Model

```
#################### Set Up Working Directory #######################


# Name working directory (must use forward slashes as separators and have a
    ↪ slash at the end)
myWD = "H:/Baruch/Fall 2016/Financial Statistics/Project/"

# Set my working directory as myWD, an example
setwd(myWD)

##############################################################################

##########################################################################
#################### Load S&P500 Tickers #######################
library(quantmod)
mySPReturnData <- read.csv(file ="Data/BloombergSP500HistoryFinal.csv", header=
    ↪ TRUE, sep=",", na.strings=c("NA"), stringsAsFactors=FALSE) #Path must be
    ↪ relative to myWD
myTicks <- colnames(mySPReturnData)
myTicksVec <-unlist(strsplit(myTicks, split=" "))
myTicksVec <- myTicksVec[-1] #vector of S&P 500 tickers
myTicksVec
head (myTicksVec)#show the first part of the ticker vector
tail (myTicksVec)#show the last part of the ticker vector


##########################################################################
#################### Correlation Matrix with Tawny #######################
#################### Setup #######################
library(tawny)# tawny package must be installed
library(tawny.types)# tawny.types package must be installed
require(tawny)
require(tawny.types)

#################### Do a Test with a Few Stocks #######################
#get returns
# test with a few stocks and inspect first
myTicksVecTest <-c('A','AA','AAPL')
m = length(myTicksVecTest)
Q = 5 # quality parameter
t=m*Q #should be 15 in this case
myEndDate <- as.Date("2016-11-30")
hTest <- getPortfolioReturns(myTicksVecTest, obs=t, end=myEndDate)
hTest
pTest <- TawnyPortfolio(hTest, t) #Represents a portfolio. Contains information
    ↪ about the portfolio composition, returns, window, etc. TawnyPortfolio(
```

```
    ↪ returns, window = 90)
str(pTest) #view a summary
numTickersTestCheck ← length(pTest[[1]])
numDates ← pTest$obs
corrsample ← cor.empirical(pTest$returns) #sample correlation matrix
corrsample # inspect

##################### Build a Portfolio from the S&P 500
    ↪ ########################
m = length(myTicksVec) # number of stocks
Q = 5 # quality parameter
t=m*Q  # number of observations for each stock
myEndDate ← as.Date("2016-11-30")
h ← getPortfolioReturns(myTicksVec, obs=t, end=myEndDate)
sapply(h, function(h) sum(is.na(h))) #view any missing values (there should be
    ↪  none)
#Note: The next step can take a while for large m and t!
p ← TawnyPortfolio(h, t) #Represents a portfolio. Contains information about
    ↪ the portfolio composition, returns, window, etc. TawnyPortfolio(returns,
    ↪  window = 90)
class(p) # p is a list
str(p)#view a summary

#check how many tickers were actually included
p[1]# view the tickers
numTickersCheck = length(p[[1]])#count the tickers included in the portfolio
paste((m - numTickersCheck), " tickers were dropped.")
#check how many datapoints for each ticker were included
numDatesCheck ← p$obs
paste((t - numDatesCheck), " dates were dropped.")

