VBA: Class 1

```
Function PriceBond(y, face, couponRate, m)
 '<Calculate the bond value>
    price = 0
    For i = 1 To m
        price = ((1 + y) ^ -i) + price
    Next i
    PriceBond = price * face * couponRate + face * ((1 + y) ^ -m) '<Return the bond value>
End Function

Function MyMatMult(vec, mat)
    '<Perform matrix multiplication>
    Dim Ans(0 To 2) As Integer
    For i = 0 To 2
        For j = 0 To 2
            Ans(j) = Ans(j) + vec(i) * mat(i, j)
        Next j
    Next i
    MyMatMult = Ans '<Return the answer>
End Function

'Have this code run for start=1 and finish=15 values


Function FizzBuzz(start, finish)

  n = finish - start + 1

  ReDim charvec(0 To n - 1)

  For i = 0 To (n - 1)
      If (i + 1) Mod 3 = 0 Then
          charvec(i) = "fizz"
      ElseIf (i + 1) Mod 5 = 0 Then
          charvec(i) = "buzz"
      Else
          charvec(i) = i + 1
      End If
      If (i + 1) Mod 15 = 0 Then
          charvec(i) = "fizzbuzz"
      End If
  Next i
  FizzBuzz = charvec

End Function
```

Class 2

```
Function PriceBond(y, face, couponRate, m, Optional ppy = 1)

    Dim k As Double
    k = ppy * m
    ReDim price(k) As Double
    Dim totalprice As Double
    totalprice = 0

        For i = 1 To k
            price(i) = (1 + y / ppy) ^ (-i)
            totalprice = totalprice + price(i)
        Next i

    PriceBond = totalprice * couponRate / ppy * face + face * price(k)

End Function

'Have this code run for any start and finish values
Function FizzBuzz(start, finish)

  n = finish - start + 1

  ReDim charvec(0 To n - 1)

  For i = 0 To (n - 1)
      If (i + 1) Mod 3 = 0 Then
          charvec(i) = "fizz"
      ElseIf (i + 1) Mod 5 = 0 Then
          charvec(i) = "buzz"
      Else
          charvec(i) = i + 1
      End If
      If (i + 1) Mod 15 = 0 Then
          charvec(i) = "fizzbuzz"
      End If
  Next i
  FizzBuzz = charvec

End Function


Function MyMatMult(vec, mat)

    m = UBound(vec, 1)
    m = UBound(mat, 1)
    n = UBound(mat, 2)
    ReDim out(n) As Double
    For i = 0 To m
        For j = 0 To n
            out(j) = out(j) + vec(i) * mat(i, j)
        Next j
    Next i
    MyMatMult = out '<Return the answer>
End Function
```

```
Function MyTripDataObj()
    Dim d(0 To 10, 0 To 1, 0 To 2, 0 To 3)
Data = Range("C38:Z48").Value
For r = 0 To 10
    For c = 0 To 1
        For k = 0 To 2
            For g = 0 To 3
            d(r, c, k, g) = Data(r, c + (k - 1) * 8 + (g - 1) * 2)
            Next g
        Next k
    Next c
Next r
    MyTripDataObj = d(r, c, k, g)
End Function
Sub Macro5()
'
' Macro5 Macro
'

'
    Range("B2:E2").Select
    Selection.AutoFilter
    ActiveSheet.Range("$B$2:$E$869").AutoFilter Field:=1, Criteria1:="<43190"
    Range("G28").Select
    ActiveSheet.Paste
    Selection.NumberFormat = "m/d/yy"
End Sub


' Create a function that returns the data for the range name that is passed to it.
