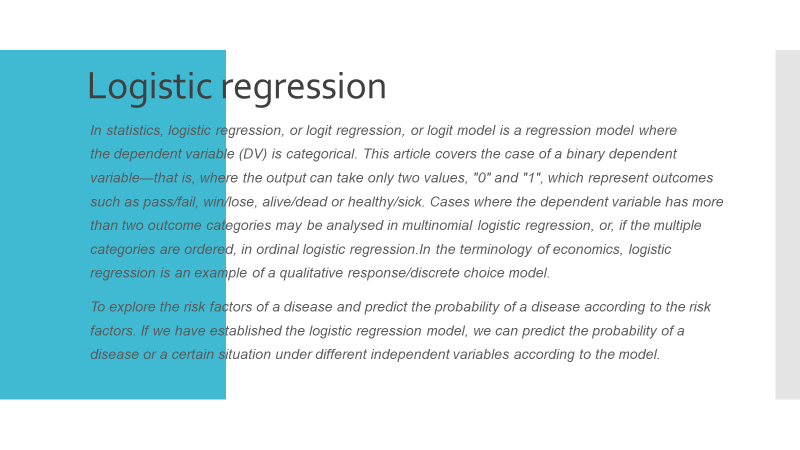
**Homework 1**

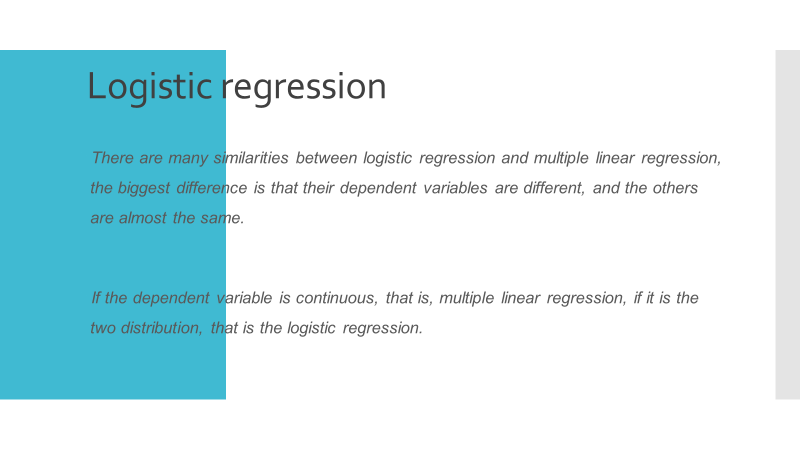
1. **Calculate the increase of memory of PCs over the last 30 years and check whether the FMRI analysis could have been done 20 years ago.**

|  |  |  |  |
| --- | --- | --- | --- |
| ***year*** | ***Byte*** | ***year*** | ***Byte*** |
| 1970 | 262144 | 1990 | 2097152 |
| 1971 | 262144 | 1991 | 16777216 |
| 1972 | 262144 | 1992 | 16777216 |
| 1973 | 262144 | 1993 | 16777216 |
| 1974 | 262144 | 1994 | 16777216 |
| 1975 | 262144 | 1995 | 16777216 |
| 1976 | 262144 | 1996 | 268435456 |
| 1977 | 262144 | 1997 | 268435456 |
| 1978 | 262144 | 1998 | 1073741824 |
| 1979 | 262144 | 1999 | 1073741824 |
| 1980 | 262144 | 2000 | 1073741824 |
| 1981 | 262144 | 2004 | 4294967296 |
| 1982 | 262144 | 2009 | 8589934592 |
| 1988 | 2097152 | 2014 | 17179869184 |
| 1989 | 2097152 |  |  |

1. **logistic regression**







1. **Github Account:**  <https://github.com/DongyuWang282/Home-Work-for-BDIF>

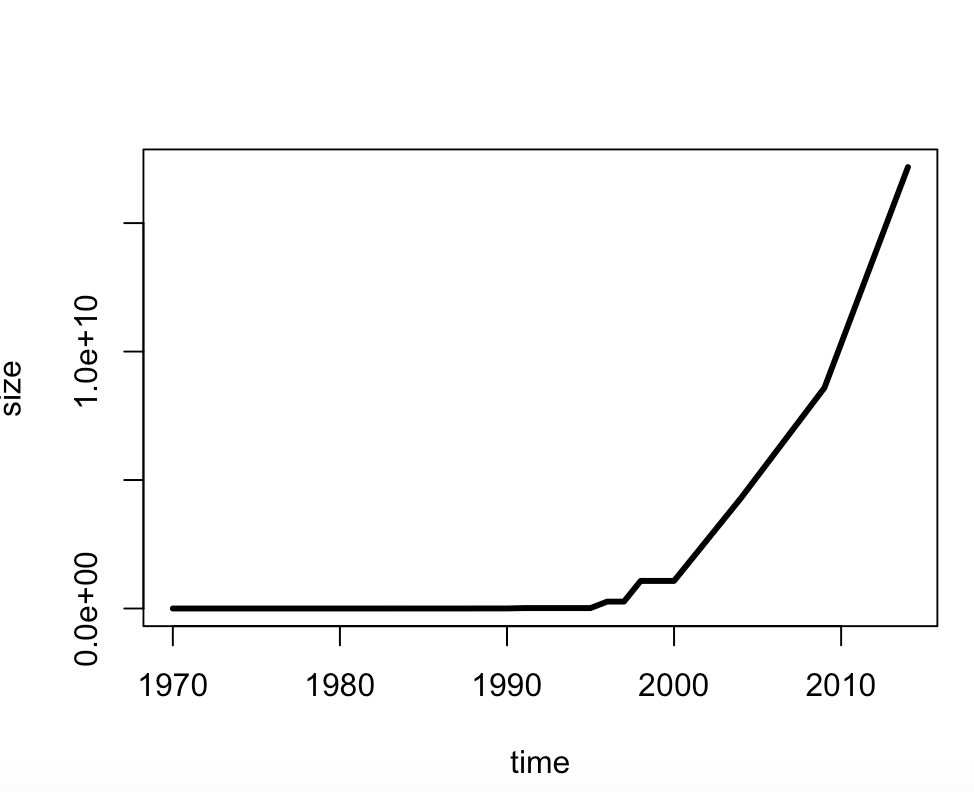
**Homework 2**

1. **Make an R Quantlet to solve HW #1 from unit 1 with R and show it on Github (GH). Hint: use the CMB Qs for this work.**

*library(readr)*

*RAM\_size <- read\_csv("~/R data/Home-Work-for-BDIF/RAM\_size.csv")*

*plot(RAM\_size,type="l",xlab = "time",ylab = "size",lwd=3)*

**

1. **use R with B-spline code to solve HW#1**

*splines.reg.l1 = smooth.spline(x = RAM\_size$year, y = RAM\_size$Byte, spar = 0.2) # lambda = 0.2*

