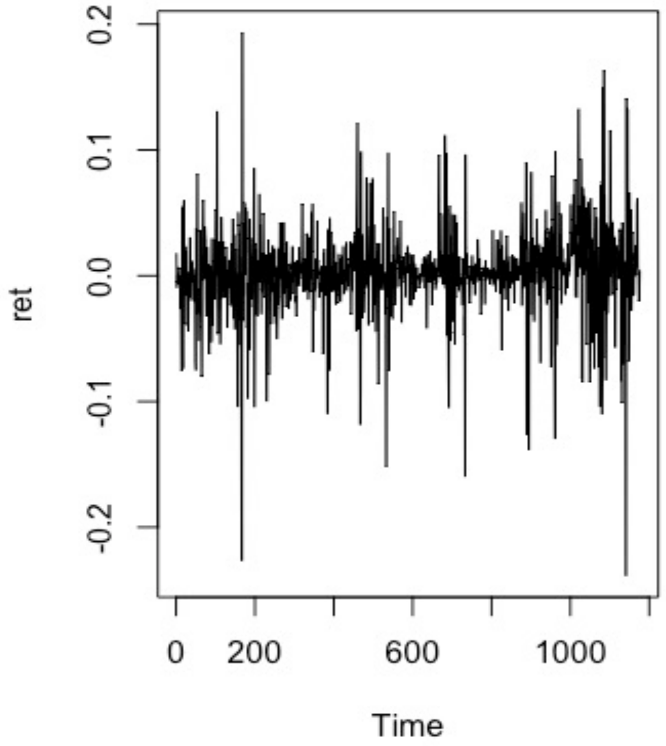
HW4

Lichao Zheng

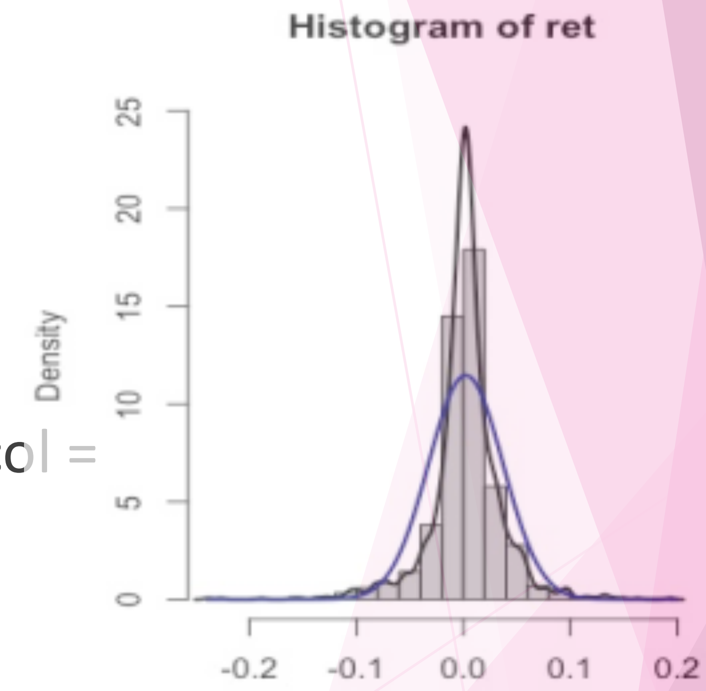
27720161153035

```r
#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_data_frame = as.data.frame(json_data)
n<-dim(crix_data_frame)
a<-seq(1,n[2],2)
b<-seq(2,n[2],2)
date<-t(crix_data_frame[1,a])
price<-t(crix_data_frame[1,b])


ts.plot(price)
ret<-diff(log(price))
plot(ret)
ts.plot(ret)
```



```
# 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)

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

# plot of pacf
autopcorr = pacf(ret, lag.max = 20, ylab = "Sample Partial
Autocorrelation", main = NA, ylim = c(-0.3, 0.3), lwd = 2)
```

```
# 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)
```

```
# 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)
```