Homework 3

Zhehao Yu

Q1 Code

```
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")
```

Digital Signature Algorithm

Zhehao Yu

1 Introduction

The Digital Signature Algorithm (DSA) is a Federal Information Processing Standard for digital signatures. In August 1991 the National Institute of Standards and Technology (NIST) proposed DSA for use in their Digital Signature Standard (DSS) and adopted it as FIPS 186 in 1993.Four revisions to the initial specification have been released: FIPS 186-1 in 1996, FIPS 186-2 in 2000, FIPS 186-3 in 2009, and FIPS 186-4 in 2013.

DSA is covered by U.S. Patent 5,231,668, filed July 26, 1991 and attributed to David W. Kravitz, a former NSA employee. This patent was given to "The United States of America as represented by the Secretary of Commerce, Washington, D.C.", and NIST has made this patent available worldwide royalty-free. Claus P. Schnorr claims that his U.S. Patent 4,995,082 (expired) covered DSA; this claim is disputed. DSA is a variant of the ElGamal signature scheme.

2 Common digital signature algorithm

- RSA-based signature schemes, such as RSA-PSS.
- DSA and its elliptic curve variant ECDSA.
- Edwards-curve Digital Signature Algorithm and its Ed25519 variant.
- ElGamal signature scheme as the predecessor to DSA, and variants Schnorr signature and Pointcheval–Stern signature algorithm.

3 Vertify

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

4 How to use it

Digital signatures are based on public key encryption, also known as asymmetric encryption. Using a public key algorithm such as RSA, you can generate two keys that are mathematically linked: a private and a public key. To create a digital signature, the signature software (such as an e-mail program) creates a one-way hash of the electronic data to be signed. Then the private key is used to encrypt the hash. The encrypted hash and other information such as the hash algorithm are digital signatures. The reason for encrypting a hash rather than an entire message or document is that the hash function can convert any input to a fixed-length value, which is usually much shorter. This saves time because the hash is much faster than the signature.

Jason data save and read in R again

Zhehao Yu

1 Code

```
install.packages('RJSONIO')
library(RJSONIO)
jobs<-c("scientist","engineer","teacher","doctor")
name<-c("JACK","BOB","MARY","JUSTIN")
data<-data.frame(name,jobs)
da<-as.matrix(data)
cat(toJSON(da))
```

2 Outcome


```
[ {
 "name": "JACK",
"jobs": "scientist"
},
{
 "name": "BOB",
"jobs": "engineer"
},
{
 "name": "MARY",
"jobs": "teacher"
},
{
 "name": "JUSTIN",
"jobs": "doctor"
} ]>
```

Q4 Code

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

#Arima
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")

#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)
#3)Parameter Estimation
#estimation of p and q
a.fin1=auto.arima(dr)
summary(a.fin1)
```

Q4 Code

```
#ARMA(0,0) therefore r fits ARIMA(0,1,0)
a.fin2=arima(r,order=c(0,1,0))
summary(a.fin2)
help("forecast.Arima")
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)

#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
```

Q4 Code

```
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.
```