## Part I Signal Generation

### Factor Models and Pairs-Trading

**What does the sign of $\beta$ indicate?**

Beta is indicative of the correlation between a stock with a comparative factor (in this case, XLK ETF). Simply put, it's like a bridge that measures the linkage of two stocks. A positive sign of beta indicates that both are moving in same direction. A negative sign indicates movement are opposite. Basically, we can say that the sign of the correlation is determinant of sign of beta.

**What do you think $\beta$ will be (positive or negative) for the Microsoft stock and the XLK factor? Why?**

We know that XLK ETF represents the Technology Select Sector SPDR ETF. Because Microsoft is a tech firm and XLK ETF is fund consisting of technology sectors, we could generally say that the beta would be positive, as both would ideally move in the same direction—positively correlated.

**What is a market-neutral trading strategy? Describe dollar-neutral trading strategies. What is $\beta$-neutral trading?**

Market neutral trading strategy aims at minimizing market risks by creating zero correlation for the portfolio to the market. According to Avellaneda and Lee, a market neutral portfolio has a property of having the sum of product of [dollar amounts in stocks] and [beta between factors] be zero. Regarding dollar-neutral trading strategy, it aims at managing risks while generating reasonable returns by means of keeping the correlation low compared to market averages. Regarding beta-neutral trading, it also aims at creating a portfolio with weighted average beta of zero such that market neutrality is accomplished to its best (meaning that market risks are avoided as much as possible without too much exposure).

**In terms of market-neutral trading, what do you think is better, a $\beta$ with large absolute value, or small absolute value? Why?**

If we want to achieve market-neutral, we'd better have very small beta in absolute value (hopefully very close to zero). This is because stocks that perfectly correspond the market has beta of exactly 1; stocks that are perfectly uncorrelated has -1 beta. Now we don't want exposure to market in order to minimize market risk. That's why beta with small absolute value is ideal.

### Using the Given Financial Data

We have provided you with the daily historical prices for both MSFT (Microsoft stock) and the XLK ETF (Technology Select Sector SPDR ETF) in CSV files, downloaded from Yahoo Finance.

**Suppose you want to use three months1 of historical data in the regression to determine the $\beta$ (via regression) for each day in the trading window, March 31st, 2018 to Dec 31st, 2019. What date range should you select to generate $\beta$'s for the entire time horizon?**

From the trading days calendar above, to determine beta for each day beginning March 31st, we need to get approximately 60 trading days prior to March 31st, 2018. This could be seen from the first three rows of calendar above.

**What is a better fundamental piece of data: the daily return or the daily stock price? What events could trigger a large, yet algorithmically insignificant, discrepancy between the price on two contiguous days? How would you ensure that all of your stock prices are in the same units (reference the same price at a given time)?**

I believe the daily return, instead of daily stock price, is a better fundamental piece of data. Sometimes, events like stock market crash would generates tremendous discrepancy between stock price from day to day. If I use daily return, because return is dependent only on percentage rather than price, the data wouldn't be so dramatically significant since price changes are algorithmically insignificant under market neutrality. The way to ensure all prices are in the same units is to divide all stock prices with a constant. This is why we would prefer daily return to daily stock price.

**Write a R, Python or MATLAB script to read in the data you downloaded and regress the (daily log-) returns of MSFT on the (daily log-) returns of the XLK ETF to estimate the $\beta$ for each day from March 31st, 2018 to Dec 31st, 2019, using the opening prices. The regression should use 60 days of historical data, so as not to depend on future information.**

In [1]:

```
# This function is for usage in the next block
# At each iteration of every 60 trading days, input two stock vectors with
# corresponding beta and intercept, output X_t for this iteration
get_Xk <- function(vec_msft,vec_XLK,beta_val,intercept) {
  for(i in 1:length(vec_msft)){
    epsilon = vec_msft - beta_val*vec_XLK
    X_k = rep(0,60)
    X_k[1] = epsilon[1]
    for(i in 2:60){
      X_k[i] = X_k[i-1] + epsilon[i] - intercept
    }
  }
  return(X_k)
}
```

## A Model For the Residual Process

**Questions:**

The above function gives us the $X_t$ ($X_k$ in my function) required by the question. It makes sense to assume $X_t$ to be mean reverting because sometimes, the price of microsoft stock price might be overpriced or underpriced due to company-related factors. Under EMH, in a highly efficient market, other stock prices of tech sector would also be affected. Hence, with mean-reversion time of the residual process, potential difference due to the incidents would be offset.

