Final Exam

Zhehao Yu

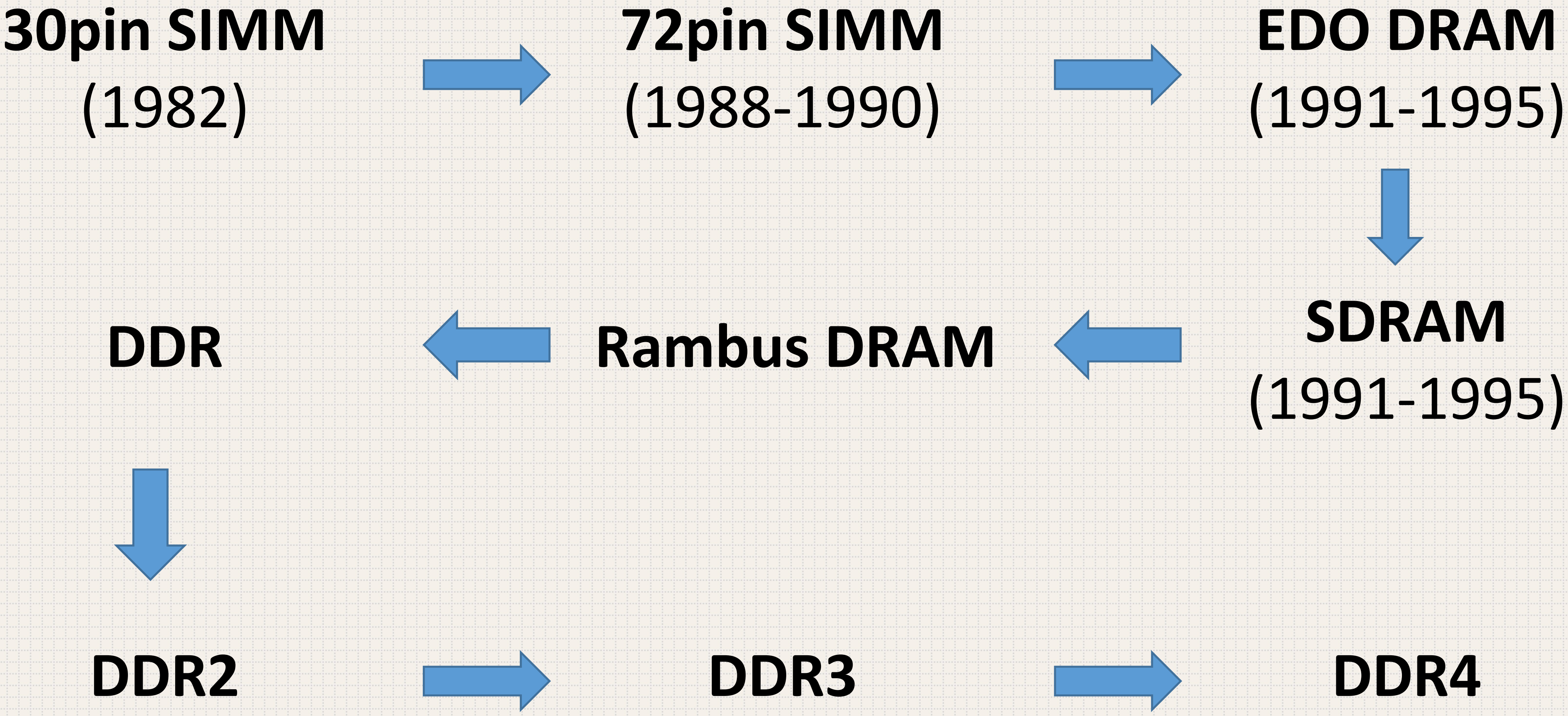
Homework 1

Zhehao Yu

The development of home computer memory

Zhehao Yu

1 The histoy of memory



30pin SIMM
(1982)

72pin SIMM
(1988-1990)

EDO DRAM
(1991-1995)

SDRAM
(1991-1995)

