FE8828 Programming Web Applications in Finance

Week 5 Building Financial Applications

Dr. Yang Ye <Email:yy@runchee.com>

Nov 26, 2018

Lecture 10: Building Financial Applications

Starter

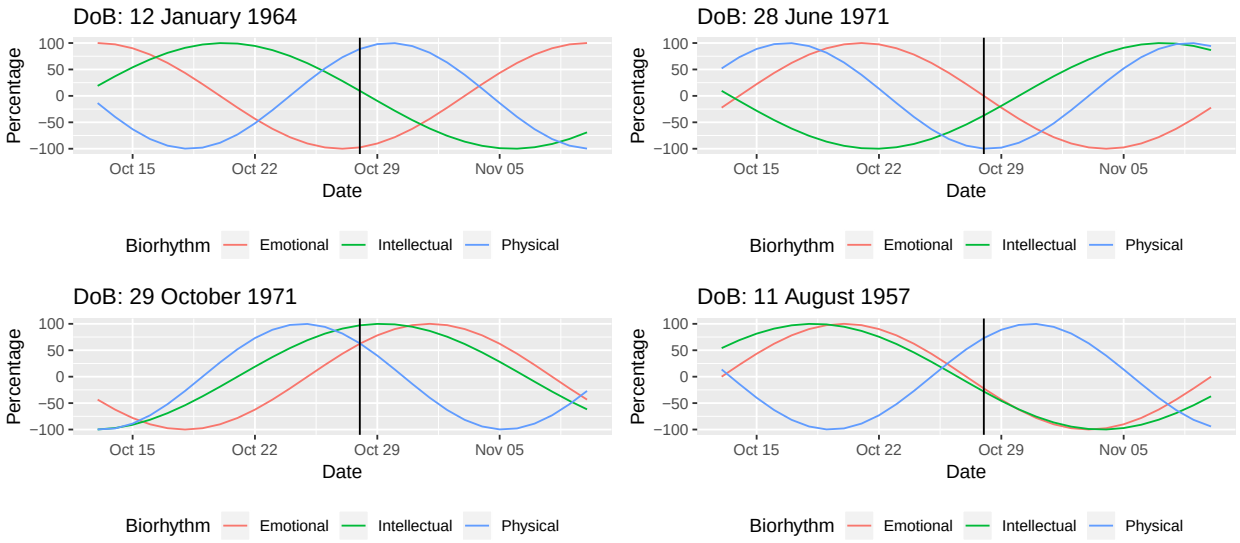
```
# biorhythm.R

library(dplyr)
library(tidyr)
library(ggplot2)

biorhythm <- function(dob, target = Sys.Date()) {
  dob <- as.Date(dob)
  target <- as.Date(target)
  t <- round(as.numeric(difftime(target, dob)))
  days <- (t - 14) : (t + 14)
  period <- tibble(Date = seq.Date(from = target - 15, by = 1, length.out = 29),
                       Physical = sin (2 * pi * days / 23) * 100,
                       Emotional = sin (2 * pi * days / 28) * 100,
                       Intellectual = sin (2 * pi * days / 33) * 100)
  period <- gather(period, key = "Biorhythm", value = "Percentage", -Date)
  ggplot(period, aes(x = Date, y = Percentage, col = Biorhythm)) +
    geom_line() +
    ggtitle(paste("DoB:", format(dob, "%d %B %Y"))) +
    geom_vline(xintercept = as.numeric(target)) +
    theme(legend.position = "bottom")
}
```

Starter - Result

```
# I took four people's birthdays. Hope they are in good mode today.
g1 <- biorhythm("1964-01-12", Sys.Date())
g2 <- biorhythm("1971-06-28", Sys.Date())
g3 <- biorhythm("1971-10-29", Sys.Date())
g4 <- biorhythm("1957-08-11", Sys.Date())
grid.arrange(g1, g2, g3, g4, ncol = 2, nrow = 2)
```



Main course

- We need following packages as a start. Use c() to install multiple packages.

```
install.packages(c("tidyquant", "Quandl", "fOptions", "fExoticOptions", "dygraph", "forecast"))
```

- `tidyquant` is also a collection of packages: `xts`, `quantmod`.
- Please validate option pricing code.
  - For example, I found Asian Option `TurnbullWakemanAsianApproxOption()` in `fExoticOptions` is strangely implemented. Maybe I am wrong.

tidyquant or Quandl?

