## R code to run data on cryptocurrencies data

```{r}

library(readr)

p <- read\_csv("~/SP/futureonly.csv")

library(plyr)

library(zoo)

library(survey)

library(lmtest)

library(tseries)

library(fUnitRoots)

```

```{r, echo=FALSE}

# eliminate at most one independent variable from regression

elim\_regress\_variable <- function(vars, elim="")

{

for (i in 1:NROW(vars))

{

if(elim==vars[i])

{

vars\_new <- vars[-i]

}

}

return(vars\_new)

}

# eliminate variable from group listings

elim\_group\_variable <- function(indep\_var\_groups, elim="")

{

groups <- indep\_var\_groups

# find group with this possible elimination candidate

for (i in 1:NROW(groups))

{

grp <- groups[[i]]

grp\_size <- NROW(grp)

for (j in 1:grp\_size)

{

#print(grp[j])

if (!is.na(grp[j]) && elim==grp[j])

{

grp <- grp[-j]

}

}

groups[[i]] <- grp

}

return(groups)

}

# create regression string for lm()

create\_regress\_str <- function(dep\_var, indep\_var, long\_term)

{

str <- paste(dep\_var," ~ ")

first <- 1

for (i in 1:NROW(indep\_var))

{

if (first==1)

{

str <- paste(str,indep\_var[i],sep="")

first <- 0

}

else

{

str <- paste(str,indep\_var[i],sep=" + ")

}

}

for (i in 1:NROW(long\_term))

{

str <- paste(str,long\_term[i],sep=" + ")

}

return(str)

}

# automatically eliminate least significant variable making sure

# to retain at least one per group; stop when all are at least

# significant at 10% level or are last in respective groups.

choose\_elim\_variable <- function(vars, groups, results)

{

sig\_level = 0.1

results\_data <- results$coeff

probs <- results\_data[,4]

prob\_order <- order(probs,decreasing=TRUE)

elim = "-1"

index = 1

max\_probs <- NROW(probs)

#print(paste("max\_probs: ",max\_probs, sep=''))

#print(probs)

#print(prob\_order)

repeat

{

if (index > max\_probs) break

var\_name <- names(probs[prob\_order[index]])

var\_value <- probs[[prob\_order[index]]]

#print(paste("index=",index,sep=''))

#print(paste("var\_name=",var\_name,sep=''))

#print(paste("var\_value=",var\_value,sep=''))

# find group with this possible elimination candidate

for (i in 1:NROW(groups))

{

grp <- groups[[i]]

grp\_size <- NROW(grp)

for (j in 1:grp\_size)

{

if (grp[j] == var\_name)

{

#print(grp[j])

if (grp\_size > 1 && var\_value > sig\_level)

{

return(var\_name)

}

}

}

}

index <- index + 1

}

return(elim)

}

# make the input file a CSV file from your Excel spreadsheet, and write to the output file in sink().