Rambus DRAM

DDR

DDR2

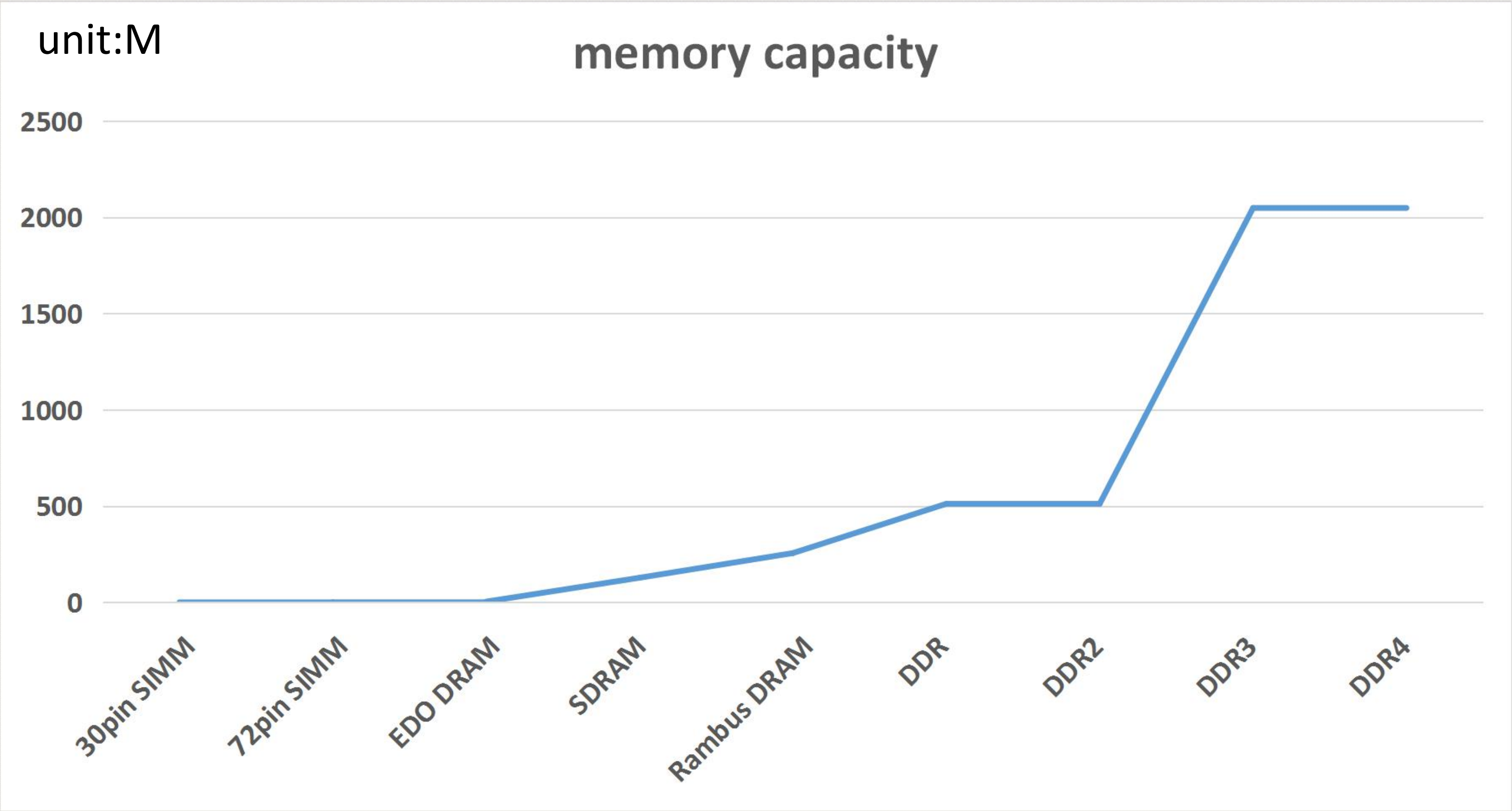
DDR3

DDR4

2 Main data

memory	30pin SIMM	72pin SIMM	EDO DRAM	SDRAM	Rambus DRAM	DDR	DDR2	DDR3	DDR4
memory capacity	256KB	512KB	4M	128M	256M	512M	512M	2GB	2GB
memory bandwitdth			32bit/s	64bit/s	1064M/s		3.2GB/s		

3 Figure


unit:M
memory capacity
2500
2000
1500
1000
500
0
30pin SIMM
72pin SIMM
EDO DRAM
SDRAM
Rambus DRAM
DDR
DDR2
DDR3
DDR4

Logistic regression

Zhehao Yu

1 Introduction of the Logistic regression

In statistics, logistic regression, or logit regression, or logit model is a regression model where the dependent variable (DV) is categorical. This article covers the case of a binary dependent variable—that is, where the output can take only two values, "0" and "1", which represent outcomes such as pass/fail, win/lose, alive/dead or healthy/sick. Cases where the dependent variable has more than two outcome categories may be analysed in multinomial logistic regression, or, if the multiple categories are ordered, in ordinal logistic regression.In the terminology of economics, logistic regression is an example of a qualitative response/discrete choice model.

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.

2 Fields and example applications

Logistic regression is used in various fields, including machine learning, most medical fields, and social sciences. For example, the Trauma and Injury Severity Score (TRISS), which is widely used to predict mortality in injured patients, was originally developed by Boyd et al. using logistic regression. Many other medical scales used to assess severity of a patient have been developed using logistic regression. 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.).