Determining factors:

- `tidyquant/quantmod` can connect to various services: google, yahoo (still active), av (AlphaAdvantage).
- Quandl only connects to Quandl
- It's subjected to where you can find the data.
  - US ETF on Quandl is a premium service.
  - ETF in Google/AlphaAdvantage is free.

tidyquant or Quandl?

Technical details:

- quantmod returns `xts` object. Quandl returns data frame or `xts`
- xts object is can `collapse` to daily, weekly, monthly price.

Tidyquant/quantmod

```
# library(tidyquant)

# use Google
```

```
getSymbols('SPY', src = 'yahoo', adjusted = TRUE, output.size = 'full')
##
## WARNING: There have been significant changes to Yahoo Finance data.
## Please see the Warning section of '?getSymbols.yahoo' for details.
##
## This message is shown once per session and may be disabled by setting
## options("getSymbols.yahoo.warning"=FALSE).
## [1] "SPY"
str(SPY)
## An 'xts' object on 2007-01-03/2018-10-26 containing:
##   Data: num [1:2977, 1:6] 142 141 141 141 141 ...
##  - attr(*, "dimnames")=List of 2
##   ..$ : NULL
##   ..$ : chr [1:6] "SPY.Open" "SPY.High" "SPY.Low" "SPY.Close" ...
##   Indexed by objects of class: [Date] TZ: UTC
##   xts Attributes:
## List of 2
##  $ src    : chr "yahoo"
##  $ updated: POSIXct[1:1], format: "2018-10-28 23:16:14"

# Sign up with AlphaAdvantage to get a token
# getSymbols('SPY', src = 'av', output.size = 'full', api.key = token_av)
# str(SPY)
```

Tidyquant/quantmod

```
# What's get returned?
head(SPY)
##            SPY.Open SPY.High SPY.Low SPY.Close SPY.Volume SPY.Adjusted
## 2007-01-03   142.25   142.86  140.57    141.37   94807600     111.1660
## 2007-01-04   141.23   142.05  140.61    141.67   69620600     111.4019
## 2007-01-05   141.33   141.40  140.38    140.54   76645300     110.5134
## 2007-01-08   140.82   141.41  140.25    141.19   71655000     111.0245
## 2007-01-09   141.31   141.60  140.40    141.07   75680100     110.9301
## 2007-01-10   140.58   141.57  140.30    141.54   72428000     111.2997
tail(SPY)
##            SPY.Open SPY.High SPY.Low SPY.Close SPY.Volume SPY.Adjusted
## 2018-10-19   277.13   279.30  275.47    276.25  139901600       276.25
## 2018-10-22   277.00   277.36  274.41    275.01   82415800       275.01
## 2018-10-23   270.95   274.87  268.61    273.61  146352700       273.61
## 2018-10-24   273.33   273.76  264.70    265.32  177806700       265.32
## 2018-10-25   267.38   271.81  266.23    270.08  138061500       270.08
## 2018-10-26   265.92   271.00  262.29    265.33  201498400       265.33

symbols <- c("MSFT", "AAPL")
getSymbols(symbols, src = 'yahoo', adjusted = TRUE, from = "2016-01-01")
## [1] "MSFT" "AAPL"
```

xts object

- xts is a wide format. In contrast, ggplot/tidy uses long format.
- We have gather/spread to convert between long/wide format.

- Create xts object:
  - Put index aside, which is usually date
  - Store prices in columns.

```
library(xts)

# if df is a data frame.
# Date | V | GS
xts1 <- xts(x=df[, -1, drop = F], order.by = df[1])

# coredata: returns a matrix from xts objects
core_data <- coredata(xts2)

# index: vector of any Date, POSIXct, chron, yearmon, yearqtr, or DateTime classes
index(xts1)
```