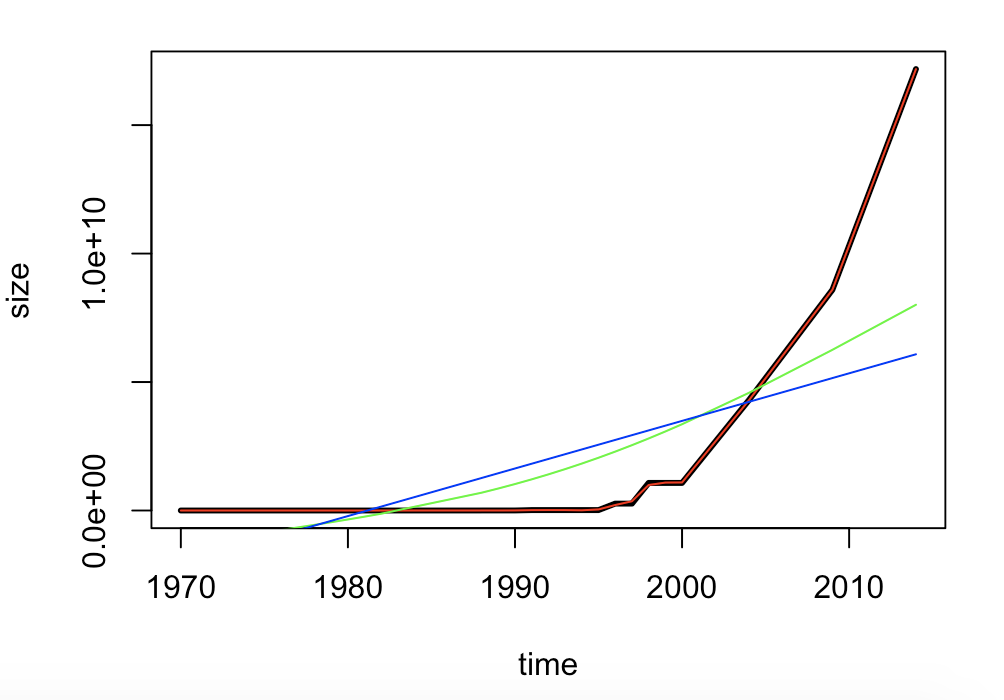
*splines.reg.l2 = smooth.spline(x = RAM\_size$year, y = RAM\_size$Byte, spar = 1) # lambda = 1*

*splines.reg.l3 = smooth.spline(x = RAM\_size$year, y = RAM\_size$Byte, spar = 2) # lambda = 2*

*lines(splines.reg.l1, col = "red", lwd = 1) # regression line with lambda = 0.2*

*lines(splines.reg.l2, col = "green", lwd = 1) # regression line with lambda = 1*

*lines(splines.reg.l3, col = "blue", lwd = 1) # regression line with lambda = 2*

**

Comments: The larger the spar is, more smooth the line is.

1. **Poisson Distribution**

*lambda=4*

*x=6*

*dpois(x,lambda)*

*lambda=5*

*x=0*

*dpois(x,lambda)*

**Homework 3**

1. **hash code**

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

*library(digest)*

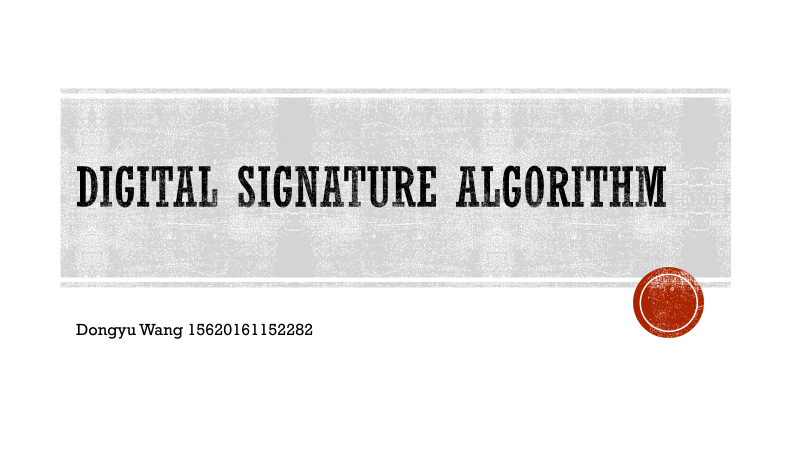
*digest("I learn a lot from this class when I am proper listening to the professor","sha256")*

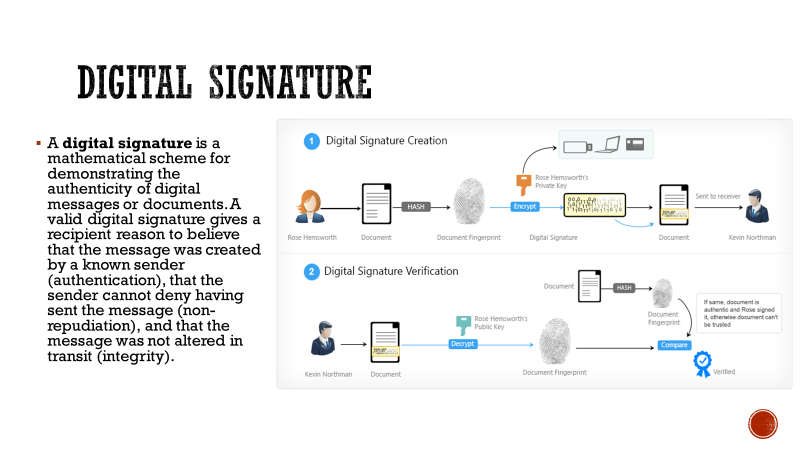
*# "c16700de5a5c1961e279135f2be7dcf9c187cb6b21ac8032308c715e1ce9964c"*

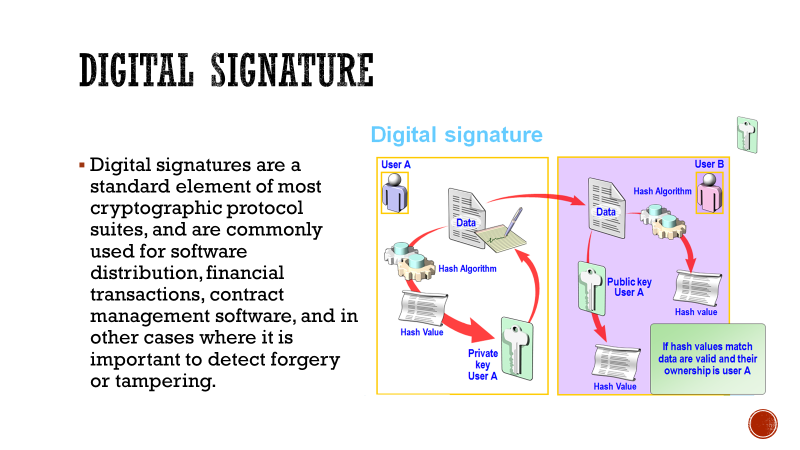
*digest("I do not learn a lot from this class when I am absent and playing on my Iphone","sha256")*