In [2]:

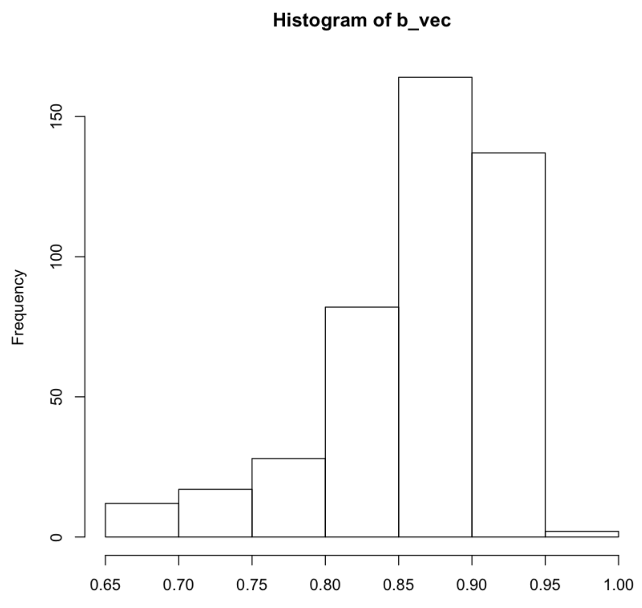
```
library(forecast)
```

```
Registered S3 method overwritten by 'xts':
  method     from
  as.zoo.xts zoo

Registered S3 method overwritten by 'quantmod':
  method            from
  as.zoo.data.frame zoo

Registered S3 methods overwritten by 'forecast':
  method             from
  fitted.fracdiff    fracdiff
```

```
residuals.fracdiff fracdiff
```

In [3]:

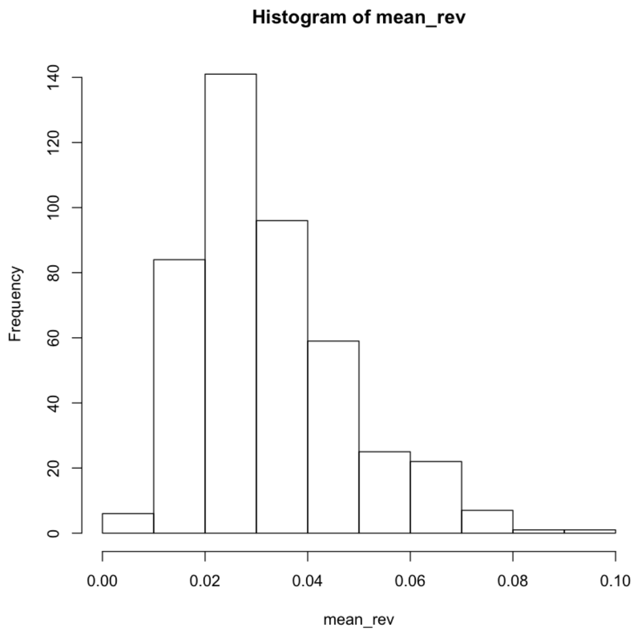
```
data_msft <- read.csv(file = 'MSFT.csv')
data_XLK <- read.csv(file = 'XLK.csv')

Beta=c() #capture series of beta for part 2.1.q2 plotting

## Variables in this section is for usage in Part 3 and Part 4
b_vec=c() # b in AR-1 Model
s_val=c() # s scales the displacement from mean by average size of deviation
s_list = c() #capture series of s_val
mean_rev=c() # half-life of process

#iterate through every 60 trading days
for(i in 1:(length(data_msft$Open)-60)){

  #calculate daily return using open price for two stocks
  vec_price_msft = data_msft[c(i:(i+60)), "Open"]
  vec_logreturn_msft <- log(vec_price_msft[-1]/vec_price_msft[-length(vec_price_msft)])
  vec_price_XLK <- data_XLK[c(i:(i+60)), "Open"]
  vec_logreturn_XLK <- log(vec_price_XLK[-1]/vec_price_XLK[-length(vec_price_XLK)])

  #regression
  model <- lm(vec_logreturn_msft ~ vec_logreturn_XLK)
  beta = model$coefficients[2]
  intercept = model$coefficients[1]

  #append information
  X_k=c()
  X_k <- c(X_k,get_Xk(vec_logreturn_msft,vec_logreturn_XLK,beta, intercept))
  Beta <- c(Beta, beta)

  #perform AR(1)
  model = arima(X_k, order=c(1,0,0))
  a = model$coef[2]
  b = model$coef[1]
  b_vec <- c(b_vec, b)
  k = -log(b)*252
  mean_rev <- c(mean_rev,1/k)
  m = a/(1-b)
  sigma_2 = model$sigma2[1]
  sigma = sqrt(sigma_2*2*k/(1-b^2))
  sigma_eq = sigma/sqrt(2*k)
  s_val= -m/sigma_eq
  s_list <- c(s_list,s_val)
}

hist(b_vec)
```