3 Fields and example applications

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. The technique can also be used in engineering, especially for predicting the probability of failure of a given process, system or product. It is also used in marketing applications such as prediction of a customer's propensity to purchase a product or halt a subscription, etc. In economics it can be used to predict the likelihood of a person's choosing to be in the labor force, and a business application would be to predict the likelihood of a homeowner defaulting on a mortgage. Conditional random fields, an extension of logistic regression to sequential data, are used in natural language processing.

Homework 2

Zhehao Yu

Code

Q1

```
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 )
```

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 ")
```

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.
```

Homework 4

Zhehao Yu

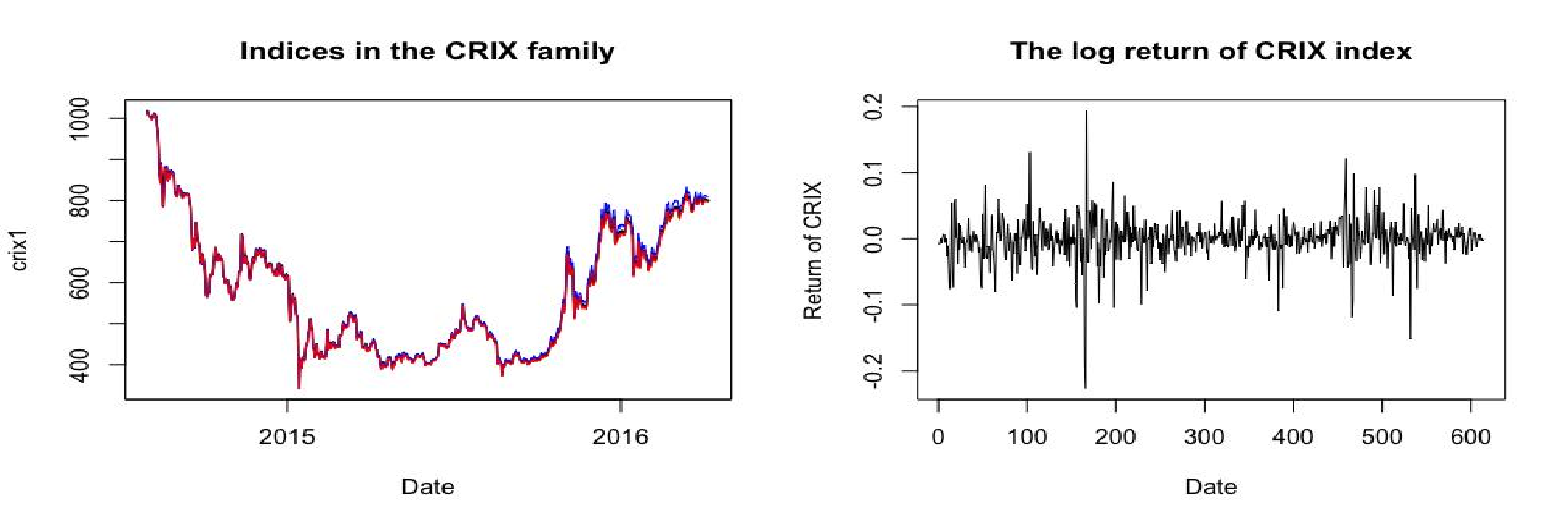
Q1 Code

```
rm(list = ls(all = TRUE))
graphics.off()

# install and load packages
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 = "C:/Users/Administrator/Desktop/crix.RData")
load(file = "C:/Users/Administrator/Desktop/ecrix.RData")
load(file = "C:/Users/Administrator/Desktop/efcrix.RData")
# three indices return
ecrix1 = zoo(ecrix, order.by = index(crix1))
efcrix1 = zoo(efcrix, order.by = index(crix1))

# plot with different x-axis scales with zoo
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", xlab =
"Date")
# plot of crix
# plot(as.xts(crix), type="l", auto.grid=FALSE, main =
NA)
plot(crix1, ylab = "Price of CRIX", xlab = "Date")

# plot of crix return
ret   = diff(log(crix1))
# plot(as.xts(ret), type="l", auto.grid=FALSE, main =
NA)
plot(ret, ylab = "Return of CRIX", xlab = "Date")

# stationary test
adf.test(ret, alternative = "stationary")
kpss.test(ret, null = "Trend")
```