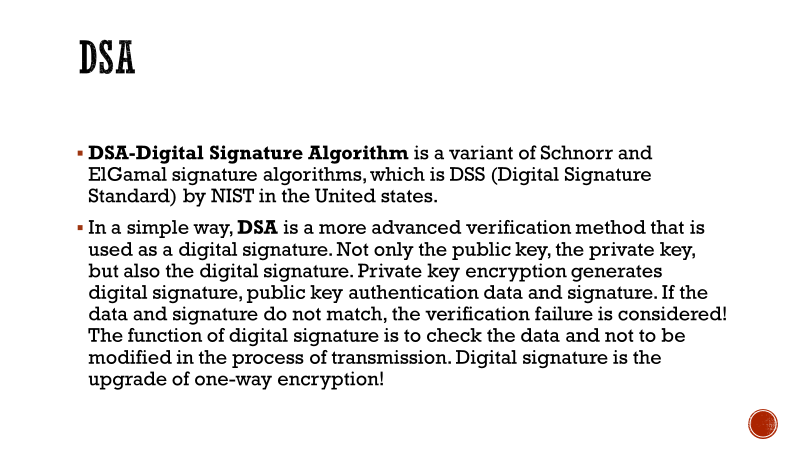
*# "2533d529768409d1c09d50451d9125fdbaa6e5fd4efdeb45c04e3c68bcb3a63e"*