' For example getNamedRange("xVector")

Function getNamedRange(name)
x = Range(name).Value
getNamedRange = x
End Function
```

```
Class 3
Option Explicit
Option Base 1

Function PriceBond(y, face, couponRate, m, Optional ppy = 1)

    Dim k As Double
    k = ppy * m
    ReDim price(k) As Double
    Dim totalprice As Double
    Dim i As Double
    totalprice = 0

        For i = 1 To k
            price(i) = (1 + y / ppy) ^ (-i)
            totalprice = totalprice + price(i)
        Next i

    PriceBond = totalprice * couponRate / ppy * face + face * price(k)

End Function

Function MyMatMult(vec, mat)
    Dim m As Integer
    Dim n As Integer
    Dim r As Integer
    Dim c As Integer
    m = UBound(vec, 1)
    m = UBound(mat, 1)
    n = UBound(mat, 2)
    ReDim out(n) As Double
    For c = 1 To n
        For r = 1 To m
            out(c) = out(c) + vec(r) * mat(r, c)
        Next r
    Next c
    MyMatMult = out
End Function

Function FizzBuzz(start, finish)
   Dim n As Integer
   Dim i As Integer
   n = finish - start + 1

   ReDim charvec(0 To n - 1)

   For i = 0 To (n - 1)
        If (i + 1) Mod 3 = 0 Then
            charvec(i) = "fizz"
        ElseIf (i + 1) Mod 5 = 0 Then
            charvec(i) = "buzz"
        Else
            charvec(i) = i + 1
        End If
        If (i + 1) Mod 15 = 0 Then
            charvec(i) = "fizzbuzz"
        End If
   Next i
   FizzBuzz = charvec
End Function
```

```
Function MyTripDataObj()
Dim data As Variant
Dim i_s As Integer
Dim i_l As Integer
Dim i_p As Integer
Dim i_t As Integer
Dim d(1 To 11, 1 To 2, 1 To 3, 1 To 4)
data = Range("C38:Z48").Value
For i_s = 1 To 11
   For i_l = 1 To 2
      For i_p = 1 To 3
         For i_t = 1 To 4
         d(i_s, i_l, i_p, i_t) = data(i_s, i_l + (i_p- 1) * 8 + (i_t - 1) * 2)
         Next i_t
      Next i_p
   Next i_l
Next i_s
MyTripDataObj = d(i_s, i_l, i_p, i_t)
End Function
```

R language : Class 4

```
getBondPrice = function(y, face, couponRate, m, ppy=1){
  pvcfsum=0
  cf= face * couponRate
  for(t in 1:(m*ppy)){
    pv=(1+y/ppy)^(-t)
    pvcf = pv*cf
    pvcfsum=pvcfsum+pvcf
  }

  pvcfsum = pvcfsum/ppy + face*pv
  return(pvcfsum)
}

getBondDuration = function(y, face, couponRate, m){
  pvcfsum=0
  pvcftsum=0
  cf= face * couponRate
  for(t in 1:m){
    pv=(1+y)^(-t)
    pvcf = pv*cf
    pvcft=pvcf*t
    pvcfsum=pvcfsum+pvcf
    pvcftsum=pvcftsum+pvcft
    print(pvcfsum)
  }
  pvcfsum=pvcfsum+face*pv
  pvcftsum=pvcftsum+face*pv*t
  duration= pvcftsum/pvcfsum
  return(duration)
}

getBPV = function(y, face, couponRate, m,ppy = 1){
  pvcfsum=0
  cf= face * couponRate
  for(t in 1:(m*ppy)){
    pv=(1+y/ppy)^(-t)
    pvcf = pv*cf
    pvcfsum=pvcfsum+pvcf
  }
  pvcfsum = pvcfsum/ppy + face*pv

  newpvcfsum=0
  cf= face * couponRate
  for(t in 1:(m*ppy)){
    newpv=(1+(y+0.0001)/ppy)^(-t)
    newpvcf = newpv*cf
    newpvcfsum=newpvcfsum+newpvcf
  }

  newpvcfsum = newpvcfsum/ppy + face*newpv
  BPV=newpvcfsum-pvcfsum
  return(BPV)
}
```

```
getStockData = function (MYSTOCKNAME){
  install.packages("quantmod")
  library(quantmod)
  mystock = getSymbols('MYSTOCKNAME', auto.assign = F)
  return(mystock)
}

FizzBuzz = function(start,finish){
 n = finish-start+1
  v = vector(mode = "character",length = n)
  for (i in start : finish){
    index = i-start+1
    if (i%%15==0){
      v[index] = "fizzbuzz"
    } else {
      if(i%%3==0){
        v[index] = "fizz"
      }else if(i%%5==0){
        v[index] = "buzz"
      }else{
        v[index]  =i
      }
    }
  }
  return(v)
}

getStockData2CSV = function (MYSTOCKNAME,MYFILENAME){
  x = getStockData(MYSTOCKNAME)
  MYFILENAME = "SOMEFILENAME.csv"
  write.csv(x = x,file = MYFILENAME)
  return(x)
}

MyMatMult = function(vec,mat){
  ans = vector(mode = "numeric",length = 3)
  for (c in 1:3){
    for (r in 1:3){
      ans[c]<-ans[c]+vec[r]*mat[r,c]
    }
  }
  print(ans)
  return(ans)
}
```

Class 5

```