Get data from xts object

```
# What price history is stored here.
str(SPY)
## An 'xts' object on 2007-01-03/2018-10-26 containing:
##   Data: num [1:2977, 1:6] 142 141 141 141 141 ...
##  - attr(*, "dimnames")=List of 2
##   ..$ : NULL
##   ..$ : chr [1:6] "SPY.Open" "SPY.High" "SPY.Low" "SPY.Close" ...
##   Indexed by objects of class: [Date] TZ: UTC
##   xts Attributes:
## List of 2
##  $ src    : chr "yahoo"
##  $ updated: POSIXct[1:1], format: "2018-10-28 23:16:14"
```

```
SPY2003 <- SPY["2003"]
SPY2 <- SPY["2003/2007"]
SPY3 <- SPY["2003-03-01/2007-07-01"]
SPY4 <- SPY["/2007-07-01"] # till
SPY5 <- SPY["2007-07-01/"] # from
SPY6 <- SPY["2007-07-01/", "SPY.High"]
SPY7 <- SPY["2007-07-01/", c("SPY.High", "SPY.Close")]
```

Quandl

```
library(Quandl)
library(tidyverse)

# Sign up with Quandl to get a token
# token_qd <- "xxxx"
Quandl.api_key(token_qd)
## You don't get SPY: SPDR 500 ETF from Quandl from free service.
## rates <- Quandl(c("EOD/SPY"), start_date="2000-01-01", end_date="2013-06-07")
# You get price for Visa
rates <- Quandl(c("EOD/V"), start_date="2000-01-01", end_date="2013-06-07" )
# Get price for GS in xts
```

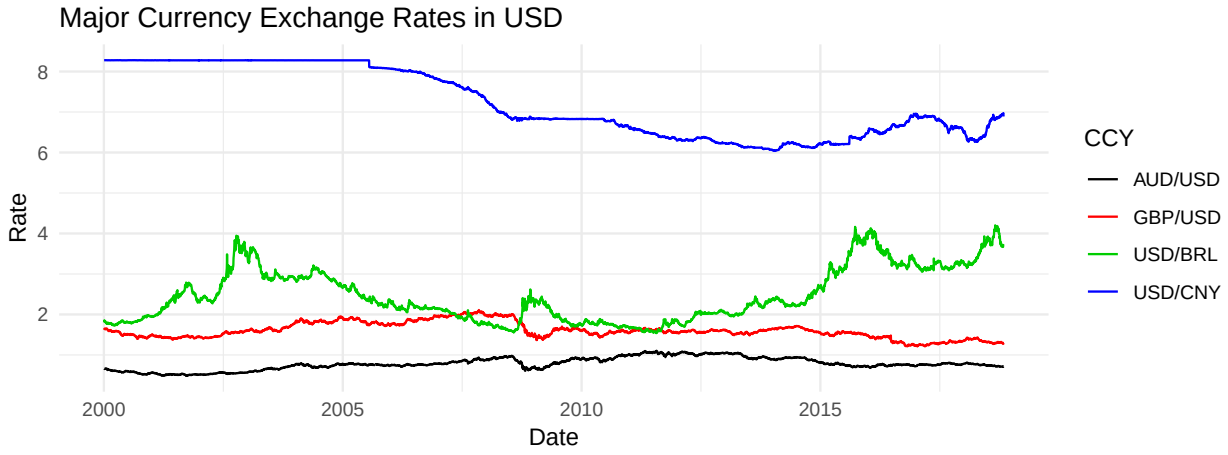
```
rates <- Quandl(c("EOD/GS"), start_date="2000-01-01", end_date="2013-06-07",
                type = "xts")
```

Quandl

```
library(Quandl)                 # Quandl package
library(ggplot2)                # Package for plotting
library(tidyverse)              # Package for reshaping data

Quandl.api_key(token_qd)                # Authenticate your token
# Build vector of currencies
rates <- Quandl(c("FRED/DEXUSAL", "FRED/DEXBZUS", "FRED/DEXUSUK", "FRED/DEXCHUS"),
                start_date="2000-01-01",
                end_date = "2018-11-28")
colnames(rates) <- c("Date", "AUD/USD", "USD/BRL", "GBP/USD", "USD/CNY")
meltdf <- gather(rates, key = "CCY", value = "Rate", -Date)

ggplot(meltdf, aes(x = Date, y = Rate, colour = CCY, group = CCY)) +
  geom_line() +
  scale_colour_manual(values=1:22)+
  ggtitle("Major Currency Exchange Rates in USD") +
  theme_minimal()
```