1. **Digital Signature Algorithms**

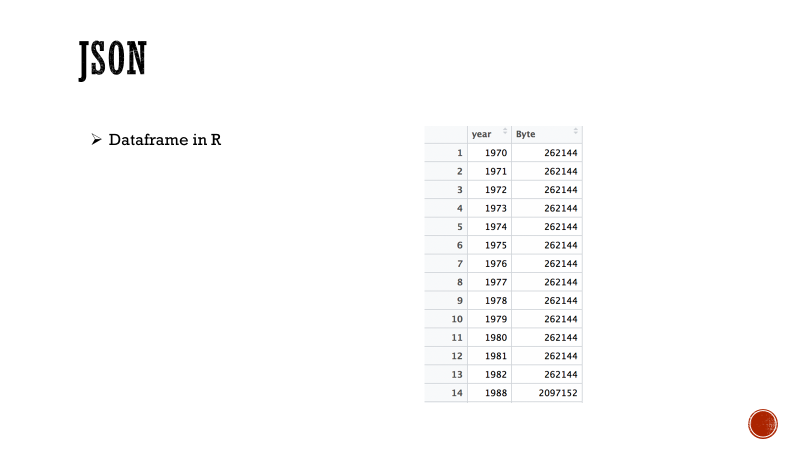


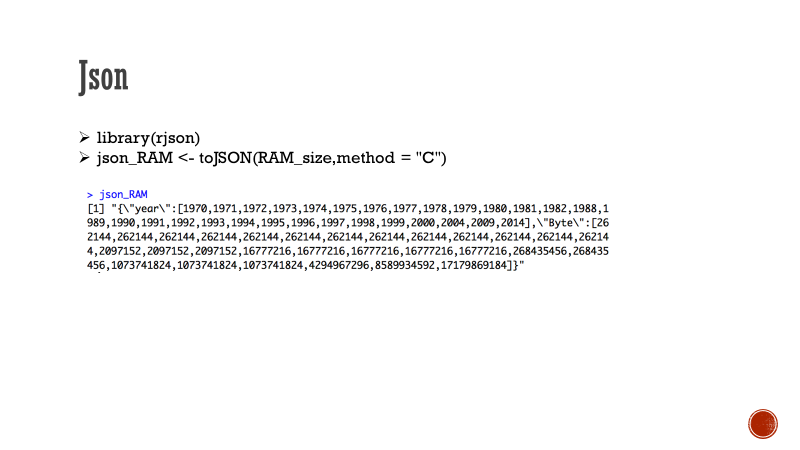






1. **Json data**





1. **CRIX data**

*#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)*

*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)*

*#install.packages("forecast")*

*#install.packages("tseries")*

*library(forecast)*

*library(tseries)*

*ts.plot(crix\_data\_frame$price)*

*Acf(crix\_data\_frame$price)*

*crixreturn <- diff(log(crix\_data\_frame$price))*

*for(i in 1:length(crix\_data\_frame$price)){*

*crixreturn[i] <- log(crix\_data\_frame$price[i+1]/crix\_data\_frame$price[i])*

*}*

*ts.plot(crixreturn)*

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

*# Box-Ljung test*

*data: crixreturn*

*X-squared = 34.644, df = 20, p-value = 0.02209*

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

*Acf(crixreturn)*

*Pacf(crixreturn)*

*arima(crixreturn,order = c(2,0,2))*

*# Call:*

*arima(x = crixreturn, order = c(2, 0, 2))*

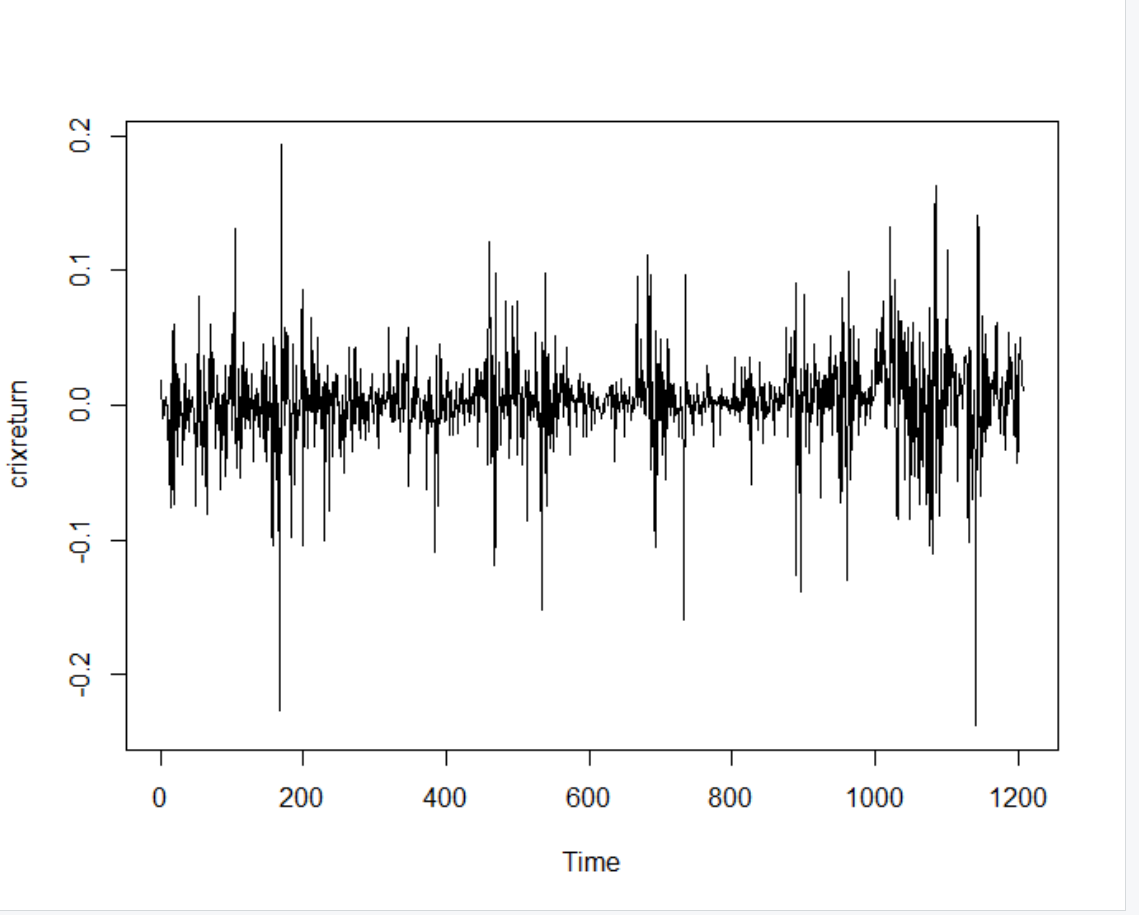
*Coefficients:*

*ar1 ar2 ma1 ma2 intercept*

*-0.8191 -0.7830 0.8308 0.7396 0.0024*

*s.e. 0.1773 0.1113 0.1945 0.1245 0.0010*

*sigma^2 estimated as 0.001182: log likelihood = 2355.48, aic = -4698.96*



**Homework 4**

1. **Figure3,4,5,6**

*#HW4.1*

*library(rjson)*

*json\_file = "http://crix.hu-berlin.de/data/crix.json"*