b_vec

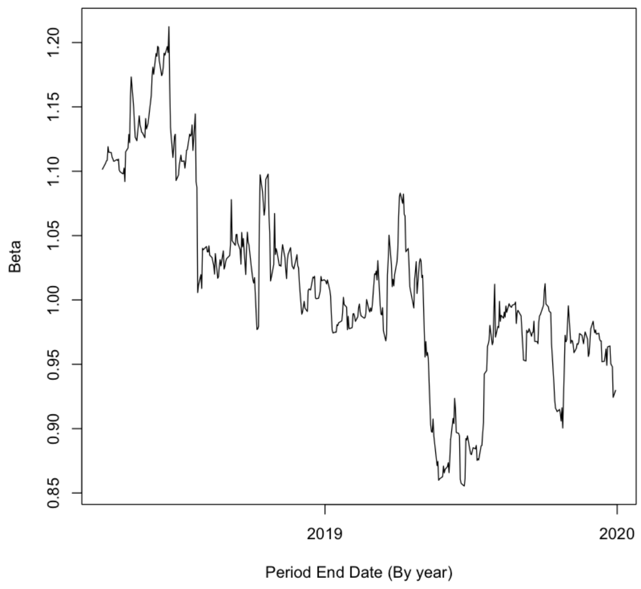
In [4]:

```
hist(mean_rev)
```



In [5]:

```
D = data_msft[c(61:(length(data_msft$Open))), "Date"]
plot(Beta~as.Date(D,"%Y-%m-%d"),type="l",xlab="Period End Date (By year)",ylab="Beta")
```



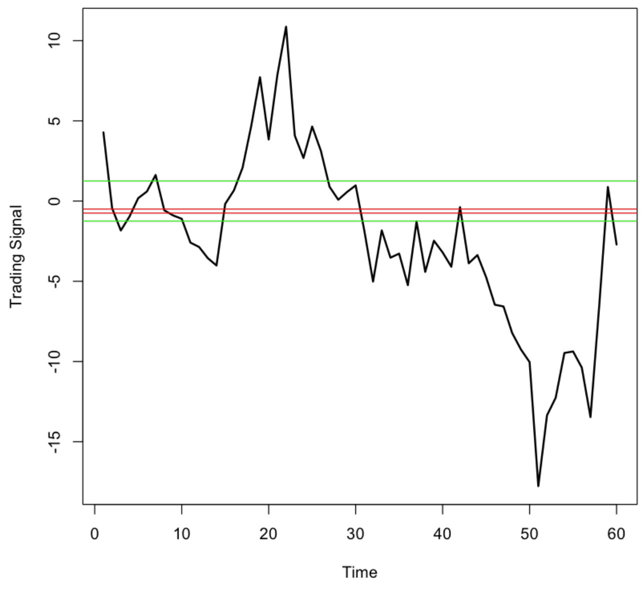
From above, the mean_rev one is more skewed than beta one. This implies that the hypothesis in our mean reversion fails more than the hypothesis in beta (not close to 1).