Q1 Code

```
par(mfrow = c(1, 2))
# histogram of returns
hist(ret, col = "grey", breaks = 20, freq = FALSE, ylim = c(0, 25), xlab = "Return of CRIX")
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)

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

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

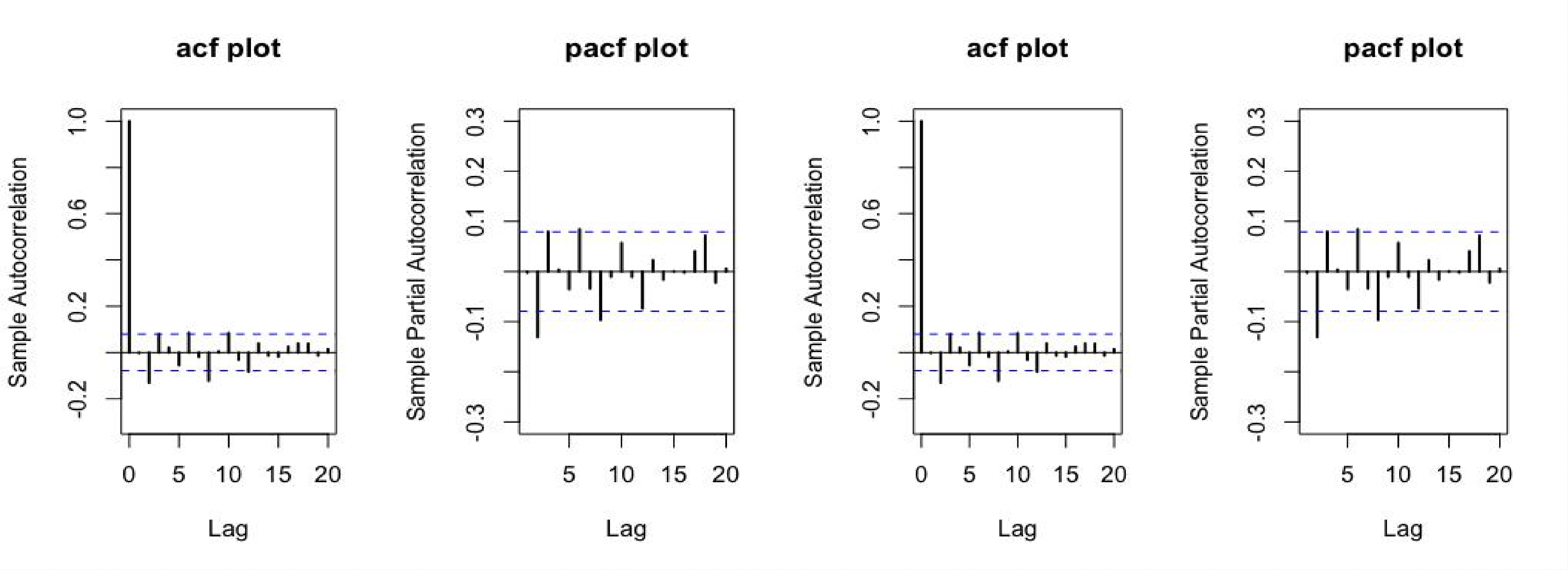
Q1 Figure



Indices in the CRIX family

The log return of CRIX index

Q1 Figure



acf plot