MatMult2 = function(vec,mat){
  nRowMat = dim(mat)[1]
  nColMat = dim(mat)[2]
  nRowVec = dim(vec)[1]
  nColVec = dim(vec)[2]

  ans = matrix(0,nRowVec,nColMat)
  for (i in 1:nColMat){
    ans[,i] = sum(vec*mat[,i])
  }
  return(ans)
}

MatMult1 = function(vec,mat){

  nRowMat = dim(mat)[1]
  nColMat = dim(mat)[2]
  nRowVec = dim(vec)[1]
  nColVec = dim(vec)[2]

  ans = matrix(0,nRowVec,nColMat)

  for (i in 1:nColMat){
    ans[,i] = sum(vec*mat[,i])
  }
  print(ans)
  return(ans)
}

TMAT1 = function(vec1,vec2){
  Matr=length((unique(vec1)))
  Matc=length((unique(vec2)))
  mat = matrix(0,Matr,Matr)
  Alph="ABCDEFGHIJKLMNOPQRSTUVWXYZ"
  l=length(vec1)

for (i in 1:l){
    TranR = unlist(gregexpr(vec1[i],Alph))
    TranC = unlist(gregexpr(vec2[i],Alph))
    mat[TranR,TranC]=mat[TranR,TranC]+1
  }
  for (i in 1:Matr){
    mat[i,]=mat[i,]/sum(mat[i,])
  }
  print(mat)
  return(mat)
}

MatMult = function(vec,mat){
  nRowMat = dim(mat)[1]
  nColMat = dim(mat)[2]
  nRowVec = dim(vec)[1]
  nColVec = dim(vec)[2]

  ans = matrix(0,nRowVec,nColMat)
  for (i in 1:nColMat){
    ans[,i] = sum(vec*mat[,i])
  }
  print(ans)
  return(ans)
}
```

```
getBondPrice = function(y, face, couponRate, m,
ppy=1){

  PV=seq(1,ppy*m)
  PVt=(1+y/ppy)^(-PV[])
  bondPrice = sum(PVt)*face*couponRate/
ppy+PVt[ppy*m]*face
  print(bondPrice)
  return (bondPrice)
}

getBondDuration = function(y, face, couponRate, m){

  cf= face * couponRate
  PV=seq(1,m)
  PVt=(1+y)^(-PV[])

  x=sum(PVt)
   pvcfsum = sum(PVt)*cf+PVt[m]*face
    q=face*couponRate+face

    pvcftsum = PVt[m]*face*m+sum(PVt*PV)*cf

  duration= pvcftsum/pvcfsum

  print(duration)
  return(duration)
}

getBondPriceYC = function(y, face, couponRate, m){
  cf= face * couponRate
  if (y == YC){
  PV=seq(1,m)
  PVt=(1+YC)^(-PV[])
  bondprice = (sum(PVt)*couponRate+PVt[m])*face
    print(bondprice)
  }else{
    PV=seq(1,m)
    PVt=(1+y)^(-PV[])
    bondprice = (sum(PVt)*couponRate+PVt[m])*face
    print(bondprice)
  }
    return (bondprice)
}

FizzBuzz = function(start,finish){
  n = finish-start+1
  arr = c(start:finish)
  v = vector(mode = "character",length = n)
  fb_checked = ifelse((arr%%15)==0,15,arr)
  f_checked = ifelse((fb_checked%%3)==0
&fb_checked!=15,3,fb_checked)
  b_checked = ifelse((f_checked%%5)==0&fb_checked!
=15,5,f_checked)
  v= ifelse(b_checked==15,"fizzbuzz",b_checked)
  v= ifelse(b_checked==3,"fizz",v)
  v= ifelse(b_checked==5,"buzz",v)

  v= gsub(" ", "", v)
    return(v)
}
```

```
#install.packages("quantmod")

library(quantmod)

getStockData = function(symbol){
  mydata = quantmod::getSymbols(Symbols =
symbol,auto.assign = F)
  prices = mydata[,6]
  return(prices)
}

prices = getStockData('gs')
class(prices)
pricevec = as.numeric(prices) # NEED TO CHANGE TO
MATRIX OR NUMERIC

# Example of part of the work (using the wrong value for
n):
n = length(pricevec)
ratiovec = pricevec[2:n]/pricevec[1:(n-1)]

getReturns = function(pricevec){
  n = length(pricevec)
  ratiovec = pricevec[2:n]/pricevec[1:(n-1)]
  returns =  ratiovec-1
  return(returns)
}
```

Class 6

```
Datavec2DataFrame = 
function(vec,fieldname1,fieldname2,splitstr){
  mynewlist = stringr::str_split(myvec,"_")
  mynewvec = unlist(mynewlist)
  mynewvec[1]
  location= c(mynewvec[1],mynewvec[3],
  mynewvec[5], mynewvec[7], mynewvec[9],
  mynewvec[11])
  weather= c(mynewvec[2],mynewvec[4],
  mynewvec[6], mynewvec[8], mynewvec[10],
  mynewvec[12])
  df=data.frame(location,weather)
  return(df)
}

MyLM = function(formula,data){
  myformulaStr = formula
  class(myformulaStr)
  fit=lm(formula = myformulaStr,data=data)
    return(fit)
}

PlotSLRCI = function(fit,data){
  plot(data)
  conf=confint(fit1)
  upper=abline(a = conf[1,1], b = conf[2,1], col = 'red')
  lower=abline(a = conf[1,2], b = conf[2,2], col = 'yellow')
  mid=abline(fit,col='green')

}

TMAT1 = function(vec1,vec2){
  myfullset= union(rLast,rNow)
  myuniquesubset = unique(myfullset)