## Generating a Trading Signal

Suppose we want to buy a dollar in microsoft and sell beta amount of ETF, the instantaneous rate is supposedly $k(m - X_t)d_t$. As a rule of thumb, we would want to remain in a position so long as the $m - X_t$ is positive, i.e. expected additional rate of return is positive. Specifically speaking, when $m - X_t$ is positive enough, we would maintain in the position to account for transaction costs; upon becoming negative, we would exit as some point. A positive signal indicates that shorting the MSFT and buying the ETF is a strong decision; a negative one indicates the other way around. For s, we're supposed to have [short enter > short exit > 0] and [long enter < long exit < 0].

In [6]:

```
plot(s_list[1:60],type='l', col="black", lwd = 2, xlab="Time",ylab="Trading Signal") + abline(h=1.2
5, col="green") + abline(h=-1.25, col="green") + abline(h=-0.5, col="red") + abline(h=-0.75, col="re
d")
```



I entered long position for 5 times and entered short position for 2 times.

Evaluating Sharpe ratio for the profit by the strategy, assuming a 2% constant growth, we'd have at least an at least 3-times-better ratio compared to trading without pair-trading-signal.

## Part II Empirical Factors

## Empirical Correlation Matrices

**Questions:**

**Write a R, Python or MATLAB script to compute the eigen-portfolios on a given day.**

In [7]:

```
library(R.matlab)
```

```
R.matlab v3.6.2 (2018-09-26) successfully loaded. See ?R.matlab for help.


Attaching package: ‘R.matlab’


The following objects are masked from ‘package:base’:

    getOption, isOpen
```

In [ ]:

```
final_proj_data <- readMat("finalproject_data.mat")

#process data
num_stocks <- final_proj_data$nStocks[1,1]

num_days <- final_proj_data$nDays[1,1]

return_dat <- final_proj_data$returns
dat <- as.data.frame(t(as.matrix(return_dat)))

stock_name_list <- c()
for(i in 1:422){
  each_stock_name= toString(final_proj_data$names[[i]])
  #print(each_stock_name)
  stock_name_list <- c(stock_name_list,each_stock_name)
}

stock_ticker_list <- c()
for(i in 1:422){
  each_ticker_name= toString(final_proj_data$tickers[[i]])
  names(dat)[i] <- each_ticker_name
  stock_ticker_list <- c(stock_ticker_list,each_ticker_name)
}
```

In [ ]:

```
F1d_vec <- c()

# Beginning from day 252, up to day 502
for(d in (252:num_days)){
  interim = return_dat[,(d-251):d]

  rbar_vec = rowSums(interim)/252
  s_square = rowSums((interim - rbar_vec)^2)/251

  Y = interim / sqrt(s_square)

  #get matrix rho
  rho = Y%*%t(Y)/251
  #retrieve eigen values and vectors
  V = eigen(rho)$vectors
  lambda_val = eigen(rho)$values
  lambda_val_precentage = lambda_val/num_stocks
  lambda = diag(lambda_val)

  P = V
  P_hat = P

  PR = 0

  for(k in (1:num_stocks)){
    P[,k] = V[,k]/sqrt(s_square)
    P_hat[,k] = P[,k]/sum(P[,k])

    # Calculating cumulative return
    PR = PR + (P_hat[k,1]*return_dat[k,d])

  }
  F1d_vec <- c(F1d_vec,PR)

}
```