*json\_data = fromJSON(file=json\_file)*

*lst <- lapply(json\_data,function(x){*

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

*return(df)*

*})*

*crix\_data\_frame <- Reduce(rbind,lst)*

*crix\_data\_frame <- crix\_data\_frame[-1,]*

*load(file = "ecrix.RData")*

*load(file = "efcrix.RData")*

*length(ecrix)=length(crix\_data\_frame$price)*

*length(efcrix)=length(crix\_data\_frame$price)*

*ecrix\_data\_frame <- as.data.frame(ecrix)*

*efcrix\_data\_frame <- as.data.frame(efcrix)*

*#install.packages("dplyr")*

*library(dplyr)*

*sum\_crix <- cbind(crix\_data\_frame,ecrix\_data\_frame,efcrix\_data\_frame)*

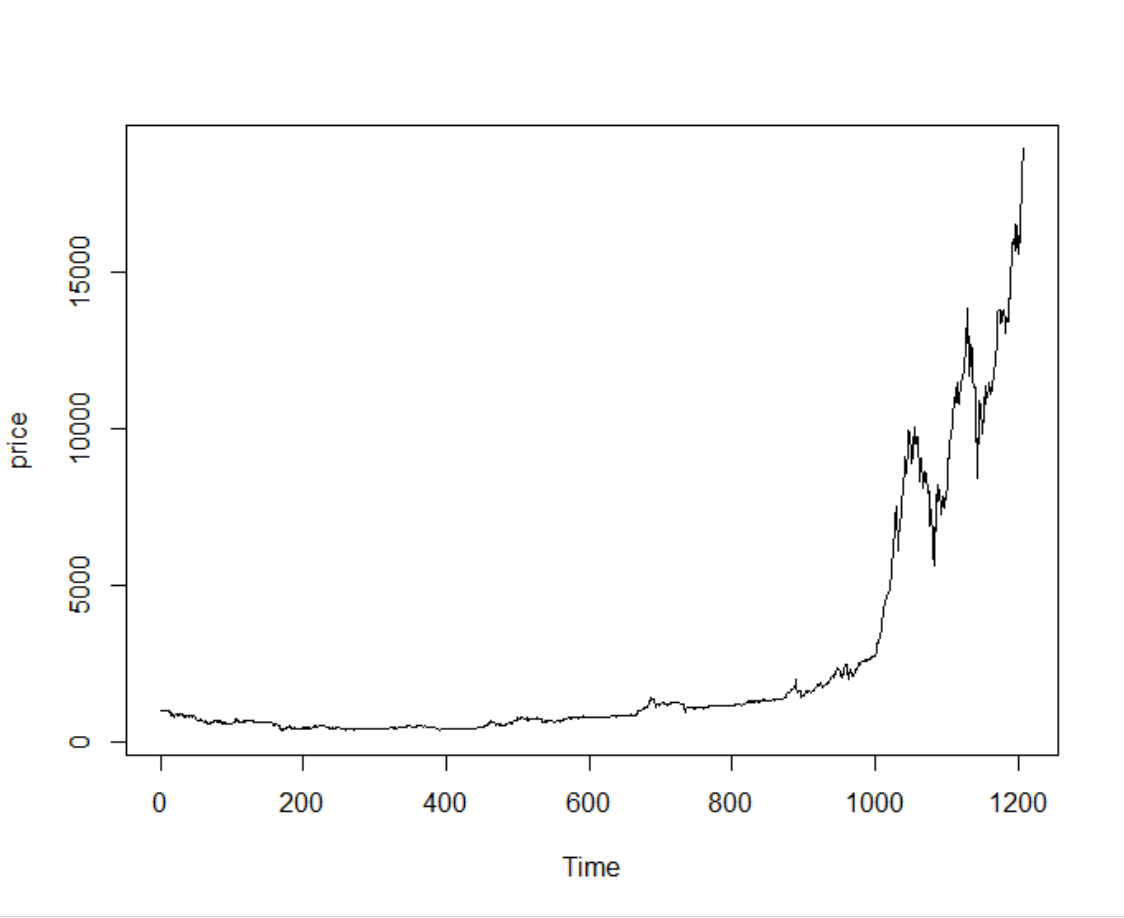
*#figure3*

*ts.plot(sum\_crix$price)*

*lines(sum\_crix$price,col="black",lwd=0.5)*

*lines(sum\_crix$ecrix,col="blue",lwd=1)*

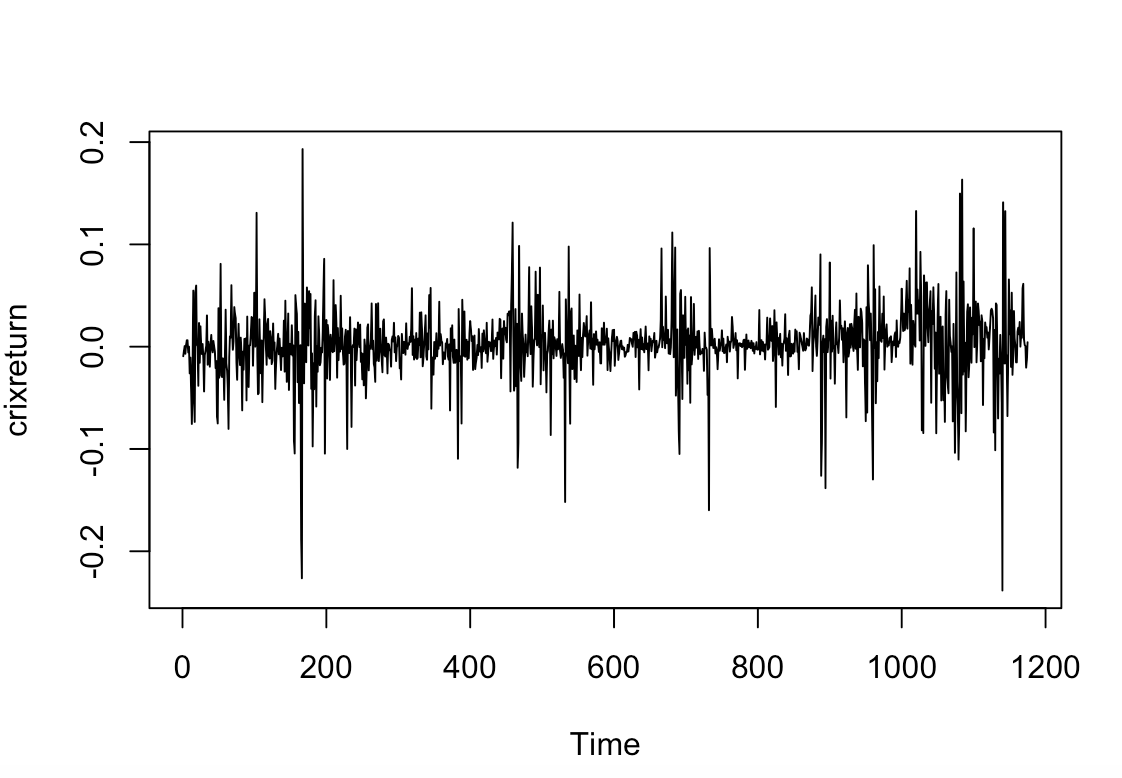
*lines(sum\_crix$efcrix,col="red",lwd=1)*



*#figure4*

*crixreturn <- diff(log(crix\_data\_frame$price))*

*ts.plot(crixreturn)*

**

*#figure5*

*hist(crixreturn,col = "grey",breaks = 20,freq = FALSE,ylim = c(0,25),xlab = NA)*

*lines(density(crixreturn),lwd=1)*

*mu = mean(crixreturn)*

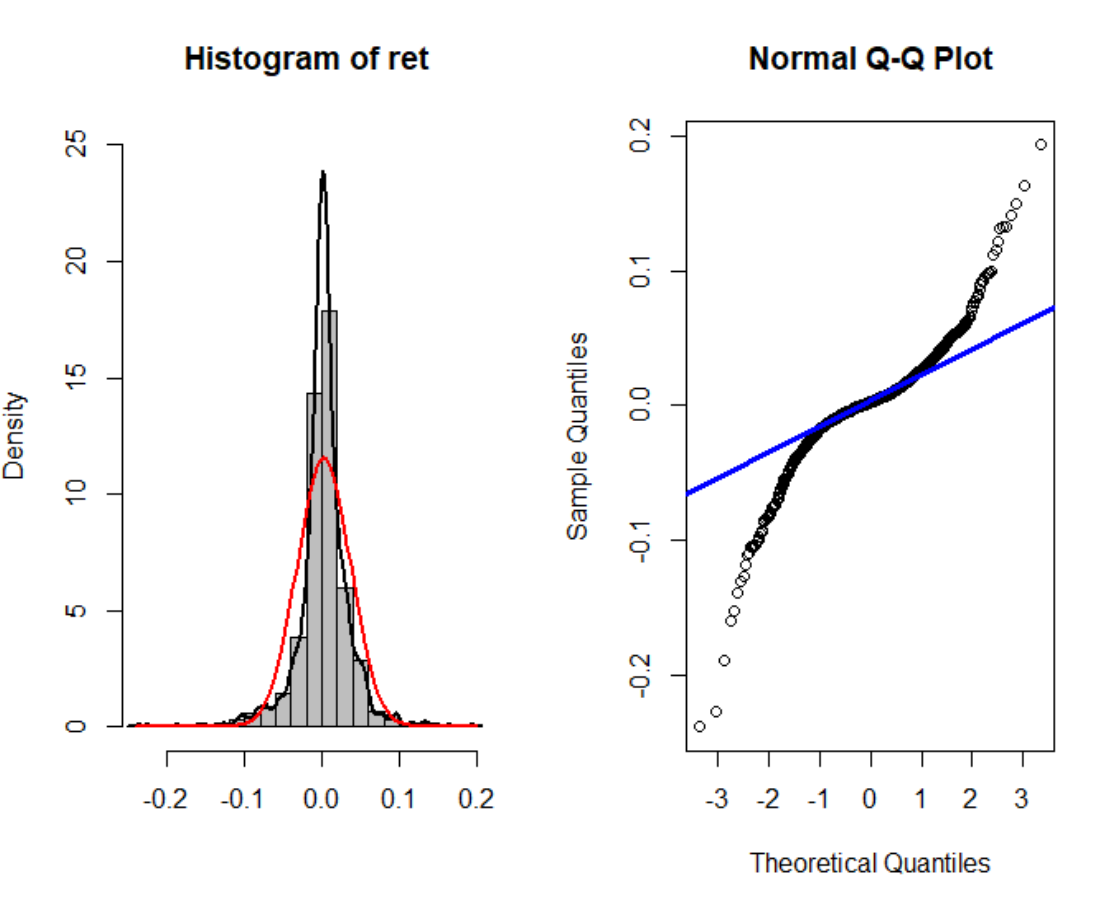
*sigma = sd(crixreturn)*

*x = seq(-4, 4, length = 100)*

*curve(dnorm(x, mean = mean(crixreturn), sd = sd(crixreturn)), add = TRUE, col = "darkblue", lwd = 1)*

