Final Exam

Li Zhang

27720161153031

Homework 1

Q1 Computer memory

```
year<-
c(1971,1972,1974,1976,1978,1979,1982,1985,1988,1989,1990,
1991,1992,1993,

1994,1995,1996,1997,1998,1999,2000,2001,2002,2003,2004,20
05,2006,2007,2008)
number<-
c(1,1,1,1,1,1,2,1,1,1,1,1,4,1,3,3,2,5,12,21,22,27,44,31,27,16,29,
44,59)
plot(year,number,type = "b",
        col="black",main = "The history of computer memory",
        sub = "This is the change in the number of types of
computer memory",
        xlab = "year",ylab = "The number of memory")
barplot(number,
             xlab = "year",ylab ="The number of memory ")
```

Q2 logistic regression

Logistic regression was developed in 1958. The binary logistic model is used to estimate the probability of a binary response based on one or more predictor variables. It predicts that the presence of a risk factor increases the probability of a given outcome by a specific factor.

Logistic regression can be used in various fields, including medical fields, and social sciences.

Compared with multiple linear regression, they both belong to generalized linear model, but they have different features. Mainly for the dependent variables: if dependent variable is continuous, then it is multiple linear regression. If dependent variable is binomial distribution, then it's logistic regression.

Homework 2

Q1

```
year<-
c(1971,1972,1974,1976,1978,1979,1982,1985,1988,1989,1990,
1991,1992,1993,

1994,1995,1996,1997,1998,1999,2000,2001,2002,2003,2004,20
05,2006,2007,2008)
number<-
c(1,1,1,1,1,1,2,1,1,1,1,1,4,1,3,3,2,5,12,21,22,27,44,31,27,16,29,
44,59)
```

Q2

```
plot(year,number,type = "b",
        col="black",main = "The history of computer memory",
        sub = "This is the change in the number of types of
computer memory",
        xlab = "year",ylab = "The number of memory")
barplot(number,
              xlab = "year",ylab ="The number of memory ")
```

Q3

```
lambda=2
x=seq(0:6)
P<-data.frame(dpois(x,lambda))
sum<-(P[1,]+P[7,]+P[2,]+P[6,]+P[3,]+P[5,]+P[4,]+P[4,])
sum
lambda=5
```

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

Homework 3

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")
```

Q2

What is DSA

- Digital signatures are essential to verify the sender of a document's identity. The signature is computer using a set of rules and algorithm such that the identity of the person can be verified.
- The signature is generated by the use of a private key that known only to the user. The signature is verified when a public key is corresponds to the private key. With every user having a public/private key pair, this is an example of public-key cryptography.

•Public keys, which are known by everyone, can be used to verify the signature of a user. The private key, which is never shared, is used in signature generation, which can only be done by the user.

Function of DSA

•Digital signatures are used to detect unauthorized modifications to data. Also, the recipient of a digitally signed document in proving to a third party that the document was indeed signed by the person who it is claimed to be signed by. This is known as nonrepudiation, because the person who signed the document cannot repudiate the signature at a later time.

•Digital signature algorithms can be used in e-mails, electronic funds transfer, electronic data interchange, software distribution, data storage, and just about any application that would need to assure the integrity and originality of data.

1st part of DSA

•Choose a prime number q, which is called the prime divisor.

•Choose another primer number p, such that p-1 mod q = 0. p is called the prime modulus.

•Choose an integer g, such that 1 < g < p, g**q mod p = 1 and g = h**((p–1)/q) mod p. q is also called g's multiplicative order modulo p.

•Choose an integer, such that 0 < x < q.

•Compute y as g**x mod p.

•Package the public key as {p,q,g,y}.

•Package the private key as {p,q,g,x}.

2nd part of DSA

•To generate a message signature, the sender can follow these steps:

•Generate the message digest h, using a hash algorithm like SHA1.

•Generate a random number k, such that 0 < k < q.

•Compute r as (g**k mod p) mod q. If r = 0, select a different k.

•Compute i, such that k*i mod q = 1. i is called the modular multiplicative inverse of k modulo q.

•Compute s = i*(h+r*x) mod q. If s = 0, select a different k.

•Package the digital signature as {r,s}.

Steps for verifying

•Generate the message digest h, using the same hash algorithm.

•Compute w, such that s*w mod q = 1. w is called the modular multiplicative inverse of s modulo q.

•Compute u1 = h*w mod q.

•Compute u2 = r*w mod q.

•Compute v = (((g**u1)*(y**u2)) mod p) mod q.

•If v == r, the digital signature is valid.

Q3

R-code:

```
>library(RJSONIO)
> letter<-LETTERS[1:10]
>country<-c("China","the US","the UK","Russia",
        "Korea","Japan","Italy","Brazil","India","Germany")
> data<-data.frame(letter,country)
> da<-as.matrix(data)
>cat(toJSON(da))
[ { "letter": "A","country": "China" },
{"letter": "B","country": "the US" },
{"letter": "C","country": "the UK" },
{ "letter": "D","country": "Russia" },
{"letter": "E","country": "Korea" },
{"letter": "F","country": "Japan" },
{"letter": "G","country": "Italy" },
{ "letter": "H","country": "Brazil" },
{"letter": "I","country": "India" },
{"letter": "J","country": "Germany" } ]
```

Q4

```
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)
# load dataset #
load(file = "C:/Users/xiumei/Desktop/big data/crix.RData")
ret = diff(log(crix))
# 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}}
# 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}}
```

```
# 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, 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)

# Produces GARCH estimation results using ARIMA model
residuals
rm(list = ls(all = TRUE))
graphics.off()

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

# load dataset
```

```
load(file = "C:/Users/xiumei/Desktop/big data/crix.RData")
ret = diff(log(crix1))

# vol cluster
fit202 = arima(ret, order = c(2, 0, 2))
par(mfrow = c(1, 1))
res = fit202$residuals
res2 = fit202$residuals^2

# different garch model
fg11 = garchFit(data = res, data ~ garch(1, 1))
summary(fg11)
fg12 = garchFit(data = res, data ~ garch(1, 2))
summary(fg12)
fg21 = garchFit(data = res, data ~ garch(2, 1))
summary(fg21)
fg22 = garchFit(data = res, data ~ garch(2, 2))
summary(fg22)

# residual plot
reszo = zoo(fg11@residuals, order.by = index(crix1))
plot(reszo, ylab = NA, lwd = 2)
```

```
par(mfrow = c(1, 2))
fg11res2 = fg11@residuals
acfres2    = acf(fg11res2, lag.max = 20, ylab = "Sample
Autocorrelation", main = NA, lwd = 2)
pacfres2 = pacf(fg11res2, lag.max = 20, ylab = "Sample Partial
Autocorrelation", main = NA, lwd = 2, ylim = c(-0.5, 0.5))
fg12res2 = fg12@residuals
acfres2 = acf(fg12res2, lag.max = 20, ylab = "Sample
Autocorrelation", main = NA, lwd = 2)
pacfres2 = pacf(fg12res2, lag.max = 20, ylab = "Sample Partial
Autocorrelation", main = NA, lwd = 2, ylim = c(-0.5, 0.5))

# qq plot
par(mfrow = c(1, 1))
plot(fg11, which = 13)    #9,10,11,13

# kp test
set.seed(100)
x = rnorm(200)

# Do x and y come from the same distribution?
```

```
ks.test(x, fg11@residuals)
```

Homework 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)
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 = 'black', 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 = ‘blue’, lwd = 2)

# qq-plot
qqnorm(ret)
qqline(ret, col = ‘grey’, 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$pred, col = ‘green’, lwd = 3)
lines(crpre$pred + 2 * crpre$se, col = ‘green’, lty = 3, lwd = 3)
lines(crpre$pred - 2 * crpre$se, col = ‘green’, lty = 3, lwd = 3)
```