Sample Autocorrelation
1.0
0.6
0.2
-0.2
0
5
10
15
20
Lag
pacf plot
Sample Partial Autocorrelation
0.3
0.2
0.1
-0.1
-0.3
5
10
15
20
Lag
acf plot
Sample Autocorrelation
1.0
0.6
0.2
-0.2
0
5
10
15
20
Lag
pacf plot
Sample Partial Autocorrelation
0.3
0.2
0.1
-0.1
-0.3
5
10
15
20
Lag

Q2 Code

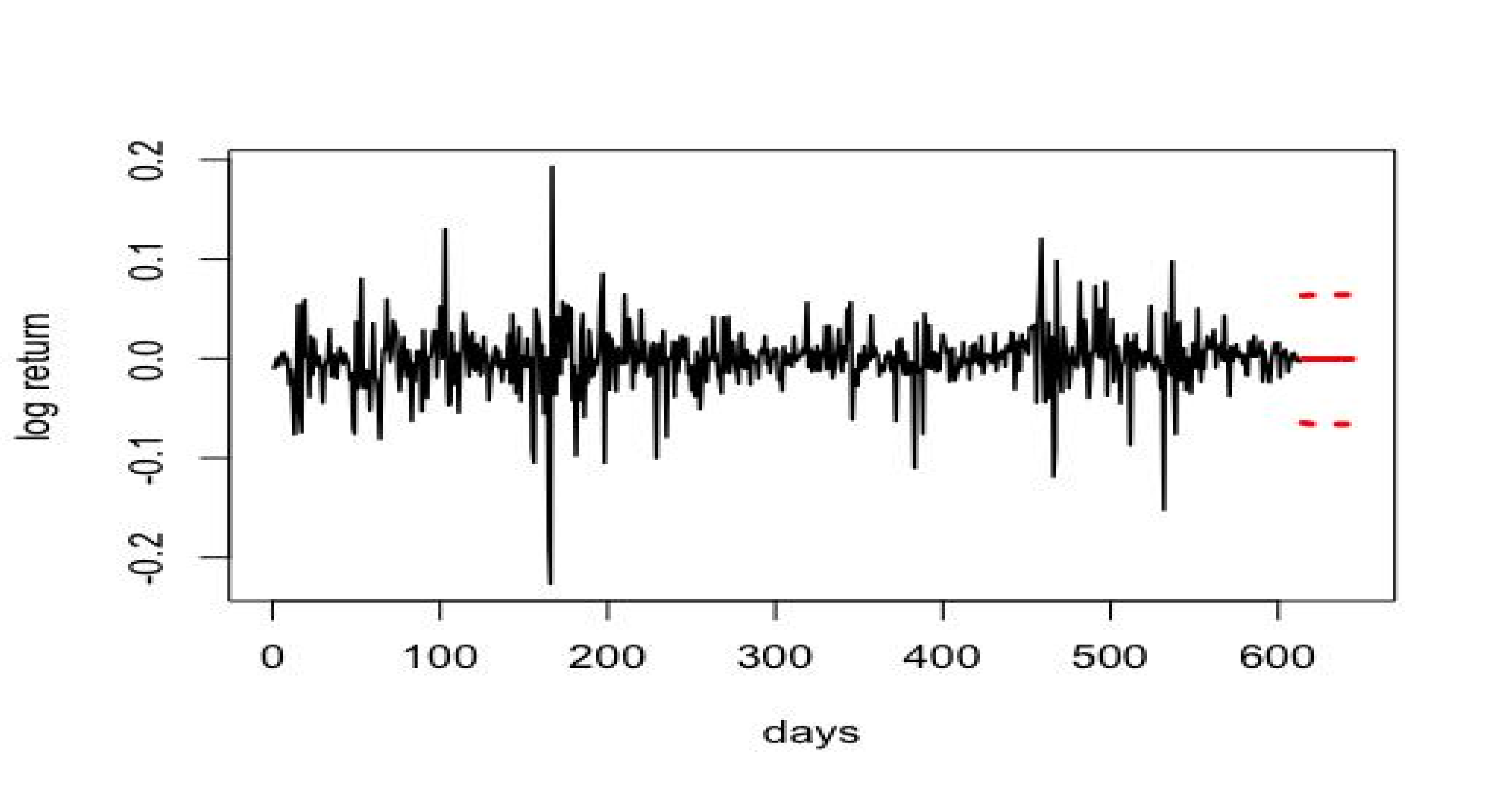
```
# arima model
par(mfrow = c(1, 1))
fit1 = arima(ret, order = c(1, 0, 1))
tsdiag(fit1)
Box.test(fit1$residuals, lag = 1)