*qqnorm(crixreturn)*

*qqline(crixreturn, col = "blue", lwd = 2)*



*#figure6*

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

*adf.test(crixreturn, alternative = "stationary")*

*kpss.test(crixreturn, null = "Trend")*

*par(mfrow = c(1, 2))*

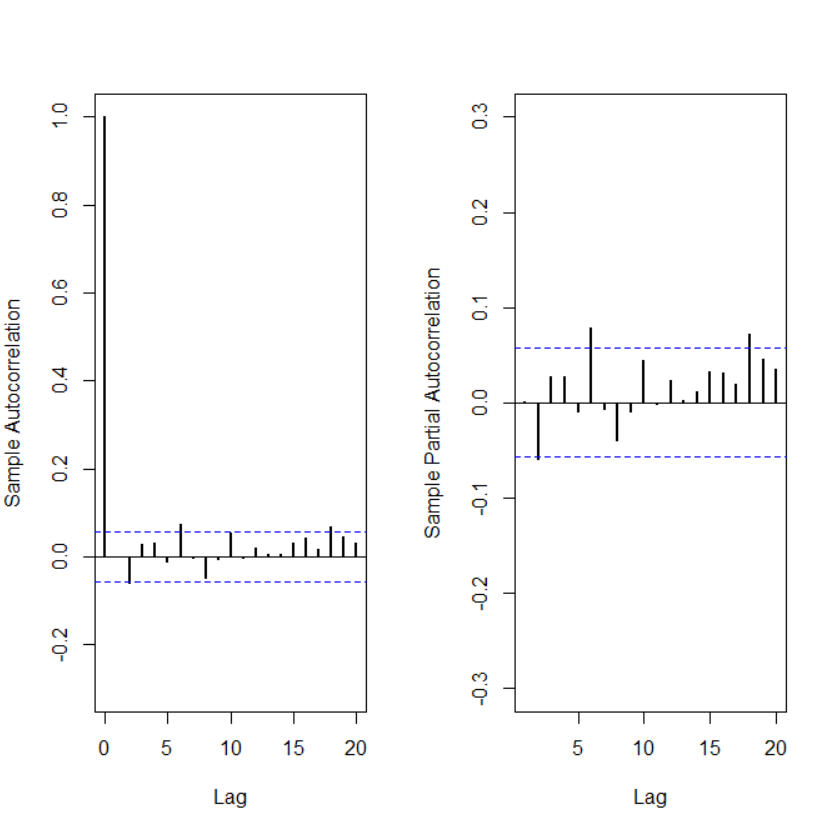
*autocorr = acf(crixreturn, lag.max = 20, ylab = "Sample Autocorrelation", main = NA, lwd = 2, ylim = c(-0.3, 1))*

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

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

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

*autopcorr = pacf(crixreturn, lag.max = 20, ylab = "Sample Partial Autocorrelation",main = NA, ylim = c(-0.3, 0.3), lwd = 2)*

**

1. **Figure 7**

*par(mfrow = c(1, 1))*

*auto.arima(crixreturn)*

*fit1 = arima(crixreturn, order = c(1, 0, 1))*

*tsdiag(fit1)*

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

*aic = matrix(NA, 6, 6)*

*for (p in 0:4) {*

*for (q in 0:3) {*

*a.p.q = arima(crixreturn, order = c(p, 0, q))*

*aic.p.q = a.p.q$aic*

*aic[p + 1, q + 1] = aic.p.q*

*}*

*}*

*aic*

*bic = matrix(NA, 6, 6)*

*for (p in 0:4) {*

*for (q in 0:3) {*

*b.p.q = arima(crixreturn, order = c(p, 0, q))*

*bic.p.q = AIC(b.p.q, k = log(length(crixreturn)))*

*bic[p + 1, q + 1] = bic.p.q*

*}*

*}*

*bic*

*fit4 = arima(crixreturn, order = c(2, 0, 3))*

*tsdiag(fit4)*

*Box.test(fit4$residuals, lag = 1)*

*fitr4 = arima(crixreturn, order = c(2, 1, 3))*

*tsdiag(fitr4)*

*Box.test(fitr4$residuals, lag = 1)*

*fit202 = arima(crixreturn, order = c(2, 0, 2))*

*tsdiag(fit202)*

*tsdiag(fit4)*

*tsdiag(fitr4)*

*AIC(fit202, k = log(length(crixreturn)))*

*AIC(fit4, k = log(length(crixreturn)))*

*AIC(fitr4, k = log(length(crixreturn)))*

*fit202$aic*

*fit4$aic*

*fitr4$aic*

*fit202 = arima(crixreturn, order = c(2, 0, 2))*

*crpre = predict(fit202, n.ahead = 30)*

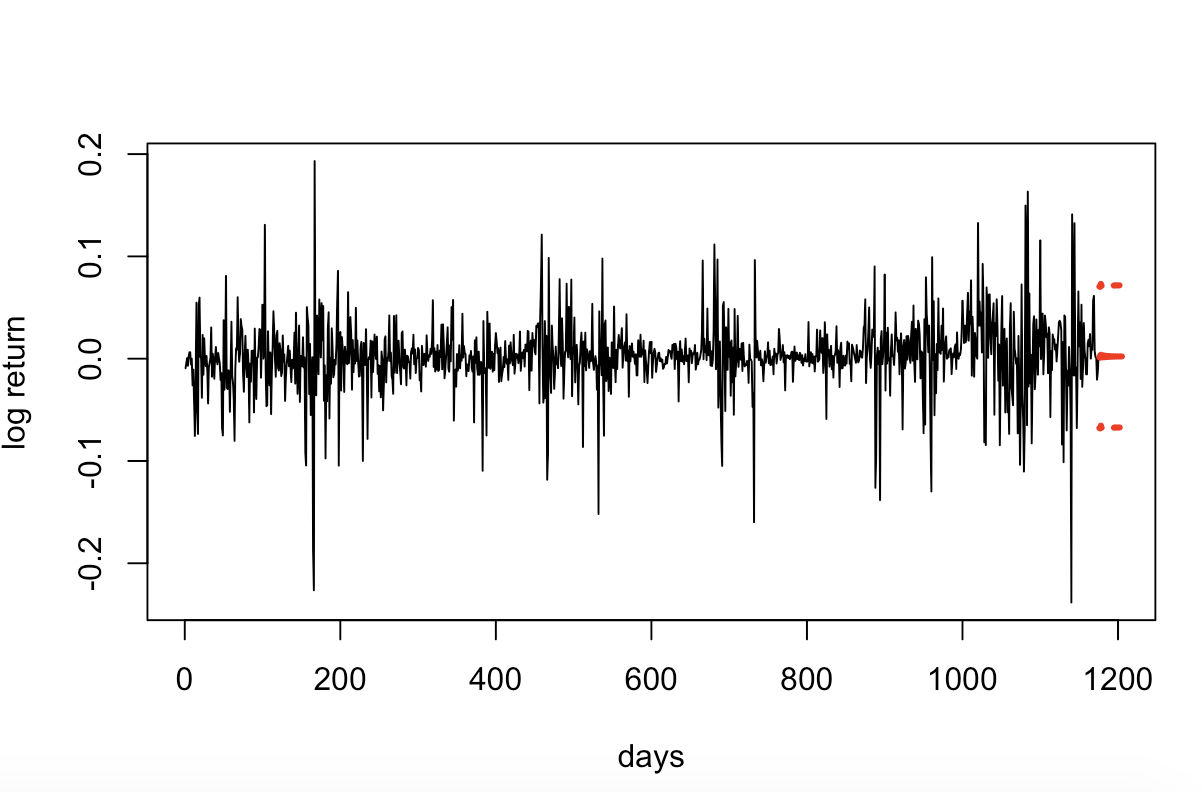
*dates = seq(as.Date("02/08/2014", format = "%d/%m/%Y"), by = "days", length = length(crixreturn))*