##################### Sample Correlation Matrix  ########################
#create a raw sample correlation matrix
corrsample ← cor.empirical(p$returns)
myeigenraw ← eigen(corrsample)$values
myeigenraw
myminraw= min(myeigenraw); myminraw
mymaxraw= max(myeigenraw); mymaxraw
numEigenRaw ← length(myeigenraw)
my2ndmaxraw = sort(myeigenraw,partial=numEigenRaw-1)[numEigenRaw-1]
my2ndmaxraw


##################### Correlation Matrix Denoised Using RMT
    ↪ ########################
#####################  Fit the density ########################
```

```
#################### Real sample ##########################################
#################### Prep the data by removing the largest eigenvalues ######
myeigenrawSorted <-sort(myeigenraw) #sorted
class(myeigenrawSorted)

myeigenrawSortedList <-as.list(myeigenrawSorted)
myeigenrawSortedTruncatedList <- myeigenrawSortedList[myeigenrawSortedList < 5]
    ↪  # remove largest eigenvalues before fitting
myeigenrawSortedTruncatedList <- myeigenrawSortedTruncatedList[
    ↪ myeigenrawSortedTruncatedList>0]
#check
truncMax<-max(unlist(myeigenrawSortedTruncatedList)) #3.845671
min(unlist(myeigenrawSortedTruncatedList))
truehist(unlist(myeigenrawSortedTruncatedList), h=0.1, xlim=range(c(0,truncMax
    ↪ ))) # h is the bin width


myeigenrawSortedTruncatedVector <- as.vector(unlist(
    ↪ myeigenrawSortedTruncatedList))
head(myeigenrawSortedTruncatedVector)

#################### Fit the Marcenko Pastur distribution and denoise the
    ↪ matrix ######
library(fitdistrplus) #requires the fitdistrplus package
library(covmat) #covmat package must be installed (needed for dmp)
#fit Marcenko Pastur to the bulk of the distribution
myfitRaw <- fitdist(myeigenrawSortedTruncatedVector, dmp, list(var=1, ndf=
    ↪ numDatesCheck, pdim=numTickersCheck, svr=(numDatesCheck/numTickersCheck)
    ↪ ), method="mge", gof="CvM")
 #sigma^2 0.4419794
 #Q =    2.0470204

###################### Replace eigenvalues below the cutoff with average
    ↪ ######
sigmaFitted = sqrt(myfitRaw$estimate["var"]); sigmaFitted
QFitted = myfitRaw$estimate["svr"]; QFitted
lambdaMaxCutoff = sigma * (1 + sqrt(1/Q))^2; lambdaMaxCutoff
meanRandom = mean(myeigenrawSortedTruncatedList<=lambdaMaxCutoff)
#replace all values below the cutoff with the mean
myeigenFiltered <-myeigenrawSortedList
myeigenFiltered[myeigenFiltered<= lambdaMaxCutoff] <- meanRandom


myeigenclean <-unlist(myeigenFiltered)
length(myeigenclean)
min(myeigenclean) # now the average

#################### Plot the Histograms
```

```
    ↪ ##############################################
par(mfrow=c(1,2))#graphical parameters
library(MASS)# MASS package must be installed to run the histograms
##################histograms
truehist(myeigenraw, h=0.1, prob=TRUE) # h is the bin width
truehist(myeigenclean, h=0.05, ylim=c(0,2), prob=TRUE)

myminclean = min(myeigenclean); myminclean
mymaxclean = max(myeigenclean); mymaxclean

#zoom in on the first part of the scale
maxx = 20 # x-axis cut-off for the zoom in
truehist(myeigenraw, h=0.1, xlim=range(c(0,maxx)), prob=TRUE) # h is the bin
    ↪ width

Q=numDatesCheck/numTickersCheck; Q
sigma=1
#superimpose Marcenko-Pastur on the zoomed-in sample
lambdaMax = sigma * (1 + sqrt(1/Q))^2
lambdaMax
lambdaMin = sigma * (1 - sqrt(1/Q))^2
lambdaMin
par(col="red", lwd=2);
curve ( Q /(2*pi*sigma^2) * sqrt( (lambdaMax - x)* (x-lambdaMin))/x, from =
    ↪ lambdaMin, to = lambdaMax, add=TRUE);
segments(0, 0, lambdaMin, 0, col="red")
segments(lambdaMax, 0, maxx, 0, col="red")

#superimpose the fitted Marcenko-Pastur on the zoomed-in sample
sigma = sigmaFitted
Q=QFitted
lambdaMax = sigma * (1 + sqrt(1/Q))^2
lambdaMax
lambdaMin = sigma * (1 - sqrt(1/Q))^2
lambdaMin
par(col="green", lwd=2);
curve ( Q /(2*pi*sigma^2) * sqrt( (lambdaMax - x)* (x-lambdaMin))/x, from =
    ↪ lambdaMin, to = lambdaMax, add=TRUE);
segments(0, 0, lambdaMin, 0, col="green")
segments(lambdaMax, 0, maxx, 0, col="green")


truehist(myeigenclean, h=0.1, xlim=range(c(0,maxx)),  ylim=c(0,2), prob=TRUE)
#output in: eigenvaluesbeforeandafter.bmp

#################### De-Diagonalized to get Correlation Matrix
    ↪ #####################