# 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)
```

Q2 Code

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

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

dates = seq(as.Date("02/08/2014", format = "%d/%m/%Y"), by = "days", length = length(ret))
plot(ret, type = "l", xlim = c(0, length(ret)+predict_num), ylab = "log return", xlab = "days", lwd = 1.5, col = "black")
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 Code



Q3 Code

```
rm(list = ls(all = TRUE))
graphics.off()

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

# please change your working directory
setwd()
load(file.choose())
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)
# arima202 predict
fit202 = arima(ret, order = c(2, 0, 2))
# vola cluster
par(mfrow = c(1, 1))
res = fit202$residuals
res2 = fit202$residuals^2
tsres202 = data.frame(Dare, res2)
plot(tsres202$Dare, tsres202$res2, type = "o", ylab = NA)
lines(tsres202$res2)

# plot(res2, ylab='Squared residuals', main=NA)
par(mfrow = c(1, 2))
acfres2 = acf(res2, main = NA, lag.max = 20, ylab = "Sample Autocorrelation", lwd = 2)
pacfres2 = pacf(res2, lag.max = 20, ylab = "Sample Partial Autocorrelation", lwd = 2, main = NA)
```

Q3 Code

```
rm(list = ls(all = TRUE))
graphics.off()

# install and load packages
libraries = c("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.choose())
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)
```

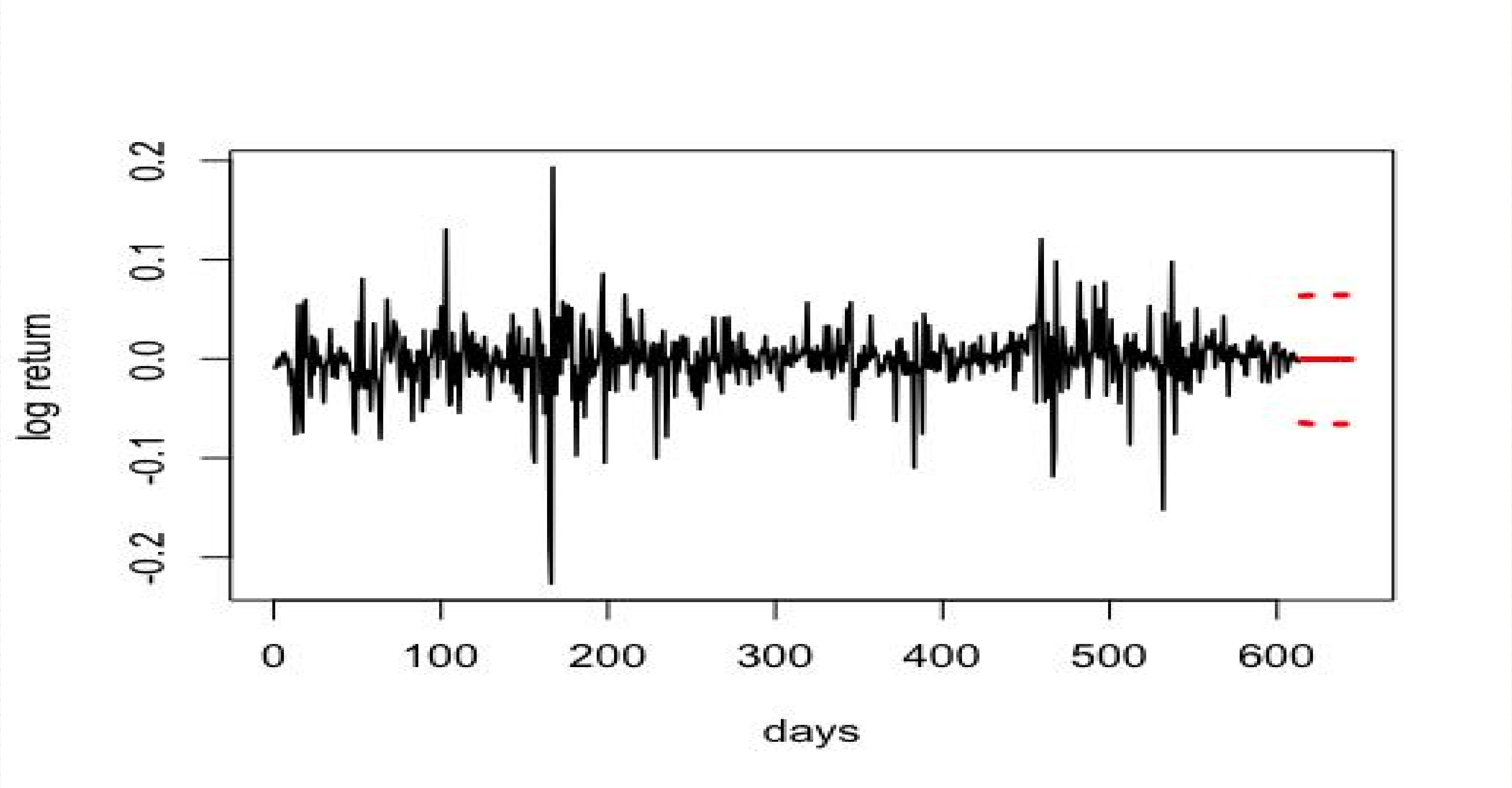
Q3 Code