  mysortedset = sort(myuniquesubset)

  states=mysortedset
  n = length(states)

  mat=matrix(NA,n,n)

  for(i in 1:n){
    for(j in 1:n){
      mat[i,j] = sum(rLast == states[i] & rNow == states[j])
    }
    mat[i,]=mat[i,]/sum(mat[i,])
  }
  mat = mat
  return(mat)
}

Forecast_nPeriod= function(vec,tmat,n){
  # Use a loop to multiply a t tmat by itself n times
  out <- list(vec%*%tmat)
  if (n==1) return(out)
  P <- vec
  for (i in 1:n) out[[i]] <- (P <- P %*% tmat)
  out=out[[i]]
  return(out)
}
```

```
Forecast_nPeriod_Recursive = function(vec,mat,n){
  if(n==0){
    return(vec)
  }
  else
  {
    Forecast_nPeriod_Recursive(vec%*%mat,mat,n-1)
  }
}

getBondDuration = function(y, face, couponRate, 
m,ppy = 1){

  cf= face * couponRate
  PV=seq(1,ppy*m)
  PVt=(1+y/ppy)^(-PV[])


  x=sum(PVt)
  pvcfsum = sum(PVt)*cf/ppy+PVt[ppy*m]*face
  q=face*couponRate+face

  pvcftsum = PVt[ppy*m]*face*m+sum(PVt*PV/ppy)*cf/
ppy

  print(pvcftsum)
  duration= pvcftsum/pvcfsum

  print(duration)
  return(duration)
}

getReturns = function(pricevec, lag = 1){

  pricevec=x
  n=length(pricevec)

  lgstring=vector(mode='numeric', length=n)

  returns=(pricevec[(1+lag):n]/pricevec[1:(n-lag)])-1
  print(returns)
  return(returns)
}

PercentVaR = function(r,alpha){
  Range=qnorm(alpha)
  VaR=quantile(r,1-alpha)
  hist(r)
  out = abs(VaR)
    return(out)
}
```

```
ES = function(losses,alpha = NULL,VaR = NULL){
  # losses = positively stated loss values
  # alpha = risk level e.g. 99%
  # VaR = either a dollar value or a percent
  #
  # out = the Expected Shortfall
  #
  # If !(is.null(VaR)), out = average of losses > VaR
  # If is.null(VaR),
  # Identify the percentile loss matching alpha
  # Redefine VaR as the percentile loss
  # Then return out = losses > VaR

  if (!(is.null(VaR))){
   aL = abs(mean(losses[losses>VaR]))
  } else if (is.null(VaR)){
    VaRper = quantile(losses,alpha)
    aL = abs(mean(losses[losses>VaRper]))
  }
  out = aL

    return(out)
}
```

Class 7

```
MyStep = function(startModel, finishModel, backward_or_forward, AIC_or_BIC){

  scope = list(lower = formula(startModel), upper = formula(finishModel))

  if (backward_or_forward == 'forward'){
    if (AIC_or_BIC == 'AIC'){
      forwardAIC = step(startModel, scope, direction = "forward", k = 2)
      Mstp = forwardAIC
    } else if (AIC_or_BIC == 'BIC'){
      n = startModel$df.residual + 1
      forwardBIC = step(startModel, scope, direction = "forward", k = log(n))
      Mstp = forwardBIC
    }
  } else if (backward_or_forward == 'backward'){
    if (AIC_or_BIC == 'AIC'){
      backwardAIC = step(finishModel, scope, direction = "backward", k = 2)
      Mstp = backwardAIC
    } else if (AIC_or_BIC == 'BIC'){
      n = startModel$df.residual + 1
      backwardBIC = step(finishModel, scope, direction = "backward", k = log(n))
      Mstp = backwardBIC
    }
  }
  return(Mstp)
}

library(ggplot2)
PlotBetaFits = function(fit,mydata,nsim){
  plot(mydata[1:2])
  library(mvtnorm)
  cFit=coefficients(fit)
  vFit=vcov(fit)
  mysim = mvtnorm::rmvnorm(nsim,cFit,vFit)
  for (i in 1:nsim){
    abline(a = mysim[i,1], b = mysim[i,2])
  }
}

TMAT1= function(vec1,vec2,weights = 1){

  uniAlph = sort(unique(union(vec1,vec2)))

  n = length(uniAlph)
  mat = matrix(NA,n,n)

  for (r in 1:n){
    for (c in 1:n){
      mat[r,c] = sum(weights*(vec1 == uniAlph[r] & vec2 == uniAlph[c]))
    }
    mat[r,] = mat[r,]/sum(mat[r,])
  }
  out = mat
  return(out)
}
```

```
library(shiny)
library(quantmod)

ui <- fluidPage(
  sidebarLayout(
    sidebarPanel(
      textInput(inputId = 'Company',
                label = 'Please enter a company name'),
      numericInput(inputId = 'Lag',
                   label = 'Please enter your lag',
                   value = 50),
      numericInput(inputId = 'ConfidenceLevel',
                   label = 'Please enter your confidence level',