DiagonalFilteredM = matrix(nrow = numTickersCheck , ncol = numTickersCheck )
```

```
#assign filtered eigenvalues back to a matrix
for(i in 1:(numTickersCheck )){
  for (j in 1:numTickersCheck ){

    if (i == j){
      DiagonalFilteredM[i,j] <-  as.vector(myeigenclean)[i]
    }

  }
}
DiagonalFilteredM[is.na(DiagonalFilteredM)]<-0

#Since the intial correaltion matrix was diagonalized as L = V^-1 C V, so V L
    ↪ = C V and V L V-^1 = C
myRotM <- eigen(corrsample)$vectors # rotation matrix of eigenvectors used to
    ↪ go from the original matrix to the diagonlized eigenvalue matrix
myFilteredM <-  myRotM %*%DiagonalFilteredM %*% solve(myRotM) #rotate back to
    ↪ original correlation matrix
myFilteredM <- (myFilteredM + t(myFilteredM))/2 #symmetrize

detach_package(covmat)# need to detach to avoid conflict with the Matrix
    ↪ package
library(Matrix)# Matrix package must be installed

myFilteredM <- as.matrix(nearPD(myFilteredM, corr=TRUE)$mat)   # find nearest
    ↪ correlation matrix (positive semi-definite)
head(myFilteredM)
sum(diag(myFilteredM))# check that the eigenvalues add to m


diag(DiagonalFilteredM)#Tawny -type methodology
#c.clean <- myRotM  %*% DiagonalFilteredM %*% t(myRotM )
#diags <- diag(c.clean) %o% rep(1, nrow(c.clean))
#c.clean <- c.clean / sqrt(diags * t(diags))


#################### KL Divergence Using Tawny Package ####################
myKL <- divergence.kl(corrsample, myFilteredM);myKL

myKL <- divergence.kl(myFilteredM, corrsample);myKL #1st argument is assumed to
    ↪  be the "true distribution"


myFilteredMTawny <-denoise(p, RandomMatrixDenoiser())
#uses:
  #lambda.plus <- estimator$cutoff.fn(correlation, es, estimator)
  #estimator$clean.fn(es, lambda.plus)
```

```
sum(diag(myFilteredMTawny))# check that the eigenvalues add to m

myKL1 <- divergence.kl(myFilteredM, myFilteredMTawny);myKL1 #1st argument is
    ↪ assumed to be the "true distribution"




########################Utility Function######################
detach_package <- function(pkg, character.only = FALSE)
{
  if(!character.only)
  {
    pkg <- deparse(substitute(pkg))
  }
  search_item <- paste("package", pkg, sep = ":")
  while(search_item %in% search())
  {
    detach(search_item, unload = TRUE, character.only = TRUE)
  }
}
```

RMT Illustrations

```
##############################################################################
##################### Empirical Illustration of Random Matrix Theory Results
    ↪ ########################
par(mfrow=c(1,2))#graphical parameters

##############################################################################
##################### Empirical Illustration of Random Matrix Theory Results
    ↪ ########################
par(mfrow=c(1,2))#graphical parameters

#generate a symmetric random matrix whose entries are normally distributted
    ↪ and the off-diagonal elements have variance 1/N
n <- 5000;
m <- array(rnorm(n^2),c(n,n)); #rnorm creates n^2 random variates and allocates
    ↪  them to an array
m2 <- (m+t(m))/sqrt(2*n);# Make m symmetric and normalize for variance 1
lambda <- eigen(m2, symmetric=T, only.values = T);
e <- lambda$values;
hist(e,breaks=seq(-2.01,2.01,.02), main=NA, xlab="Eigenvalues",freq=F)
sigma <- 1;
par(col="red");
curve ( 1/(2*pi * sigma^2)*sqrt(4*sigma^2-x^2), -2*sigma, 2*sigma, add=TRUE);

#output in: random_matrix_eigenvalue_dist.png
##############################################################################
```

```
#generate a uniform random matrix whose entries are uniformally distributted and the off-diagonal elements have variance 1/N
#demonstrate 'universality' - distributions other than normal lead to the same  asymptotic result
n <- 5000;
mu <- array(runif(n^2),c(n,n)); #runif generates 5000^2 random deviates from the uniform distribution and adds fills them into an n x n array
mu2 <-sqrt(12)*(mu+t(mu)-1)/sqrt(2*n); # Make m symmetric and normalize for variance 1
lambdau <- eigen(mu2, symmetric=T, only.values = T);
eu <- lambdau$values;
histeu<-hist(eu,breaks=seq(-2.01,2.01,0.02), main=NA, xlab="Eigenvalues",freq=F)
#The density of eigenvalues is a semicircle, as predicted by Wigner's semicircle law.