Quandl and forecast

```
# 52-quandl-forecast.R
# Quandl and Forecast
# Forecast using state space models and automatic ARIMA modelling.

library(Quandl)
library(dplyr)
library(xts)
library(lubridate)
library(forecast)
library(dygraphs)
```

```r
# Start with daily data. Note that "type = raw" will download a data frame.
oil_daily <- Quandl("FRED/DCOILWTICO", type = "raw", collapse = "daily",
                    start_date="2006-01-01", end_date=Sys.Date())
# Now weekely and let's use xts as the type.
oil_weekly <- Quandl("FRED/DCOILWTICO", type = "xts", collapse = "weekly",
                     start_date="2006-01-01", end_date = Sys.Date())
oil_monthly <- Quandl("FRED/DCOILWTICO", type = "xts", collapse = "monthly",
                      start_date="2006-01-01", end_date = "2017-02-28")

# Have a quick look at our three  objects.
str(oil_daily)
str(oil_weekly)
str(oil_monthly)

# Change index from month to day
head(index(oil_monthly))
index(oil_monthly) <- seq(mdy('01/01/2006'), mdy('02/28/2017'), by = 'months')
str(oil_monthly)
head(index(oil_monthly))

dygraph(oil_monthly, main = "Monthly oil Prices")

forebase1 <- oil_weekly["/2016-02-28"]
forecast1 <- forecast(forebase1, h = 4 * 24)

plot(forecast1, main = "Oil Forecast1")

oil_forecast_data1 <- data.frame(
  date = seq(last(index(forebase1)),
             by = 'week', length.out = 4 * 24 + 1)[-1],
  Forecast = forecast1$mean,
  Hi_95 = forecast1$upper[,2],
  Lo_95 = forecast1$lower[,2])

oil_forecast_xts1 <- xts(oil_forecast_data1[,-1],
                         order.by = oil_forecast_data1[,1])
```

```r
forebase2 <- oil_weekly["/2018-11-30"]
forecast2 <- forecast(forebase2, h = 4 * 3)

plot(forecast2, main = "Oil Forecast2")

oil_forecast_data2 <- data.frame(
  date = seq(last(index(forebase2)),
             by = 'week', length.out = 4 * 3 + 1)[-1],
  Forecast2 = forecast2$mean,
  Hi_95_2 = forecast2$upper[,2],
  Lo_95_2 = forecast2$lower[,2])

oil_forecast_xts2 <- xts(oil_forecast_data2[,-1],
                         order.by = oil_forecast_data2[,1])

# Combine the xts objects with cbind.
oil_combined_xts <- merge(oil_weekly, oil_forecast_xts1, oil_forecast_xts2)
```

```
# Add a nicer name for the first column.
colnames(oil_combined_xts)[1] <- "Actual"

dygraph(oil_combined_xts, main = "Oil Prices: Historical and Forecast") %>%
  dySeries("Actual", label = "Actual") %>%
  dySeries(c("Lo_95", "Forecast", "Hi_95")) %>%
  dySeries(c("Lo_95_2", "Forecast2", "Hi_95_2"))
```

dygraph

dygraph for xts https://rstudio.github.io/dygraphs/shiny.html

```
dygraphOutput("dygraph")

dygraph(oil_combined_xts, main = "Oil Prices: Historical and Forecast") %>%
  # Add the actual series
  dySeries("Actual", label = "Actual") %>%
  # Add the three forecasted series
  dySeries(c("Lo_95", "Forecast", "Hi_95"))
```