```
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
```

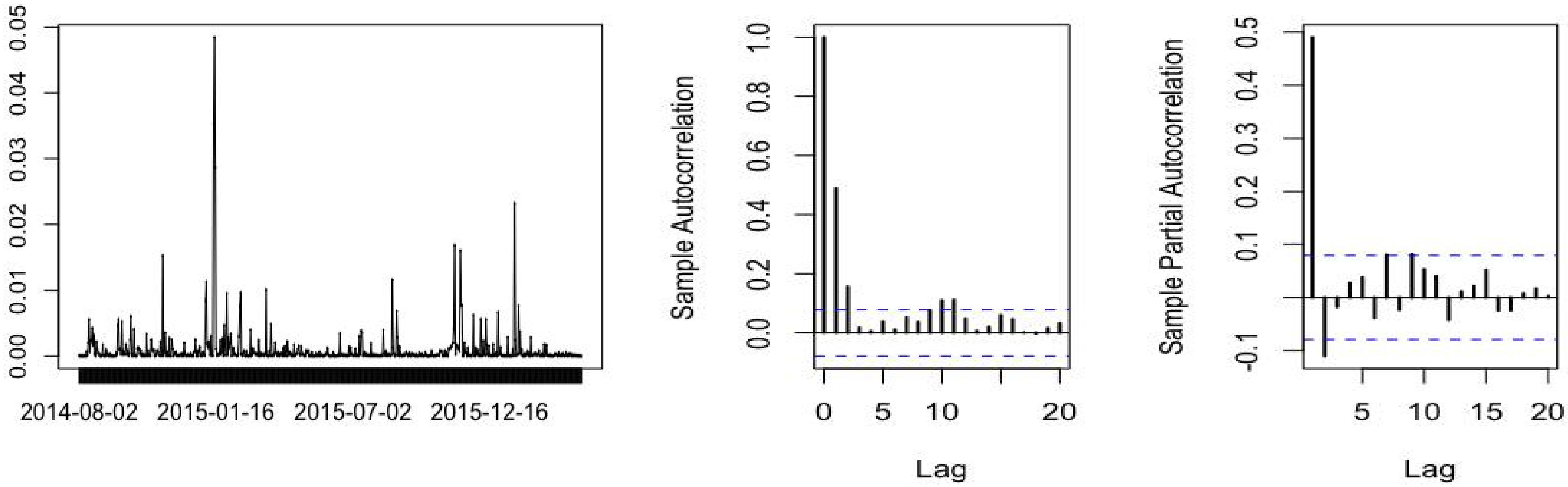
Q3 Code

```
fg11stu = garchFit(data = res, data ~ garch(1, 1), cond.dist = "std")

# different forecast with t-garch
# fg11stufore = predict(fg11stu, n.ahead = 30, plot=TRUE, mse='uncond', auto.grid=FALSE)
fg11stufore = predict(fg11stu, n.ahead = 30, plot = TRUE, cond.dist = "QMLE",
                      auto.grid = FALSE)

par(mfrow = c(1, 2))
stu.fg11res2 = fg11stu@residuals

# acf and pacf for t-garch
stu.acfres2 = acf(stu.fg11res2, ylab = NA, lag.max = 20, main = "ACF of Squared Residuals",
                  lwd = 2)
stu.pacfres2 = pacf(stu.fg11res2, lag.max = 20, main = "PACF of Squared Residuals",
                    lwd = 2, ylab = NA, ylim = c(-0.5, 0.5))