                   value = 0.95)
    ),
    mainPanel(
      textOutput(outputId = 'mymessage'),
      plotOutput(outputId = 'myplot')
    )
  )
)

server <- function(input, output, session) {
  output$mymessage = renderText("Note: VaR line is in
purple and Expected Shortfall line is in red")
  output$myplot = renderPlot({
    Syb = quantmod::getSymbols(Symbols =
input$Company, auto.assign = F)
    prc = as.numeric(Syb[,6])
    n = length(prc)
    return = (prc[(input$Lag+1):n]/prc[1:(n-input$Lag)])-1
    hist(return, main = input$Company)

    VaR = quantile(return,1-input$ConfidenceLevel)
    abline(v = VaR, col = 'purple')

    ExpSf = mean(return[return<VaR])
    abline(v = ExpSf, col = 'red')
  })
}

shinyApp(ui, server)
```

Class 8

```
# Question 1: qShiny_Stock ####
#install.packages("shinydashboard")
#install.packages('quantmod')
library(shiny)
library(quantmod)

ui <- fluidPage(
  titlePanel("qShiny_Stock    (Red Line = VaR    Blue Line
= ES)"),
  sidebarLayout(
    sidebarPanel(
      textInput(inputId = 'Sym',label = 'Select a Stock
Symbol',value=''),
      numericInput(inputId = 'Lag',label = 'Lag (Default:
5)',value=5),
      numericInput(inputId = 'CL',label = 'Confidence
Level (Default: 0.90)',value=0.90),
      numericInput(inputId = 'WS',label = 'Window Size
(Default: 10)',value=10),
      selectInput(inputId='functions',label =
'function',choices=c('Minimum'='min','Maximum'='max'
,'Standard Deviation'='sd','Variance'='var'))
    ),
    mainPanel(
      tabsetPanel(
        type = "tabs",
        tabPanel("Plot",plotOutput(outputId = 'myplot')),
        tabPanel("Rolling",plotOutput(outputId = 'rolling'))
      )
    )
  )
)

server <- function(input,output,session){
  Obj_Returns=reactive({
    req(input$Sym)
    rawdata=quantmod::getSymbols(Symbols =
input$Sym,auto.assign = F)
    pricesvc = as.numeric(rawdata[,6])
    return(pricesvc)
  })

  output$myplot = renderPlot({
    pricesvc = Obj_Returns()
    n = length(pricesvc)
    k=input$Lag
    returns = (pricesvc[1:(n-k)]/pricesvc[(1+k):n])-1
    VaR=quantile(returns,1-input$CL)
    ES=mean(returns[returns<VaR])
    hist(returns)
    abline(v= VaR,lwd=2,col="red")
    abline (v= ES,lwd=2,col="blue")
  })
output$rolling = renderPlot({
    pricesvc = Obj_Returns()
    ws=input$WS
    f=input$functions
    n=length(pricesvc)
    nw=n-ws+1
    out=rep(0,nw)
    m=1:ws
    if(f=='min'){
      for(i in 1:nw){
        out[i]=min(pricesvc[m])
        m=m+1
      }
    }else if(f=='max'){
      for(i in 1:nw){
        out[i]=max(pricesvc[m])
        m=m+1
      }
    }else if(f=='sd'){
      for(i in 1:nw){
        out[i]=sd(pricesvc[m])
        m=m+1
      }
    }else if(f=='var'){
      for(i in 1:nw){
        out[i]=var(pricesvc[m])
        m=m+1
      }
    }
    plot(out)
  })
}
shinyApp(ui,server)

# Question 2: qShiny_MLR ####
library(shiny)
library(quantmod)
library(car)
ui <- fluidPage(
  titlePanel("qShiny_MLR"),
  sidebarLayout(
    sidebarPanel(
      fileInput("file","Choose a CSV File:",multiple =
TRUE,accept = c("text/csv","text/comma-separated-
values,text/plain",".csv")),
      textInput(inputId = 'formula',label = 'Select a
formula',value='')
    ),

    mainPanel(
      tabsetPanel(
        type = "tabs",
        tabPanel("Table",tableOutput(outputId = 'table')),
        tabPanel("AVPlot",plotOutput(outputId = 'avP')),
        tabPanel("Variance Inflation
Factors",verbatimTextOutput(outputId = "VIF")),
        tabPanel("Influence Plot",plotOutput(outputId =
'IP'))
      )
    )
  )
)
```

```
server <- function(input,output,session){
  output$table <-renderTable({
    req(input$file)
    df<-read.csv(input$file$datapath)
  })

  output$avP <- renderPlot({
    req(input$file)
    req(input$formula)
    df<-read.csv(input$file$datapath)
    myfit = lm(formula = input$formula,data = df)
    car::avPlots(myfit)
  })

  output$VIF <- renderPrint({
    req(input$file)