*plot(crixreturn, type = "l", xlim = c(0, 1200), ylab = "log return", xlab = "days", lwd = 1)*

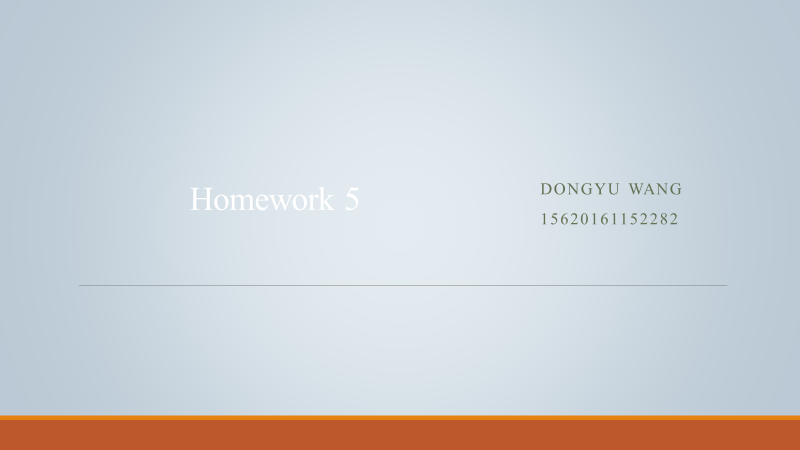
*lines(crpre$pred, col = "red", lwd = 3)*

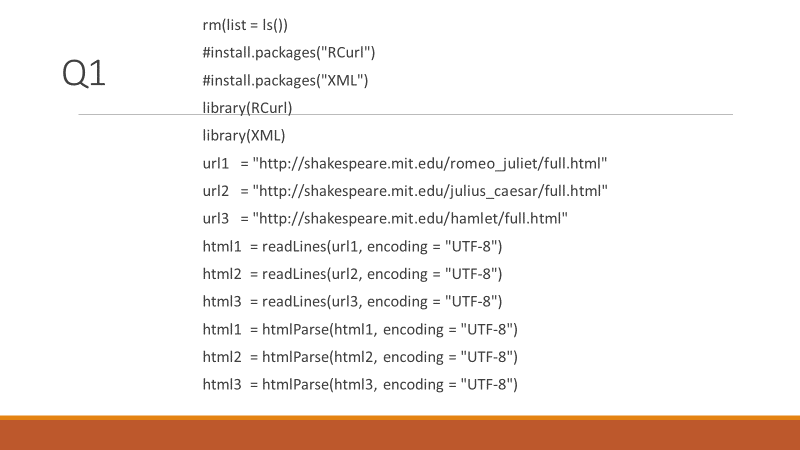
*lines(crpre$pred + 2 \* crpre$se, col = "red", lty = 3, lwd = 3)*

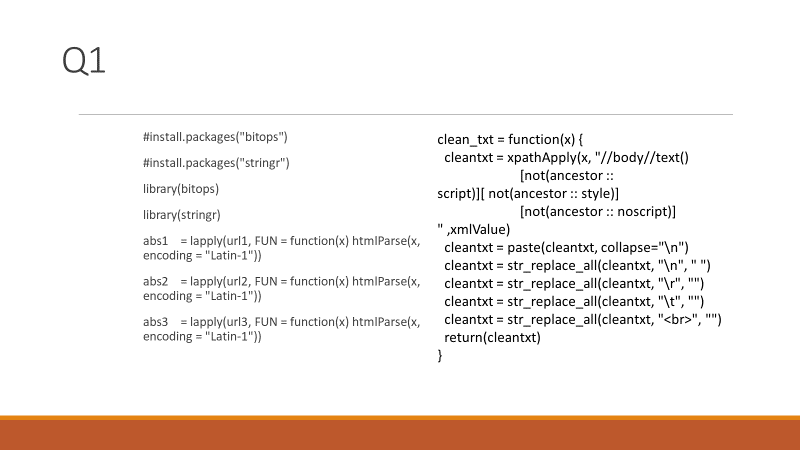
*lines(crpre$pred - 2 \* crpre$se, col = "red", lty = 3, lwd = 3)*