sigma <- 1;
par(col="pink");
curve ( 1/(2*pi * sigma^2)*sqrt(4*sigma^2-x^2), -2*sigma, 2*sigma, add=TRUE);

#output in: random_matrix_normal_and_uniform_dist_w_semicircular_law.png
##############################################################################



#generate a random correlation matrix
par(mfrow=c(1,1))#graphical parameters
t <- 5000;
m <- 1000;
Q <- t/m
h <- array(rnorm(m*t),c(m,t)); # Time series in rows
e <- h %*% t(h)/t; # Form the correlation matrix
lambdae <- eigen(e, symmetric=T, only.values = T);
ee <- lambdae$values;
hist(ee,breaks=seq(0.01,3.01,.02), main="Random Sample Correlation" ,xlab="Eigenvalues",freq=F)

#superimpose Marcenko-Pastur
lambdaMax = sigma * (1 + sqrt(1/Q))^2
lambdaMax
lambdaMin = sigma * (1 - sqrt(1/Q))^2
lambdaMin
par(col="green");
curve (  Q /(2*pi*sigma^2) * sqrt( (lambdaMax - x)* (x-lambdaMin))/x, from = lambdaMin, to = lambdaMax, add=TRUE);
segments(0, 0, lambdaMin, 0, col="green")
```

```
#output in: random_matrix_correlation_w_MP_law.png
##############################################################################
#######################################################################


#################### Comparison of Scalar Asymptotic Results to Matrix Results
    ↪ #######################
par(mfrow=c(1,2))#graphical parameters

###################Normal Distribution################################
x <- seq(-5, 5, length=100)
y <- dnorm(x, mean=0, sd=1)
plot(x, y , type="l", lty=1, xlab="x value", ylab="Density", lwd=10, col="
    ↪ black")
###################Wigner's Semi-Circular Law################################
sigma <- 1;
par(col="pink");
curve ( 1/(2*pi * sigma^2)*sqrt(4*sigma^2-x^2), -2*sigma, 2*sigma, add=FALSE,
    ↪ xlab="x value", ylab="Density", lwd=10, col="red");

par(mfrow=c(1,2))#graphical parameters
###################X^2 Distribution ################################
x <- seq(0, 20, length=100)
numdf=1
y <- dchisq(x, df=numdf)
plot(x, y , type="l", lty=1, xlab="x value", ylab="Density", lwd=10, col=1)
numdf=2
y <- dchisq(x, df=numdf)
lines(x, y , type="l", lty=1, xlab="x value", ylab="Density", lwd=10, col=2)
numdf=5
y <- dchisq(x,df=numdf)
lines(x, y , type="l", lty=1, xlab="x value", ylab="Density", lwd=10, col=3)
numdf=10
y <- dchisq(x, df=numdf)
lines(x, y , type="l", lty=1, xlab="x value", ylab="Density", lwd=10, col=4)

colors <- seq(1:4)
labels <- c("df=1", "df=2", "df=3", "df=4")

legend("topright", inset=.05,
  labels, lwd=10, lty=c(1, 1, 1, 1), col=colors)

#################### Marcenko Pastur Density #######################
x <- seq(0, 20, length=100)
Q=1
y <- dmp(x, var=1, svr=Q)
plot(x, y , type="l", lty=1, xlab="x value", ylab="Density", lwd=10, col=1,
    ↪ ylim=c(0,2))
```

```
Q=2
y <-dmp(x, var=1, svr=Q)
lines(x, y , type="l", lty=1, xlab="x value",  ylab="Density",  lwd=10, col=2)
Q=5
y <-dmp(x, var=1, svr=Q)
lines(x, y , type="l", lty=1, xlab="x value",  ylab="Density",  lwd=10, col=3)
Q=10
y <- dmp(x, var=1, svr=Q)
lines(x, y , type="l", lty=1, xlab="x value",  ylab="Density",  lwd=10, col=4)

colors <- seq(1:4)
labels <- c("Q=1", "Q=2", "Q=3", "Q=4")

legend("topright", inset=.05,
  labels, lwd=10, lty=c(1, 1, 1, 1), col=colors)
```