Quandl/Shiny/dygraph

```
# shiny-51-quandl.R

library(shiny)

library(tidyverse)
library(Quandl)
library(xts)
library(dygraphs)

goldChoice <- "CHRIS/CME_GC1.1" # gold data from CME

dataChoices <- c("WTI oil" = "FRED/DCOILWTICO", #oil data from Fred
                 "Copper" = "ODA/PCOPP_USD", # copper data from ODA
                 "Gold" = "CHRIS/CME_GC1.1",
                 "Silver" = "LBMA/SILVER.1",
                 "Copper" = "CHRIS/CME_HG1.1",
                 "Iron Ore" = "ODA/PIORECR_USD",
                 "Platinum" = "LPPM/PLAT.1",
                 "Palladium" = "LPPM/PALL.1",
                 "Bitcoin" = "BCHARTS/WEXUSD.1")

frequencyChoices <- c("days" = "daily",
                      "weeks" = "weekly",
                      "months" = "monthly")

ui <- fluidPage(
  titlePanel("Commodity"),

  sidebarLayout(
```

```
    sidebarPanel(
      selectInput("dataSet",
                  "Commodity",
                  choices = dataChoices, #Freddie mac
                  selected = "WTI oil"),

      selectInput("frequency",
                  "freq",
                  choices = frequencyChoices,
                  selected = "months"),

      dateRangeInput("dateRange",
                     "Date range",
                     start = "1980-01-01",
                     end   = Sys.Date())
    ),
    mainPanel(
      dygraphOutput("commodity"),
      dygraphOutput("commodity_gold")
    )
  )
)

server <- function(input, output, session) {
  Quandl.api_key("d9EidiiDWoFESfdk5nPy")

  gold <- reactive({
    gold <- Quandl(goldChoice,
                   start_date = format(input$dateRange[1]),
                   end_date = format(input$dateRange[2]),
                   order = "asc",
                   type = "xts",
                   collapse = as.character(input$frequency)
    )
  })

  commodity <- reactive({
    commodity <- Quandl(input$dataSet,
                        start_date = format(input$dateRange[1]),
                        end_date = format(input$dateRange[2]),
                        order = "asc",
                        type = "xts",
                        collapse = as.character(input$frequency)
    )
  })

  output$commodity <- renderDygraph({
    dd <- merge(gold(), commodity())
    dd$ratio <- dd[,1]/dd[,2]
    dd <- dd[, -1, drop = F]
    colnames(dd) <- c(names(dataChoices)[dataChoices == isolate(input$dataSet)],
                      "Gold ratio")
```

```
    dygraph(dd,
            main = paste("Price history of",
                         names(dataChoices[dataChoices==input$dataSet]),
                         sep = " "),
            group = "gold group") %>%
      dyAxis("y", label = "$") %>%
      dySeries("Gold ratio", axis = 'y2') %>%
      dyOptions(axisLineWidth = 1.5, fillGraph = TRUE, drawGrid = TRUE,
                colors = RColorBrewer::brewer.pal(3, "Set1")) %>%
      dyRangeSelector()
  })

  output$commodity_gold <- renderDygraph({
    dygraph(gold(),
            main = paste0("Ratio history of ",
                          names(dataChoices[dataChoices==input$dataSet]),
                                "/Gold"),
            group = "gold group") %>%
      dyAxis("y", label = "$") %>%
      dyOptions(axisLineWidth = 1.5, fillGraph = TRUE, drawGrid = TRUE) %>%
      dyRangeSelector()
  })
}

shinyApp(ui, server)
```