    req(input$formula)
    df<-read.csv(input$file$datapath)
    myfit = lm(formula = input$formula,data = df)
    car::vif(myfit)
  })
  output$IP <- renderPlot({
    req(input$file)
    req(input$formula)
    df<-read.csv(input$file$datapath)
    myfit = lm(formula = input$formula,data = df)
    car::influencePlot(myfit)
  })

}
shinyApp(ui,server)
```

```
# Question 3: qShiny_Portfolio ####
#install.packages('scales')
library(shiny)
library(quantmod)
library(scales)

ui <- fluidPage(
  titlePanel("qShiny_Portfolio"),
  sidebarLayout(
    sidebarPanel(
      textInput(inputId = 'Sym1',label = 'Select first Stock
Symbol',value=''),
      textInput(inputId = 'Sym2',label = 'Select second
Stock Symbol',value=''),
      textInput(inputId = 'Sym3',label = 'Select third Stock
Symbol',value=''),
      shinyWidgets::autonumericInput(inputId = "ws1",
label = "Weight for Stock 1(Default: 1/3):", value =
1/3,decimalPlaces = 4),
      shinyWidgets::autonumericInput(inputId = "ws2",
label = "Weight for Stock 2(Default: 1/3):", value =
1/3,decimalPlaces = 4),
      shinyWidgets::autonumericInput(inputId = "ws3",
label = "Weight for Stock 3(Default: 1/3):", value =
1/3,decimalPlaces = 4)
    ),

    mainPanel(
      tabsetPanel(
        type = "tabs",
        tabPanel("Histogram of Portfolio
Returns",plotOutput(outputId = 'myplot')),
        tabPanel("Raw Data",tableOutput(outputId =
'table')),
        tabPanel("Report",tableOutput(outputId = 'table2'))
      )
    )
  )
)

server <- function(input,output,session){
  PortfolioVol = function(S, w = NULL){
    n = dim(S)[1]
    if(is.null(w)){
      w = rep(1/n,n)
    }
    wm =matrix(w,length(w),1)
    transpose_w = t(wm)
    out = sqrt(transpose_w %*% S %*% wm)
    return(out)
  }
```

```
Stocks_Merge=reactive({
    req(input$Sym1)
    req(input$Sym2)
    req(input$Sym3)
    x = quantmod::getSymbols(input$Sym1,auto.assign =
F,from = "2017-01-01", to = "2018-01-01")
    y = quantmod::getSymbols(input$Sym2,auto.assign =
F,from = "2017-01-01", to = "2018-01-01")
    z = quantmod::getSymbols(input$Sym3,auto.assign =
F,from = "2017-01-01", to = "2018-01-01")

    Pr1 = quantmod::periodReturn(x[,6],period = 'daily')
    Pr2 = quantmod::periodReturn(y[,6],period = 'daily')
    Pr3 = quantmod::periodReturn(z[,6],period = 'daily')

    PR = merge.xts(Pr1,Pr2,Pr3)
    df_PR = as.data.frame(PR)
    return(PR)
  })

 Stocks_Weights = reactive({
    req(input$ws1)
    req(input$ws2)
    req(input$ws3)

    w =
c(as.numeric(input$ws1),as.numeric(input$ws2),as.num
eric(input$ws3))
    wm = matrix(w,length(w),1)

    return(wm)
  })

  Stocks_Table = reactive({
    PR = Stocks_Merge()
    df_PR= as.data.frame(PR)
    mean1 = mean(df_PR$daily.returns)
    mean2 = mean(df_PR$daily.returns.1)
    mean3 = mean(df_PR$daily.returns.2)

    p_return = mean1*input$ws1 + mean2*input$ws2+
mean3*input$ws3
    S = cov(PR)
    w =
c(as.numeric(input$ws1),as.numeric(input$ws2),as.num
eric(input$ws3))
    p_vol =  PortfolioVol(S,w)
    Sharpe_Ratio = p_return/p_vol
    df <- data.frame (Name  = c("Average daily return
Stock 1", "Average daily return Stock 2","Average daily
return Stock 3","Portfolio Return ","Portfolio Volatility
","Sharpe Ratio"),
                      Value = c(mean1,
mean2,mean3,p_return,p_vol,Sharpe_Ratio),
                      Percentage = c(mean1,
mean2,mean3,p_return,p_vol,Sharpe_Ratio)
    )
    df$Value<-format(round(df$Value,6),nsmall=6)
    df$Percentage <- percent(df$Percentage,
accuracy=.0001)
    return(df)
  })
  output$table <-renderTable({
    df<-Stocks_Merge()
    df= as.data.frame(df)
    df <- within(df, PortfolioReturn <-
daily.returns*input$ws1 + daily.returns.1*input$ws2+
daily.returns.2*input$ws3)
  })

  output$table2 <-renderTable({
    df= Stocks_Table()
  })

  output$myplot = renderPlot({
    PR = Stocks_Merge()
    df_PR = as.data.frame(PR)
    df_PR <- within(df_PR, PortfolioReturn <-