In [ ]:

```
data_SPY_yahoo <- read.csv(file = 'SPY_from_YahooFinance.csv')
SPY_price_yahoo <- data_SPY_yahoo$Open
SPY_return_yahoo <- diff(SPY_price_yahoo)/SPY_price_yahoo[-length(SPY_price_yahoo)]

plot(F1d_vec,type='l',ylim=c(-0.03,0.03),col='aquamarine4')
lines(SPY_return_yahoo,type='l',col='deeppink3')
```

**Plot the cumulative return of the principal eigen-portfolio over the course of time. How does it compare to the market? (You can use the SPY (an index based on the S&P 500) as a representation of the market; you need only look at a plot of it on, say, Google's finance page.)**

In [ ]:

```
cumu_eigen_returns = c()
cumu_SPY_returns = c()
for(i in (1:length(SPY_return_yahoo))){
  cumu_eigen_returns = c(cumu_eigen_returns,sum(F1d_vec[1:i]))
  cumu_SPY_returns = c(cumu_SPY_returns,sum(SPY_return_yahoo[1:i]))
}

plot(cumu_eigen_returns, type = 'l', col='aquamarine4')
lines(cumu_SPY_returns, type='l', col='deeppink3')
```

From the plot above, the cumulative return aligns with the market well.

**Carefully read Avellaneda and Lee's discussion about eigen-portfolios. Explain, in your own words, the financial implications of the weighting given to each stock in different eigen-portfolios. For example, if stock i has a large positive value in the second eigen-portfolio but a large negative value in the third, and stock j has a large positive value in both eigen-portfolios, what can be said about the following two positions: going long in both stock i and stock j and going long in stock j and shorting stock i?**

According to Avellaneda and Lee, the weights in the dominant eigenportfolio are inversely correlated to the stock volatility. In this paper, we also know that if we relabel evec in descending order, then the nearest stocks tend to be in the same industry sector. As number increases, this becomes not as true. Consider stock i has large positive 2nd and large negative 3rd eval; j has large positive in both 2nd and 3rd, we would have the 2nd eigenportfolio associating to longing both stocks yet 3rd eignportfolio for shorting i and longing j.

## The Marcenko-Pastur Distribution for the Distribution of Eigen- values

## The Marcenko-Pastur Distribution for the Distribution of Eigen- Values

**Let n = 5000 and m = 1000 and suppose the Xij are mean zero, variance one Gaussian random variables. Using R, Python or MATLAB, simulate a data matrix X, compute the eigen-values of its covariance matrix, and plot the distribution of the eigenvalues. On top of this, plot the limiting Marcenko-Pastur density. Comment on the results.**

In [9]:

```
#Number of rows
m = 1000
#Number of columns
n = 5000

Q = n/m

#Generate matrix
X <- matrix(rnorm(m*n,mean=0,sd=1), m, n)
A <- 1/(n-1)*X%*%t(X)

#Perform Eigenvalue decomposition
E <- eigen(A)$vectors
d <- eigen(A)$values
D <- diag(d)

interval_6a <- c(seq(0,2.5,by=0.05))
histo_6a <- hist(d, plot=FALSE, breaks=interval_6a)
density_6a <- c(histo_6a$counts)
experiment_6a = density/length(d)
max(d)
```

```
2.09849881358075
```

In [10]:

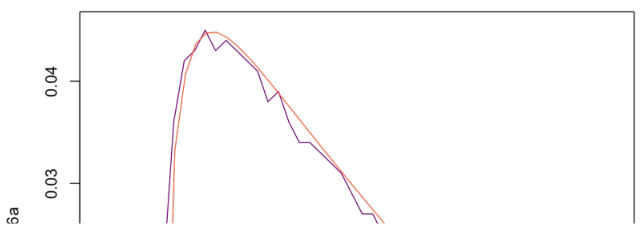
```
min(d)
```

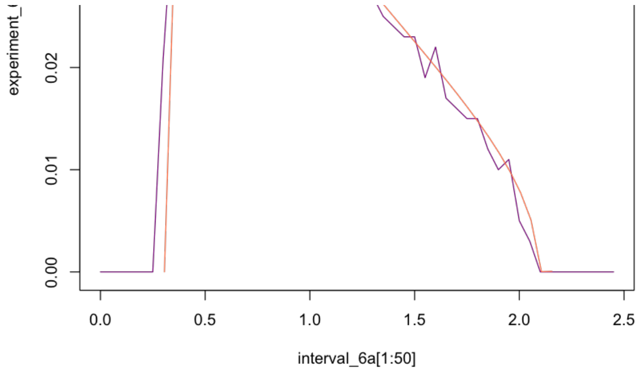
```
0.305813459481512
```

In [11]:

```
lambda_plus = (1+sqrt(1/Q))^2
lambda_minus = (1-sqrt(1/Q))^2

lambda <- c(seq(lambda_minus,2.2,by=0.05))
scalar = Q/2/pi
i = lambda_plus - lambda
j = lambda - lambda_minus
k = sqrt(as.complex(i*j))
rho = scalar * k / lambda
rho = rho/sum(rho)
#experimental
plot(interval_6a[1:50], experiment_6a, type='l', col='darkmagenta')
#Marcenko-Pastur
lines(lambda, rho, type='l', col='coral')
```

```
Warning message in xy.coords(x, y):
“imaginary parts discarded in coercion”
```





Both lines align with each other well with the defined bin size.

**Often times stock returns are modeled as iid Gaussian random variables. If this were the case, the distribution of the eigenvalues of the correlation matrix of the stock returns should look like the Marcenko-Pastur Distribution. Standardize the entire history of stock returns so that the mean for each stock is zero and the variance one (over the entire 502 day period). Compute the eigenvalues of the data's correlation matrix. Plot the density of the eigenvalues as well as the theoretical distribution given by the Marcenko-Pastur law. How do they compare?**

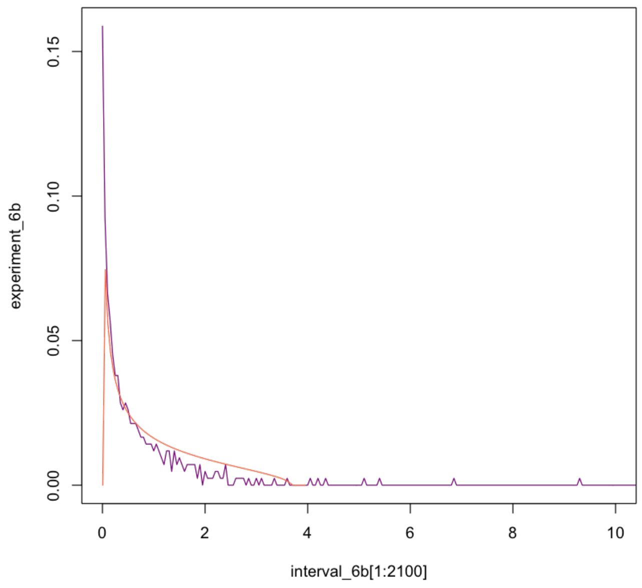
In [12]:

```
library(R.matlab)
#final_proj_data <- readMat("finalproject_data.mat")
return_dat <- final_proj_data$returns
data_mat <- data.matrix(return_dat)

#Perform standardization
for(row in 1:422){
  data_mat[row,] = data_mat[row,] - mean(data_mat[row,])
  data_mat[row,] = data_mat[row,]/sd(data_mat[row,])
}

#Correlation matrix
rho_6b = (1/501)*data_mat%*%t(data_mat)

#Eigenvalues
lambda_6b <- eigen(rho_6b)$values
D <- diag(lambda_6b)

#Create density
interval_6b <- c(seq(0,105,by=0.05))
histo_6b <- hist(lambda_6b, plot=FALSE, breaks=interval_6b)
density_6b <- c(histo_6b$counts)
experiment_6b = density_6b/length(lambda_6b)
```