Portfolio analysis

```
# 53-portfolio-2.R

library(purrr)
library(tidyverse)
library(tidyquant)
library(dygraphs)

# SPY: SPDR S&P 500 ETF Trust
# IJS: iShares S&P SmallCap 600 Value Idx
# EFA: iShares MSCI EAFE Index Fund (ETF): large- and mid-capitalization developed market equities, exc
# EEM: iShares MSCI Emerging Markets Indx (ETF)
# AGG: iShares Barclays Aggregate Bond Fund: total U.S. investment-grade bond

symbols <- c("SPY","IJS","EFA","EEM","AGG")

prices <-
  getSymbols(symbols, src = 'yahoo', from = "2005-01-01",
             auto.assign = TRUE, warnings = FALSE) %>%
  purrr::map(~Cl(get(.))) %>%  # Cl is from quantmod: get Close price
  reduce(merge) %>%
  `colnames<-`(symbols)

prices_monthly <- to.monthly(prices, indexAt = "first", OHLC = FALSE)
portfolioComponentReturns <- na.omit(Return.calculate(prices_monthly,
```

```
                                                    method = "log"))

plot_ticker <- function(ticker) {
  ts <- portfolioComponentReturns[, ticker, drop = F]
  sd_lt <- StdDev(ts)

  sd_overtime <- round(rollapply(ts, 20, function(x) StdDev(x)), 4)

  sd_overtime$SD_Longterm <- sd_lt

  dygraph(sd_overtime,
          main = paste("Volatility history of ", ticker)) %>%
    dyAxis("y", label = "%") %>%
    dyOptions(axisLineWidth = 1.5, fillGraph = FALSE, drawGrid = TRUE) %>%
    dyRangeSelector()
}

plot_ticker("SPY")
plot_ticker("IJS")
```

```
w = c(0.25, 0.20, 0.20, 0.25, 0.10)

w_1 <- w[1]
w_2 <- w[2]
w_3 <- w[3]
w_4 <- w[4]
w_5 <- w[5]

asset1 <- portfolioComponentReturns[,1]
asset2 <- portfolioComponentReturns[,2]
asset3 <- portfolioComponentReturns[,3]
asset4 <- portfolioComponentReturns[,4]
asset5 <- portfolioComponentReturns[,5]

portfolio_returns_byhand <-
  (w_1 * asset1) +
  (w_2 * asset2) +
  (w_3 * asset3) +
  (w_4 * asset4) +
  (w_5 * asset5)

names(portfolio_returns_byhand) <- "abs returns"

portfolio_returns_xts_rebalanced_monthly <-
  Return.portfolio(portfolioComponentReturns, weights = w,
                   rebalance_on = "months") %>%
  `colnames<-`("month-rebal returns")

portfolio_returns_xts_rebalanced_yearly <-
  Return.portfolio(portfolioComponentReturns, weights = w,
                   rebalance_on = "years") %>%
  `colnames<-`("year-rebal returns")

head(portfolio_returns_byhand)
```

```r
head(portfolio_returns_xts_rebalanced_monthly)
head(portfolio_returns_xts_rebalanced_yearly)

plot_portfolio <- function(portfolio_returns) {
  portfolio_returns_cum <- cumprod(portfolio_returns + 1)

  library(htmltools)

  g1 <- dygraph(portfolio_returns,
          main = paste("Return")) %>%
    dyAxis("y", label = "%") %>%
    dyOptions(axisLineWidth = 1.5, fillGraph = FALSE, drawGrid = TRUE) %>%
    dyRangeSelector()

  g2 <- dygraph(portfolio_returns_cum,
          main = paste("Cumulative Return")) %>%
    dyAxis("y", label = "%") %>%
    dyOptions(axisLineWidth = 1.5, fillGraph = FALSE, drawGrid = TRUE) %>%
    dyRangeSelector()

  sd_lt <- StdDev(portfolio_returns)

  sd_overtime <-
    round(rollapply(portfolio_returns, 20, function(x) StdDev(x)), 4)

  sd_overtime$SD_Longterm <- sd_lt

  g3 <- dygraph(sd_overtime,
          main = paste("Volatility history of ", "portfolio_returns_cum")) %>%
    dyAxis("y", label = "%") %>%
    dyOptions(axisLineWidth = 1.5, fillGraph = FALSE, drawGrid = TRUE) %>%
    dyRangeSelector()

  browsable(
    tagList(g1, g2, g3)
  )
}

plot_portfolio(portfolio_returns_byhand)
plot_portfolio(portfolio_returns_xts_rebalanced_monthly)
plot_portfolio(portfolio_returns_xts_rebalanced_yearly)

plot_bband <- function(ticker, n_days = 93) {
  ts <- prices[, ticker, drop = F]
  sd_overtime <- round(rollapply(ts, n_days, function(x) StdDev(x)), 3)
  mean_overtime <- round(rollapply(ts, n_days, function(x) mean(x)), 3)
  new_ts <- ts
  new_ts$ma <- mean_overtime
  new_ts$u2 <- mean_overtime + sd_overtime * 2
  new_ts$d2 <- mean_overtime - sd_overtime * 2

  dygraph(new_ts,
          main = paste0("Bollinger Bands ", ticker)) %>%
```

```
    dyAxis("y", label = "") %>%
    dySeries("ma", strokePattern = "dashed") %>%
    dyOptions(axisLineWidth = 1.5, fillGraph = FALSE, drawGrid = TRUE) %>%
    dyRangeSelector()
}

plot_bband("SPY")
```