daily.returns*input$ws1 + daily.returns.1*input$ws2+
daily.returns.2*input$ws3)
    hist(df_PR$PortfolioReturn)

  })

}
shinyApp(ui,server)
```

Class9

```
library(dplyr)
library(shiny)

ui <- fluidPage(

  sidebarLayout(
    sidebarPanel(

      fileInput(inputId = 'dsetin', label= 'Upload a data
set', accept = 'csv'),
      selectInput(inputId = 'sltfield',
                  label = 'Field Filter',
                  choices = '',
                  selected = ''),

      selectInput(inputId = 'rlship',
                  label = 'Logical Operation',
                  choices = c("less than","equal to","greater
than"),
                  selected = "less than"),

      numericInput(inputId = 'valselect',
                   label = 'Value Number',
                   value = 5),

      selectInput(inputId = 'col',
                  label = 'Columns Selection',
                  choices = '',
                  selected = '',
                  multiple = T),

      selectInput(inputId = 'sum',
                  label = 'Variable',
                  choices = '',
                  selected = ''),

      selectInput(inputId = 'stats',
                  label = 'Summary Statistics',
                  choices = c("min","mean","max","sd"),
                  selected = ""),

      selectInput(inputId = 'group',
                  label = 'Group',
                  choices = '',
                  selected = '')
    ),

    mainPanel(
      tabsetPanel(type = "tabs",
                  tabPanel("Filter", tableOutput(outputId =
'table1')),
                  tabPanel("Field", tableOutput(outputId =
'table2')),
                  tabPanel("Summary", tableOutput(outputId =
'table3'))
      )
    )
  )
)
```

```
server <- function(input, output, session) {

  Obj_data = reactive({
    req(input$dsetin)
    df = read.csv(file = input$dsetin$datapath)
  })

  Obj_fieldnames = reactive({
    names(Obj_data())
  })

  observe({
    req(input$dsetin$datapath)
    updateSelectInput(session = session,
                      inputId = 'sltfield',
                      choices = Obj_fieldnames(),
                      selected = '')
  })
  output$table1 = renderTable({
    Obj_filtered()
  })
  Obj_filtered = reactive({
    req(input$sltfield)
    if (input$rlship == 'less than'){
      Obj_data() %>%
        filter(get(input$sltfield) < input$valselect)
    } else if (input$rlship == 'equal to'){
      Obj_data() %>%
        filter(get(input$sltfield) == input$valselect)
    } else if (input$rlship == 'greater than'){
      Obj_data() %>%
        filter(get(input$sltfield) > input$valselect)
    }
  })
  output$table2 = renderTable({
    Obj_select()
  })

  observe({
    req(input$dsetin$datapath)
    updateSelectInput(session = session,
                      inputId = 'col',
                      choices = Obj_fieldnames(),
                      selected = '')
  })

  Obj_select = reactive({
    req(input$col)
    if (input$rlship == 'less than'){
      Obj_data() %>%
        filter(get(input$sltfield) < input$valselect) %>%
        select(input$col)
    } else if (input$rlship == 'equal to'){
      Obj_data() %>%
        filter(get(input$sltfield) == input$valselect) %>%
        select(input$col)
    } else if (input$rlship == 'greater than'){
      Obj_data() %>%
        filter(get(input$sltfield) > input$valselect) %>%
        select(input$col)
    }
  })
```

```
# For Step 3
  observe({
    req(input$dsetin$datapath)
    req(input$col)
    updateSelectInput(session = session,
                      inputId = 'group',
                      choices = input$col,
                      selected = '')
  })

  observe({
    req(input$dsetin$datapath)
    req(input$col)
    updateSelectInput(session = session,
                      inputId = 'sum',
                      choices = input$col,
                      selected = '')
  })

  output$table3 = renderTable({