In [13]:

```
#Create Marcenko-Pastur density
m=422
n=502
Q = n/m;
lambda_plus = (1+sqrt(1/Q))^2
lambda_minus = (1-sqrt(1/Q))^2
lambda_MP = c(seq(lambda_minus,4,by=0.05))

scalar = Q/2/pi
a = lambda_plus - lambda_MP
b = lambda_MP - lambda_minus
c = sqrt(as.complex(a*b))

densMP = scalar * c / lambda_MP
densMP = densMP/sum(densMP)
plot(interval_6b[1:2100],experiment_6b, type='l', col='darkmagenta', xlim=c(0,10))
lines(lambda_MP, densMP, type='l', col='coral')
```

```
Warning message in xy.coords(x, y):
“imaginary parts discarded in coercion”
```



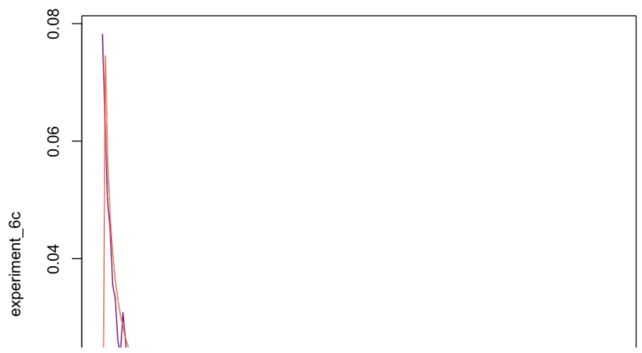
Although shapes are generally the same, for the values especially in the first half of interval, the density doesn’t agree too much as it flattens throughout the interval.

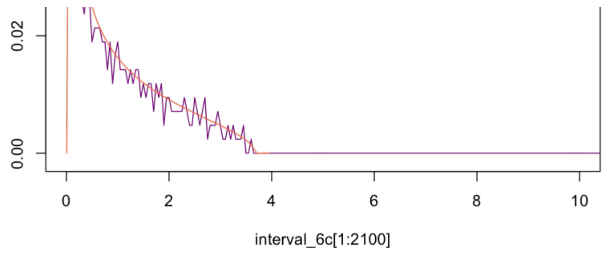
**Now take the matrix Y of standardize returns used in the previous section and apply a (di↵erent) random permutation to each time series of returns so as to remove any time correlation in the returns. Compute the correlation matrix again, and plot the distribution of its eigenvalues. Compare this with the Marcenko-Pastur distribution; how has it changed? How might you use this to determine if your returns are iid?**

In [14]:

```
# Create random permutations
for(row in 1:422){
  data_mat[row,] <- sample(data_mat[row,])
}

#Correlation matrix
rho_6c = (1/501)*data_mat%*%t(data_mat)

#Compute eigenvalues of the matrix
lambda <- eigen(rho_6c)$values

#Create density
interval_6c <- c(seq(0,105,by=0.05))
histo_6c <- hist(lambda, plot=FALSE, breaks=interval_6c)
density_6c <- c(histo_6c$counts)
experiment_6c = density_6c/length(lambda)

plot(interval_6c[1:2100],experiment_6c, type='l', col='darkmagenta', xlim=c(0,10))
lines(lambda_MP, densMP, type='l', col='coral')
```

```
Warning message in xy.coords(x, y):
“imaginary parts discarded in coercion”
```





Compared to 6.2, this seems to perform better by removing time correlation in the returns. From this, we see that by randomizing each series in the data independently (using permutation), the observed result aligns with theoretical Marcenko-Pastur one. This indicates that obversed returns are actually not IID – correlations existin the data play an important role.