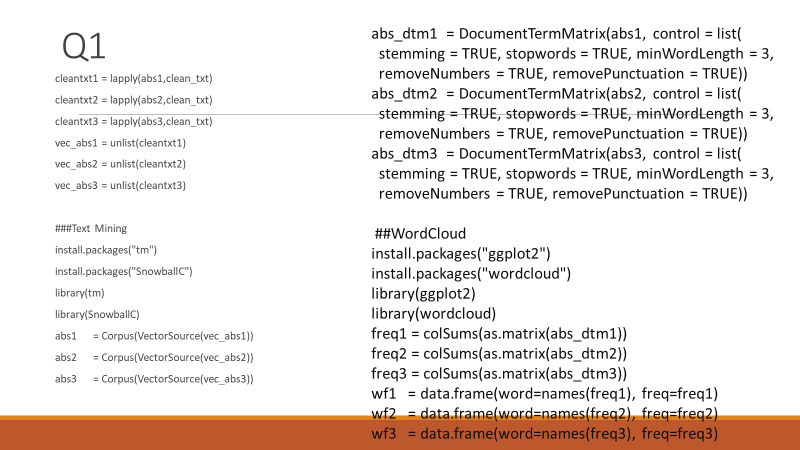
**

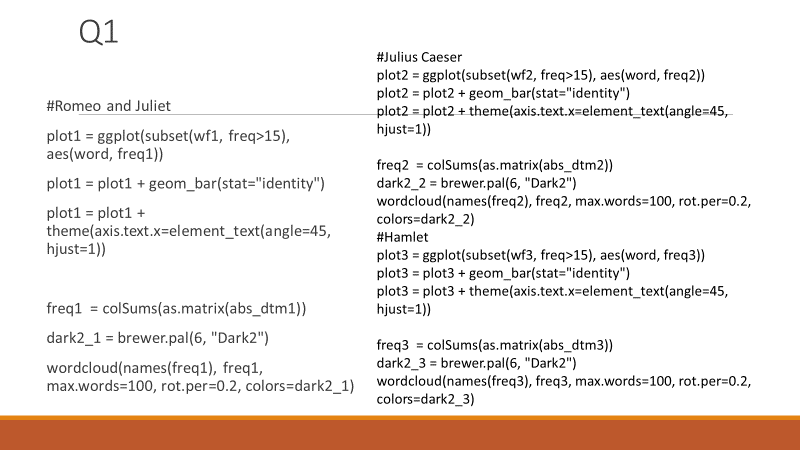
**Homework 5**

****

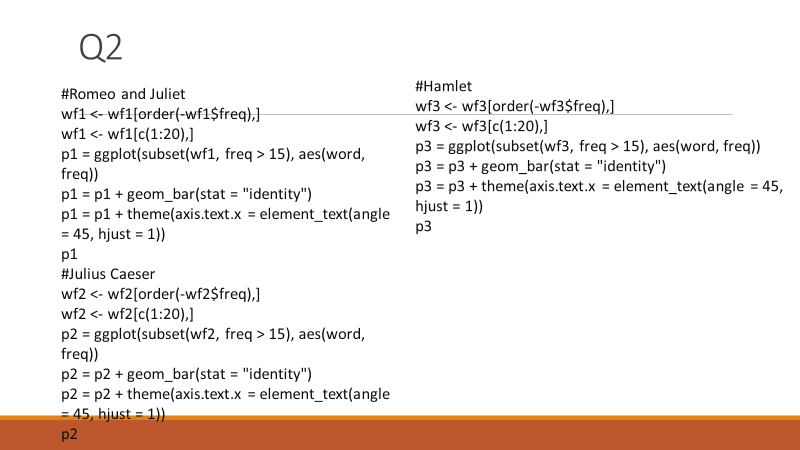
**

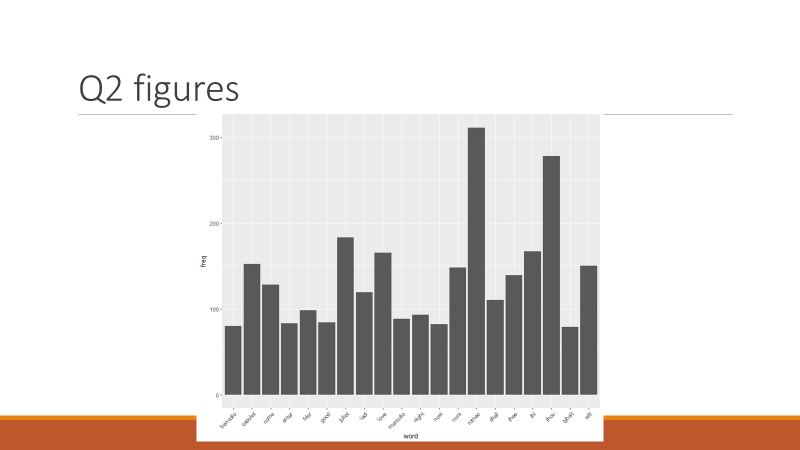
**

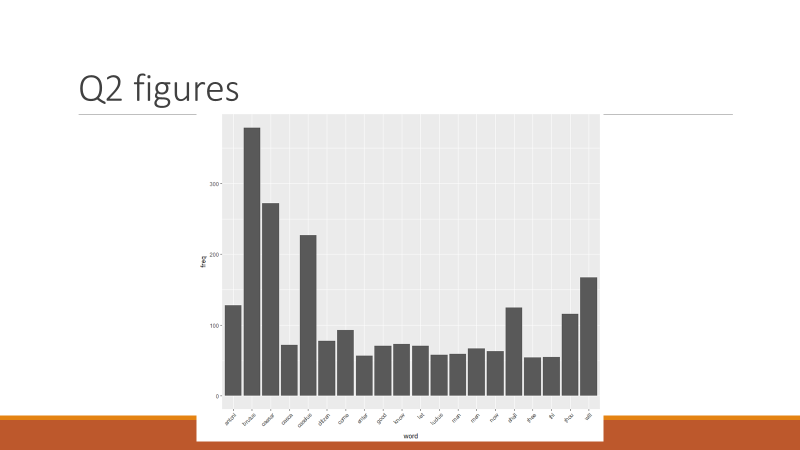


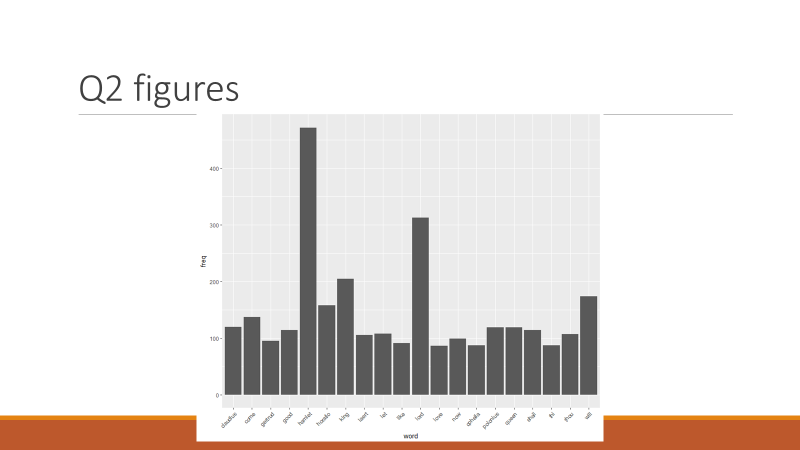
**

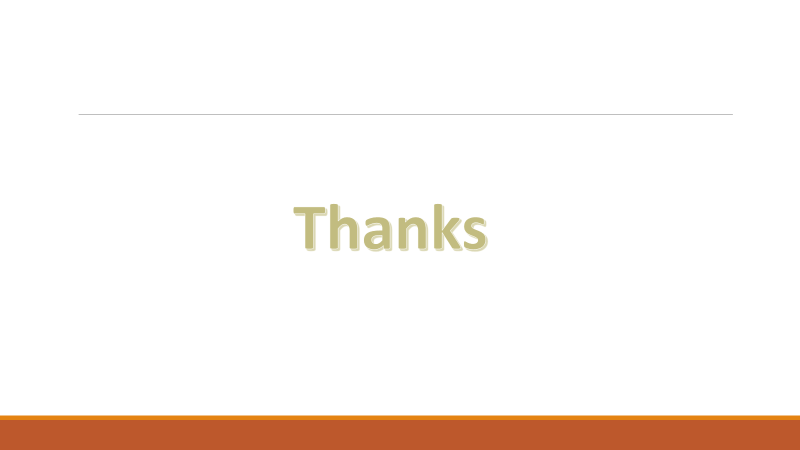
**

**

**

**

**

**