# you need to have the full directory name included!

#goldc <- read.csv("1978.csv",header=TRUE)

#sink("1978goldintg.pdf", type="output", split=TRUE)

# import all data as time series and take logs if needed; the names after the dollar sign

nl\_gg\_btc <- ts(log(p$btc\_per))

nl\_gg\_eth <- ts(log(p$eth\_per))

nl\_gg\_ltc <- ts(log(p$ltc\_per))

nl\_transact\_count <- ts(log(p$txCount))

nl\_totalcoin <- ts(log(p$generatedCoins))

nl\_sum\_vendor <-ts(log(p$sum))

nl\_cy\_at <- ts(log(p$cyber\_attack))

nl\_vix <- ts(log(p$`VIX Close`))

nl\_hash <- ts(log(p$hashrate))

gg\_btc <- lag(nl\_gg\_btc,-1)

gg\_eth <- lag(nl\_gg\_eth,-1)

gg\_ltc <- lag(nl\_gg\_ltc,-1)

transact\_count <- lag(nl\_transact\_count,-1)

totalcoin <- lag(nl\_totalcoin,-1)

sum\_vendor <-lag(nl\_sum\_vendor,-1)

cy\_at <- lag(nl\_cy\_at,-1)

vix <- lag(nl\_vix, -1)

hash <- lag(nl\_hash,-1)

# are the column names in the spreadsheet (or \*.csv file)

nl\_btc\_lg <- ts(log(p$`price(USD)`))

nl\_eth\_lg <- ts(log(p$`price(USD)\_e\_e`))

nl\_ltc\_lg <- ts(log(p$`price(USD)\_l`))

btc\_lg <- lag(nl\_btc\_lg,-1)

eth\_lg <- lag(nl\_eth\_lg,-1)

ltc\_lg <- lag(nl\_ltc\_lg,-1)

# get first difference of each variable

dBTC <- diff(btc\_lg)

dETH <- diff(eth\_lg)

dLTC <- diff(ltc\_lg)

dgg\_btc <-diff(nl\_gg\_btc)

dtransact\_count <- diff(nl\_transact\_count)

dtotalcoin <- diff(nl\_totalcoin)

dsum\_vendor <- diff(nl\_sum\_vendor)

dcy\_at <- diff(nl\_cy\_at)

dvix <- diff(nl\_vix)

# generate lag variables

lBTC1 <- lag(dBTC,-1)

lBTC2 <- lag(dBTC,-2)

lBTC3 <- lag(dBTC,-3)

lBTC4 <- lag(dBTC,-4)

lBTC5 <- lag(dBTC,-5)

lBTC6 <- lag(dBTC,-6)

lBTC7 <- lag(dBTC,-7)

lBTC8 <- lag(dBTC,-8)

lETH1 <- lag(dETH,-1)

lETH2 <- lag(dETH,-2)

lETH3 <- lag(dETH,-3)

lETH4 <- lag(dETH,-4)

lETH5 <- lag(dETH,-5)

lETH6 <- lag(dETH,-6)

lETH7 <- lag(dETH,-7)

lETH8 <- lag(dETH,-8)

lLTC1 <- lag(dLTC,-1)

lLTC2 <- lag(dLTC,-2)

lLTC3 <- lag(dLTC,-3)

lLTC4 <- lag(dLTC,-4)

lLTC5 <- lag(dLTC,-5)

lLTC6 <- lag(dLTC,-6)

lLTC7 <- lag(dLTC,-7)

lLTC8 <- lag(dLTC,-8)

ltransact\_count1 <- lag(dtransact\_count,-1)

ltransact\_count2 <- lag(dtransact\_count,-2)

ltransact\_count3 <- lag(dtransact\_count,-3)

ltransact\_count4 <- lag(dtransact\_count,-4)

ltransact\_count5 <- lag(dtransact\_count,-5)

ltransact\_count6 <- lag(dtransact\_count,-6)

ltransact\_count7 <- lag(dtransact\_count,-7)

ltransact\_count8 <- lag(dtransact\_count,-8)

ltotalcoin1 <- lag(dtotalcoin,-1)

ltotalcoin2 <- lag(dtotalcoin,-2)

ltotalcoin3 <- lag(dtotalcoin,-3)

ltotalcoin4 <- lag(dtotalcoin,-4)

ltotalcoin5 <- lag(dtotalcoin,-5)

ltotalcoin6 <- lag(dtotalcoin,-6)

ltotalcoin7 <- lag(dtotalcoin,-7)

ltotalcoin8 <- lag(dtotalcoin,-8)

lgg\_btc1 <- lag(dgg\_btc,-1)