    Obj_stat()
  })

  Obj_stat = reactive({
    req(input$group)
    x = switch(input$stats,
               "min" = min,
               "max" = max,
               "sd" = sd,
               "mean" = mean)

    if (input$rlship == 'less than'){
      req(input$group)
      req(input$stats)
      Obj_data() %>%
        filter(get(input$sltfield) < input$valselect) %>%
        select(input$col) %>%
        group_by(get(input$group)) %>%
        summarise(minimum = x(get(input$sum)))
    } else if (input$rlship == 'equal to'){
      Obj_data() %>%
        filter(get(input$sltfield) == input$valselect) %>%
        select(input$col) %>%
        group_by(get(input$group)) %>%
        summarise(minimum = x(get(input$sum)))
    } else if (input$rlship == 'greater than'){
      Obj_data() %>%
        filter(get(input$sltfield) > input$valselect) %>%
        select(input$col) %>%
        group_by(get(input$group)) %>%
        summarise(minimum = x(get(input$sum)))
    }
  })

}

shinyApp(ui, server)
```

Python : Class 11 & 12

```
def getBondDuration(y, face, couponRate, m, ppy = 1):
    pvcfsum=0
    pvcftsum=0
    cf= face * couponRate
    for t in range(1,m+1):
        pv=(1+y)**(-t)
        pvcf = pv*cf
        pvcft=pvcf*t
        pvcfsum=pvcfsum+pvcf
        pvcftsum=pvcftsum+pvcft

    pvcfsum=pvcfsum+face*pv
    pvcftsum=pvcftsum+face*pv*t
    duration= pvcftsum/pvcfsum
    return(duration)
```

```
def getBondDuration(y, face, couponRate, m, ppy = 1):
    pvcfsum=0
    pvcftsum=0
    cf= face * couponRate
    PV=np.arange(1,ppy*m+1)
    pv=(1+y/ppy)**(-PV)
    pvcf=pv*cf
    pvcftsum=pvcf@PV
    pvcfsum=sum(pvcf)
    pvcfsum=pvcfsum+face*pv
    pvcftsum=pvcftsum+face*pv*PV[m*ppy-1]
    duration= pvcftsum[m*ppy-1]/pvcfsum[m*ppy-1]
    return(duration)
```

```
def FizzBuzz(start, finish):
    v=list(range(start,finish+1))
    n=finish-start+1
    for i in range(n):
        if (start+i) % 3 == 0 and (start+i) % 5 == 0:
            v[i]='fizzbuzz'
        elif (start+i) % 3 == 0:
            v[i]='fizz'
        elif (start+i) % 5 == 0:
            v[i]='buzz'
        else:
            v[i]=start+i
    return(v)
```

```
def FizzBuzz(start, finish):
    numvec = np.arange(start,finish+1)
    objvec = np.array(numvec,dtype = object)

    list3 = numvec%3==0
    objvec[list3]='fizz'
    list5 = numvec%5==0
    objvec[list5]='buzz'
    list15 = numvec%15==0
    objvec[list15]='fizzbuzz'
    return(objvec)
```

```
def getBondPrice(y, face, couponRate, m, ppy=1):
    pvcfsum=0
    cf= face * couponRate
    for t in range(1,m*ppy+1):
        pv=((1+y/ppy))**(-t)
        pvcf = pv*cf
        pvcfsum+= pvcf
    pvcfsum= pvcfsum/ppy+pv*face
    return (pvcfsum)
```

```
def getBondPrice(y, face, couponRate, m, ppy=1):
    PV=np.arange(1,ppy*m+1)
    PVt=(1+y/ppy)**(-PV)
    bondPrice= sum(PVt)*face*couponRate/ppy+PVt[ppy*m-1]*face
    return(bondPrice)
```

```
def getBondPrice_E(face, couponRate, m, yc):
    pvcfsum=0
    cf= face * couponRate
    n=len(yc)
    for t in range(n):
        pv=(1+yc[t])**(-(t+1))
        pvcf = pv*cf
        pvcfsum+= pvcf
    pvcfsum= pvcfsum+pv*face
    return (pvcfsum)
```

```
def getBondPrice_E(face, couponRate, m, yc):
    pvcf=0
    pvcfsum=0
    for i,x in enumerate(yc):
        pvcf=(1+x)**-(i+1)
        pvcfsum += pvcf
    bondprice = pvcfsum*face*couponRate+face*pvcf

    return (bondprice)
```

```
def getBondPrice_Z(face, couponRate, times, yc):
    pvcfsum=0
    cf= face * couponRate
    n=len(yc)
    for t in range(n):
        pv=(1+yc[t])**(-(times[t]))
        pvcf = pv*cf
        pvcfsum+= pvcf
    pvcfsum= pvcfsum+pv*face
    return (pvcfsum)
```

```
def getBondPrice_Z(face, couponRate, times, yc):
    pvcf=0
    pvcfsum=0
    for i,x in zip(times,yc):
        pvcf=(1+x)**(-i)
        pvcfsum += pvcf
    bondprice = pvcfsum*face*couponRate+face*pvcf

    return (bondprice)
```