# ARIMA-t-GARCH qq plot
par(mfrow = c(1, 1))
plot(fg11stu, which = 13)
```

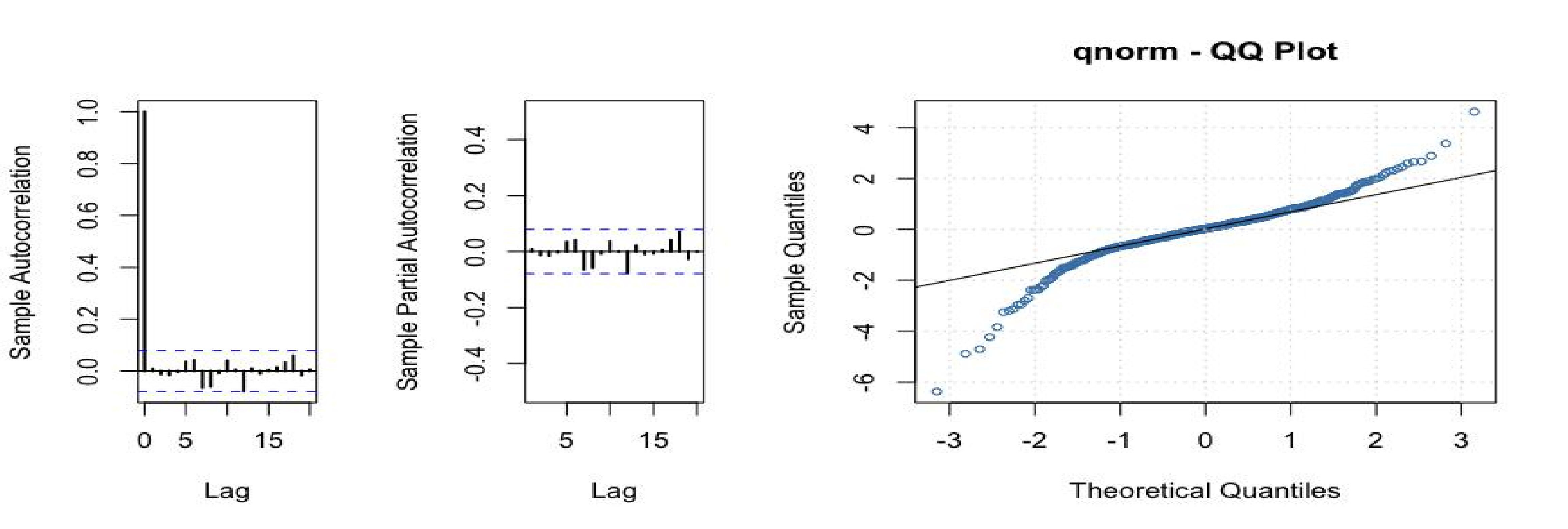
Q3 Figure



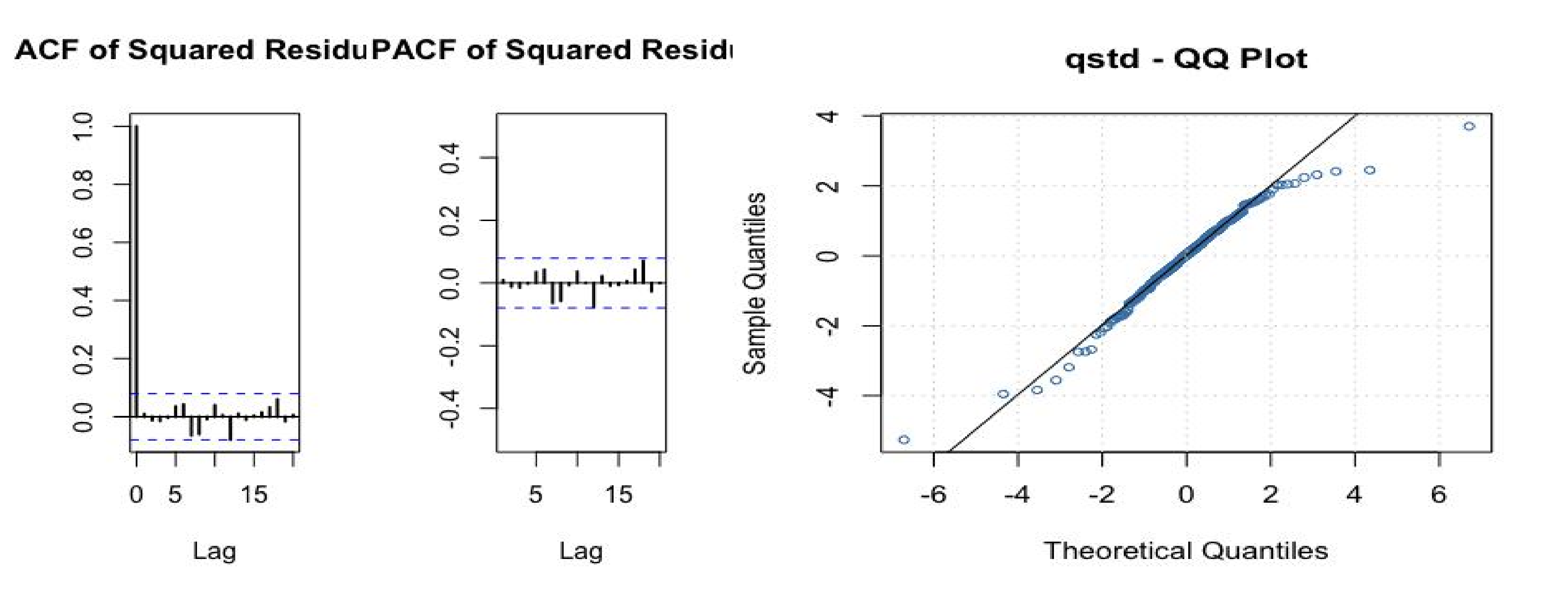
Q3 Figure


0.05
0.04
0.03
0.02
0.01
0.00
2014-08-02
2015-01-16
2015-07-02
2015-12-16
Sample Autocorrelation
1.0
0.8
0.6
0.4
0.2
0.0
0
5
10
20
Lag
Sample Partial Autocorrelation
0.5
0.4
0.3
0.2
0.1
-0.1
5
10
15
20
Lag

Q3 Figure


Sample Autocorrelation
1.0
0.8
0.6
0.4
0.2
0.0
0
5
15
Lag
Sample Partial Autocorrelation
0.4
0.2
0.0
-0.2
-0.4
5
15
Lag
qnorm - QQ Plot
Sample Quantiles
4
2
0
-2
-4
-6
-3
-2
-1
0
1
2
3
Theoretical Quantiles

Q3 Figure



ACF of Squared Residu PACF of Squared Resid

qstd - QQ Plot