lgg\_btc2 <- lag(dgg\_btc,-2)

lgg\_btc3 <- lag(dgg\_btc,-3)

lgg\_btc4 <- lag(dgg\_btc,-4)

lgg\_btc5 <- lag(dgg\_btc,-5)

lgg\_btc6 <- lag(dgg\_btc,-6)

lgg\_btc7 <- lag(dgg\_btc,-7)

lgg\_btc8 <- lag(dgg\_btc,-8)

lcy\_at1 <- lag(dcy\_at,-1)

lcy\_at2 <- lag(dcy\_at,-2)

lcy\_at3 <- lag(dcy\_at,-3)

lcy\_at4 <- lag(dcy\_at,-4)

lcy\_at5 <- lag(dcy\_at,-5)

lcy\_at6 <- lag(dcy\_at,-6)

lcy\_at7 <- lag(dcy\_at,-7)

lcy\_at8 <- lag(dcy\_at,-8)

lvix1 <- lag(dvix,-1)

lvix2 <- lag(dvix,-2)

lvix3 <- lag(dvix,-3)

lvix4 <- lag(dvix,-4)

lvix5 <- lag(dvix,-5)

lvix6 <- lag(dvix,-6)

lvix7 <- lag(dvix,-7)

lvix8 <- lag(dvix,-8)

lsum\_vendor1 <- lag(dsum\_vendor,-1)

lsum\_vendor2 <- lag(dsum\_vendor,-2)

lsum\_vendor3 <- lag(dsum\_vendor,-3)

lsum\_vendor4 <- lag(dsum\_vendor,-4)

lsum\_vendor5 <- lag(dsum\_vendor,-5)

lsum\_vendor6 <- lag(dsum\_vendor,-6)

lsum\_vendor7 <- lag(dsum\_vendor,-7)

lsum\_vendor8 <- lag(dsum\_vendor,-8)

# create dependent variable string and vector of all long term variables

dep\_var = "dBTC"

long\_term <- c("btc\_lg","eth\_lg","ltc\_lg","gg\_btc", "transact\_count", "totalcoin",

"sum\_vendor", "cy\_at","vix")

# create vector of all independent variable names

indep\_vars <- c("dBTC","lBTC1","lBTC2","lBTC3","lBTC4","lBTC5","lBTC6","lBTC7","lBTC8",

"dETH", "lETH1","lETH2","lETH3","lETH4","lETH5","lETH6","lETH7","lETH8",

"dLTC","lLTC1","lLTC2","lLTC3","lLTC4","lLTC5","lLTC6","lLTC7","lLTC8",

"dtransact\_count", "ltransact\_count1","ltransact\_count2","ltransact\_count3","ltransact\_count4","ltransact\_count5","ltransact\_count6","ltransact\_count7","ltransact\_count8",

"dtotalcoin" , "ltotalcoin1","ltotalcoin2","ltotalcoin3","ltotalcoin4","ltotalcoin5","ltotalcoin6","ltotalcoin7","ltotalcoin8",

"dgg\_btc","lgg\_btc1","lgg\_btc2","lgg\_btc3","lgg\_btc4","lgg\_btc5","lgg\_btc6","lgg\_btc7","lgg\_btc8",

"dcy\_at", "lcy\_at1","lcy\_at2","lcy\_at3","lcy\_at4","lcy\_at5","lcy\_at6","lcy\_at7","lcy\_at8",

"dvix","lvix1","lvix2","lvix3","lvix4","lvix5","lvix6","lvix7","lvix8",

"dsum\_vendor", "lsum\_vendor1","lsum\_vendor2","lsum\_vendor3","lsum\_vendor4","lsum\_vendor5","lsum\_vendor6","lsum\_vendor7","lsum\_vendor8"

)

# create groups of independent variables such that at least one member of

# each group must remain when using Henry's general to specific elimination

indep\_vars\_groups <- list(c("dBTC","lBTC1","lBTC2","lBTC3","lBTC4","lBTC5","lBTC6","lBTC7","lBTC8"),

c("dETH", "lETH1","lETH2","lETH3","lETH4","lETH5","lETH6","lETH7","lETH8"),

c("dLTC","lLTC1","lLTC2","lLTC3","lLTC4","lLTC5","lLTC6","lLTC7","lLTC8"),

c("dtransact\_count", "ltransact\_count1","ltransact\_count2","ltransact\_count3","ltransact\_count4","ltransact\_count5","ltransact\_count6","ltransact\_count7","ltransact\_count8"),

