server.R

library(shiny)

library(quantmod)

library(VGAM)

# Define server logic for random distribution application

shinyServer(function(input, output) {

# acquiring data

dataInput <- reactive({

if (input$get == 0)

return(NULL)

return(isolate({

getSymbols(input$symb,src="yahoo", auto.assign = FALSE)

}))

})

datesInput <- reactive({

if (input$get == 0)

return(NULL)

return(isolate({

paste0(input$dates[1], "::", input$dates[2])

}))

})

returns <- reactive({

if (input$get == 0)

return(NULL)

dailyReturn(dataInput())

})

xs <- reactive({

if (input$get == 0)

return(NULL)

span <- range(returns())

seq(span[1], span[2], by = diff(span) / 100)

})

# tab based controls

output$newBox <- renderUI({

switch(input$tab,

"Charts" = chartControls,

"Model" = modelControls,

"VaR" = helpText("VaR")

)

})

# Charts tab

chartControls <- div(

wellPanel(

selectInput("chart\_type",

label = "Chart type",

choices = c("Candlestick" = "candlesticks",

"Matchstick" = "matchsticks",

"Bar" = "bars",

"Line" = "line"),

selected = "Line"

),

checkboxInput(inputId = "log\_y", label = "log y axis",

value = FALSE)

),

wellPanel(

p(strong("Technical Analysis")),

checkboxInput("ta\_vol", label = "Volume", value = FALSE),

checkboxInput("ta\_sma", label = "Simple Moving Average",

value = FALSE),

checkboxInput("ta\_ema", label = "Exponential Moving Average",

value = FALSE),

checkboxInput("ta\_wma", label = "Weighted Moving Average",

value = FALSE),

checkboxInput("ta\_bb", label = "Bolinger Bands",

value = FALSE),

checkboxInput("ta\_momentum", label = "Momentum",

value = FALSE),

br(),

actionButton("chart\_act", "Add Technical Analysis")

)

)

TAInput <- reactive({

if (input$chart\_act == 0)

return("NULL")

tas <- isolate({c(input$ta\_vol, input$ta\_sma, input$ta\_ema,

input$ta\_wma,input$ta\_bb, input$ta\_momentum)})

funcs <- c(addVo(), addSMA(), addEMA(), addWMA(),

addBBands(), addMomentum())

if (any(tas)) funcs[tas]

else "NULL"

})

output$chart <- renderPlot({

chartSeries(dataInput(),

name = input$symb,

type = input$chart\_type,

subset = datesInput(),

log.scale = input$log\_y,

theme = "white",

TA = TAInput())

})

# Model tab

modelControls <- div(

br(),

sliderInput("n", "Number of bins in histogram",

min = 1, max = 250, value = 30

),

br(),

wellPanel(

selectInput("family", "Model returns as",

choices = c("normal", "double exponential", "t"),

selected = "normal"

),

sliderInput("mu", "Mean",

min = -1, max = 1, value = 0, step = 0.01

),

sliderInput("sigma", "Standard Deviation",

min = 0, max = 0.1, value = 0.05, step = 0.001

),

conditionalPanel(condition = "input.family == 't'",

sliderInput("df", "Degrees of freedom",

min = 2, max = 1000, value = 10

)

)

)

)

ys <- reactive({

if (input$get == 0)

return(NULL)

switch(input$family,

"double exponential" = dlaplace(xs(),

location = input$mu,

scale = input$sigma

),

"normal" = dnorm(xs(),

mean = input$mu,

sd = input$sigma

),

"t" = dt((xs() - input$mu) / input$sigma,

df = input$df) \* sqrt(2 \* length(returns()))

)

})

ks <- reactive({

switch(input$family,

"double exponential" = ks.test(returns(), "plaplace",

input$mu, input$sigma),

"normal" = ks.test(returns(), "pnorm",

input$mu, input$sigma),

"t" = ks.test((returns() - input$mu) / input$sigma, "pt",

input$df)

)

})

output$hist <- renderPlot({

hist(returns(), xlab = "returns", freq = FALSE,

main = paste(input$symb, "Daily Returns:",

input$dates[1], "-", input$dates[2], sep = " "),

breaks = input$n)

lines(xs(), ys(), col = "red")

})

output$ks <- renderText({

paste0("Kolmogorv-Smirnoff statistic: ", ks()$statistic)

})

output$ksp <- renderText({

paste0("P-value for model: ", ks()$p.value)

})

# VaR tab

output$text3 <- renderText({paste0(input$symb, " 3: ", input$tab)})

})

Raw

ui.R

library(shiny)

# Define UI for random distribution application

shinyUI(pageWithSidebar(

headerPanel("Stock Explorer"),

sidebarPanel(

helpText("Select a stock to examine.

Information will be collected from yahoo finance."),

textInput("symb", "Symbol", "GOOG"),

dateRangeInput("dates",

"Compare to historic returns from",

start = "2013-01-01", end = "2013-09-05"),

actionButton("get", "Get Stock"),

br(),

br(),

uiOutput("newBox")

),

# Show a tabset that includes a plot, summary, and table view

# of the generated distribution

mainPanel(

tabsetPanel(

tabPanel("Charts", plotOutput("chart")),

tabPanel("Model", div(h3(textOutput("ks"))),

div(h3(textOutput("ksp"))),

plotOutput("hist")),

tabPanel("VaR", h3(textOutput("text3"))),

id = "tab"

)

)

))