c("dtotalcoin" , "ltotalcoin1","ltotalcoin2","ltotalcoin3","ltotalcoin4","ltotalcoin5","ltotalcoin6","ltotalcoin7","ltotalcoin8"),

c("dgg\_btc","lgg\_btc1","lgg\_btc2","lgg\_btc3","lgg\_btc4","lgg\_btc5","lgg\_btc6","lgg\_btc7","lgg\_btc8"),

c("dcy\_at", "lcy\_at1","lcy\_at2","lcy\_at3","lcy\_at4","lcy\_at5","lcy\_at6","lcy\_at7","lcy\_at8"),

c("dvix","lvix1","lvix2","lvix3","lvix4","lvix5","lvix6","lvix7","lvix8"),

c("dsum\_vendor", "lsum\_vendor1","lsum\_vendor2","lsum\_vendor3","lsum\_vendor4","lsum\_vendor5","lsum\_vendor6","lsum\_vendor7","lsum\_vendor8")

)

# create time series dataset including the dependent and all possible independent variables

#NEED TO LOOK AT DATE VARIABLE

tsdata <- ts.union(dBTC,lBTC1, lBTC2,lBTC3,lBTC4,lBTC5,lBTC6,lBTC7,lBTC8,

dETH,lETH1,lETH2,lETH3,lETH4,lETH5,lETH6,lETH7,lETH8,

dLTC,lLTC1,lLTC2,lLTC3,lLTC4,lLTC5,lLTC6,lLTC7,lLTC8,

dtransact\_count,ltransact\_count1,ltransact\_count2,ltransact\_count3,ltransact\_count4,ltransact\_count5,ltransact\_count6,ltransact\_count7,ltransact\_count8,

dtotalcoin,ltotalcoin1,ltotalcoin2,ltotalcoin3,ltotalcoin4,ltotalcoin5,ltotalcoin6,ltotalcoin7,ltotalcoin8,

dgg\_btc,lgg\_btc1,lgg\_btc2,lgg\_btc3,lgg\_btc4,lgg\_btc5,lgg\_btc6,lgg\_btc7,lgg\_btc8,

dcy\_at, lcy\_at1,lcy\_at2,lcy\_at3,lcy\_at4,lcy\_at5,lcy\_at6,lcy\_at7,lcy\_at8,

dvix,lvix1,lvix2,lvix3,lvix4,lvix5,lvix6,lvix7,lvix8,

dsum\_vendor,lsum\_vendor1,lsum\_vendor2,lsum\_vendor3,lsum\_vendor4,lsum\_vendor5,lsum\_vendor6,lsum\_vendor7,lsum\_vendor8, btc\_lg, eth\_lg, ltc\_lg, transact\_count, totalcoin, gg\_btc, cy\_at, vix, sum\_vendor)

# begin with a strin$g that sets up the regression equation with dependent variable

# followed by "~" and then ALL independent variables set

#str <- create\_regress\_str(dep\_var, long\_term, indep\_vars)

str <- create\_regress\_str(dep\_var,long\_term, indep\_vars)

print(str)

regress <- lm(str, data=tsdata)

results <- summary(regress)

print(results)

# repeat regression eliminating one variable at a time for "elim" either by

# 1. TESTING: enter variables until "-1" is entered in console for testing

# 2. Calling choose\_elim\_variable() to eliminate completely through variables

repeat

{

#elim <- readline(paste("Which variable to eliminate (e.g. dlimp4 or -1 to end)? ", sep=""))

elim <- choose\_elim\_variable(indep\_vars, indep\_vars\_groups, results)

#print(elim)

if (elim=="-1") break

indep\_vars <- elim\_regress\_variable(indep\_vars, elim)

indep\_vars\_groups <- elim\_group\_variable(indep\_vars\_groups, elim)

str <- create\_regress\_str(dep\_var, indep\_vars, long\_term)

regress <- lm(str, data=tsdata)

results <- summary(regress)

}

print(results)

# RUN TESTS!!!

#("lETH1","lLTC1","gg\_btc", "transact\_count", "totalcoin", "sum\_vendor", "cy\_at")

# Wald testgg\_btc+transact\_count+totalcoin+sum\_vendor+future+ cy\_at+

wald <- regTermTest(regress, ~btc\_lg+ eth\_lg+ltc\_lg+gg\_btc+

transact\_count+totalcoin+sum\_vendor+

cy\_at, method="Wald")

print(wald)

coeffs <- regress$coefficients

coeff\_c <- -coeffs["(Intercept)"]/coeffs["btc\_lg"]

coeff\_lETH1 <- -coeffs["eth\_lg"]/coeffs["btc\_lg"]

coeff\_lLTC1 <- -coeffs["ltc\_lg"]/coeffs["btc\_lg"]

coeff\_gg <- -coeffs["gg\_btc"]/coeffs["btc\_lg"]

coeff\_tc <- -coeffs["transact\_count"]/coeffs["btc\_lg"]

coeff\_total <- -coeffs["totalcoin"]/coeffs["btc\_lg"]

coeff\_vendor <- -coeffs["sum\_vendor"]/coeffs["btc\_lg"]

coeff\_cyber <- -coeffs["cy\_at"]/coeffs["btc\_lg"]

coeff\_vix <- -coeffs["vix"]/coeffs["btc\_lg"]

# store standard errors in "stderrors"

stderrors <- results$coefficients[,2]

stderror\_c <- stderrors["(Intercept)"]

stderror\_lETH1 <- stderrors["eth\_lg"]

stderror\_lLTC1 <- stderrors["ltc\_lg"]

stderror\_gg <- -stderrors["gg\_btc"]

stderror\_tc <- -stderrors["transact\_count"]

stderror\_total <- -stderrors["totalcoin"]

stderror\_vendor <- -stderrors["sum\_vendor"]

stderror\_cyber <- -stderrors["cy\_at"]

stderror\_vix <- -stderrors["vix"]

# store t-stats values in "tstats"

tstats <- results$coefficients[,3]

tstats\_c <- tstats["(Intercept)"]

tstats\_lETH1 <- tstats["eth\_lg"]

tstats\_lLTC1 <- tstats["ltc\_lg"]

tstats\_gg <- -tstats["gg\_btc"]

tstats\_tc <- -tstats["transact\_count"]

tstats\_total <- -tstats["totalcoin"]

tstats\_vendor <- -tstats["sum\_vendor"]

tstats\_cyber <- -tstats["cy\_at"]

tstats\_vix <- -tstats["vix"]

# store probabilities in "probs"

results\_data <- results$coeff

probs <- results\_data[,4]

probs\_c <- probs["(Intercept)"]

probs\_lETH1 <- probs["eth\_lg"]

probs\_lLTC1 <- probs["ltc\_lg"]

probs\_gg <- -probs["gg\_btc"]

probs\_tc <- -probs["transact\_count"]

probs\_total <- -probs["totalcoin"]

probs\_vendor <- -probs["sum\_vendor"]

probs\_cyber <- -probs["cy\_at"]

probs\_vix <- -probs["vix"]

# print ecm coeffs, std-errors, t-stats, and probs

cat("Long Run Elasticities:\n")

cat("Variable,","Coeff,","Std-Errors,","t-stat,","prob","\n")

cat("(Intercept),",coeff\_c,",",stderror\_c,",",tstats\_c,",",probs\_c,"\n")

cat("lETH1,",coeff\_lETH1,",",stderror\_lETH1,",",tstats\_lETH1,",",probs\_lETH1,",")

cat("lLTC1,",coeff\_lLTC1,",",stderror\_lLTC1,",",tstats\_lLTC1,",",probs\_lLTC1,"\n\n")

cat("BTC\_searchterm,",coeff\_gg,",",stderror\_gg,",",tstats\_gg,",",probs\_gg,"\n\n")

cat("Transaction Count,",coeff\_tc,",",stderror\_tc,",",tstats\_tc,",",probs\_tc,"\n\n")

cat("Total Coin,",coeff\_total,",",stderror\_total,",",tstats\_total,",",probs\_total,"\n\n")

cat("Vendors,",coeff\_vendor,",",stderror\_vendor,",",tstats\_vendor,",",probs\_vendor,"\n\n")

cat("Cyberattack searchteam \t",coeff\_cyber,",",stderror\_cyber,",",tstats\_cyber,",",probs\_cyber,"\n\n")

cat("VIX Index,",coeff\_vix,",",stderror\_vix,",",tstats\_vix,",",probs\_vix,"\n\n")

# create ecm solution for long-range approx.

ecm <- btc\_lg - coeff\_c - coeff\_lETH1 \* nl\_eth\_lg - coeff\_lLTC1\*nl\_ltc\_lg - coeff\_gg\*nl\_gg\_btc - coeff\_tc\*nl\_transact\_count - coeff\_total\*nl\_totalcoin - coeff\_vendor\*nl\_sum\_vendor - coeff\_cyber\*nl\_cy\_at - coeff\_vix\*nl\_vix

#coeff\_c = -47540.92, coeff\_lETH1 =

ecm1 <- lag(ecm, -1)

tsdata <- ts.union(dBTC,lBTC1, lBTC2,lBTC3,lBTC4,lBTC5,lBTC6,lBTC7,lBTC8,

dETH,lETH1,lETH2,lETH3,lETH4,lETH5,lETH6,lETH7,lETH8,

dLTC,lLTC1,lLTC2,lLTC3,lLTC4,lLTC5,lLTC6,lLTC7,lLTC8,

dtransact\_count,ltransact\_count1,ltransact\_count2,ltransact\_count3,ltransact\_count4,ltransact\_count5,ltransact\_count6,ltransact\_count7,ltransact\_count8,

dtotalcoin,ltotalcoin1,ltotalcoin2,ltotalcoin3,ltotalcoin4,ltotalcoin5,ltotalcoin6,ltotalcoin7,ltotalcoin8,

dgg\_btc,lgg\_btc1,lgg\_btc2,lgg\_btc3,lgg\_btc4,lgg\_btc5,lgg\_btc6,lgg\_btc7,lgg\_btc8,

dcy\_at, lcy\_at1,lcy\_at2,lcy\_at3,lcy\_at4,lcy\_at5,lcy\_at6,lcy\_at7,lcy\_at8,

dvix,lvix1,lvix2,lvix3,lvix4,lvix5,lvix6,lvix7,lvix8,

dsum\_vendor,lsum\_vendor1,lsum\_vendor2,lsum\_vendor3,lsum\_vendor4,lsum\_vendor5,lsum\_vendor6,lsum\_vendor7,lsum\_vendor8,btc\_lg, eth\_lg, ltc\_lg, transact\_count, totalcoin, gg\_btc, cy\_at, vix, sum\_vendor,

ecm1)

# do FINAL regression WITH ecm

long\_term <- "ecm1"

str <- create\_regress\_str(dep\_var, indep\_vars, long\_term)

print(str)

regress <- lm(str, data=tsdata)

results <- summary(regress)

print(str)

print(regress)

print(results)

# Durbin-Watson test

dw <- dwtest(regress,data=tsdata)

print(dw)

# Breusch-Godfrey test

# bgtest(formula, order = 1, type = c("Chisq", "F"), data = list())

bp <- bgtest(regress,order=4,type="F",data=tsdata)

print(bp)

# Ramsey RESET test, fitted with quadratic

# resettest(formula, power = 2:3, type = c("fitted", "regressor", "princomp"), data = list())

ramsey <- reset(regress, power=2:3, type="fitted", data=tsdata)

print(ramsey)

# White's heteroskedastic test

#white <- vcovHC(regress, data=tsdata)

#white <- summaryw(regress)

#white <- ncv.test(regress)

#print(white)

#hetero\_cov\_mat <- hccm(regress)

#print(hetero\_cov\_mat)

# Jarque-Bera normality test

# skip first 3 NA values because of volatility

jarque\_test <- jarque.bera.test(ecm[4:length(ecm)])

print(jarque\_test)

# Dickey-Fuller unit root test

# skip first 3 NA values because of volatility

adf\_test <- adf.test(ecm[4:length(ecm)])

print(adf\_test)

# Unit root tests

print("Unit Root Test")

unitRootTest1 <- unitrootTest(ecm[4:length(ecm)],lags=4)

print(unitRootTest1)

# Augmented Dickey-Fuller test for unit roots using "urdfTest"

dickey <- urdfTest(ecm[4:length(ecm)], lags=4)

print(dickey)

# Phillips-Perron test for unit roots

phillips <- urppTest(ecm[4:length(ecm)], use.lag=4, doplot=TRUE)

print(phillips)

# Elliott-Rothenberg-Stock test for unit roots

elliot <- urersTest(ecm[4:length(ecm)])

print(elliot)

# Schmidt-Phillips test for unit roots

schmidt <- urspTest(ecm[4:length(ecm)])

print(schmidt)