FE8828 Programming Web Applications in Finance

Session 5 Building Financial Applications

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

Sep 26, 2019

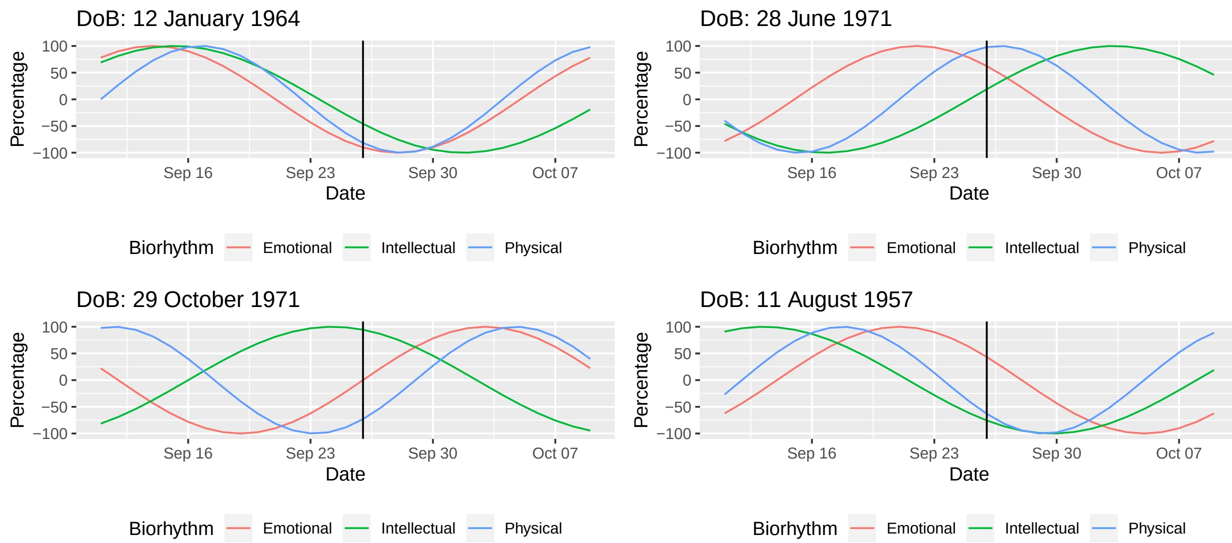
Lecture 10: Building Financial Applications

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 mood 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/Stocks 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
```

```r
getSymbols('SPY', src = 'yahoo', adjusted = TRUE, output.size = 'full')
## [1] "SPY"
str(SPY)
## An 'xts' object on 2007-01-03/2019-09-25 containing:
##   Data: num [1:3205, 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: "2019-09-26 03:05:14"

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

Tidyquant/quantmod

```r
# 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     109.0000
## 2007-01-04   141.23   142.05  140.61    141.67   69620600     109.2313
## 2007-01-05   141.33   141.40  140.38    140.54   76645300     108.3600
## 2007-01-08   140.82   141.41  140.25    141.19   71655000     108.8612
## 2007-01-09   141.31   141.60  140.40    141.07   75680100     108.7686
## 2007-01-10   140.58   141.57  140.30    141.54   72428000     109.1311
tail(SPY)
##            SPY.Open SPY.High SPY.Low SPY.Close SPY.Volume SPY.Adjusted
## 2019-09-18   300.49   301.22  298.24    301.10   73375800     299.7159
## 2019-09-19   301.53   302.63  300.71    301.08   76560500     299.6960
## 2019-09-20   300.36   300.67  297.41    298.28   89565000     298.2800
## 2019-09-23   297.55   299.00  297.27    298.21   43476800     298.2100
## 2019-09-24   299.41   299.84  294.81    295.87   94869400     295.8700
## 2019-09-25   295.96   297.55  294.33    297.46   42617733     297.4600

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.

```r
library(xts)
```

```r
# 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

```r
# What price history is stored here.
str(SPY)
## An 'xts' object on 2007-01-03/2019-09-25 containing:
##   Data: num [1:3205, 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: "2019-09-26 03:05:14"
```

```r
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")]
```

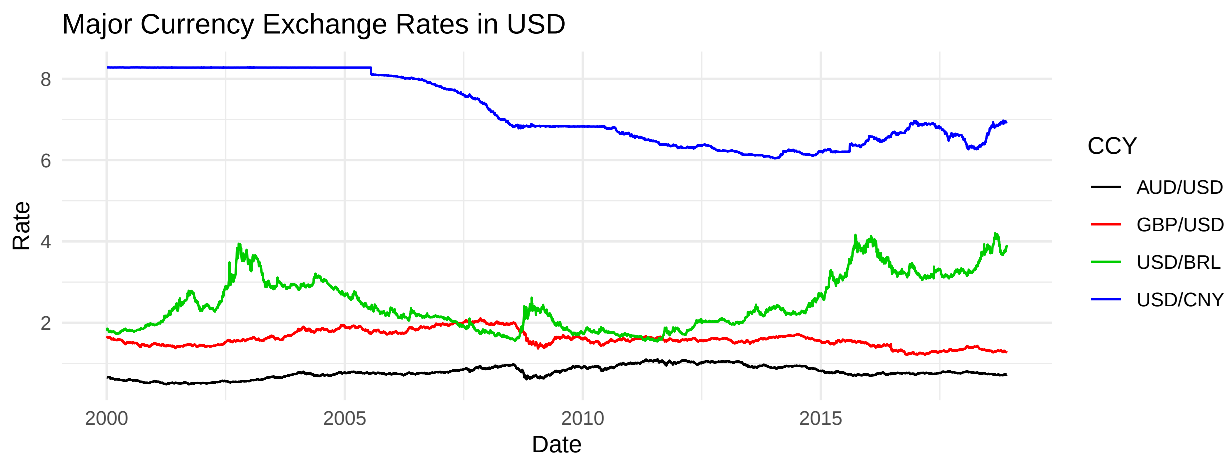
Quandl

```r
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 don't get EOD US Stocks for free from Quandl from 2019
## rates <- Quandl(c("EOD/V"), start_date="2000-01-01", end_date="2013-06-07" )
```

Quandl

```r
library(Quandl)                  # Quandl package
library(ggplot2)                 # Package for plotting
```

```r
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

```r
# 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)

# 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 = Sys.Date())
```

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

cat(paste0("daily: ", paste0(range(oil_daily$Date), collapse = ", "), "\n"))
cat(paste0("weekly: ", paste0(range(index(oil_weekly)), collapse = ", "), "\n"))
cat(paste0("monthly: ", paste0(range(index(oil_monthly)), collapse = ", "), "\n"))

# Change index from month to day
head(index(oil_monthly))
index(oil_monthly) <- seq(mdy('01/01/2006'), Sys.Date(), by = 'months')[1:length(oil_monthly)]
# index(oil_monthly) <- seq(mdy('01/01/2006'), (Sys.Date() - 365 * 2), by = 'months')
str(oil_monthly)
head(index(oil_monthly))

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

forebase1 <- oil_weekly[paste0("/", Sys.Date() - 365 * 2)]
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])

forebase2 <- oil_weekly[paste0("/", Sys.Date() - 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

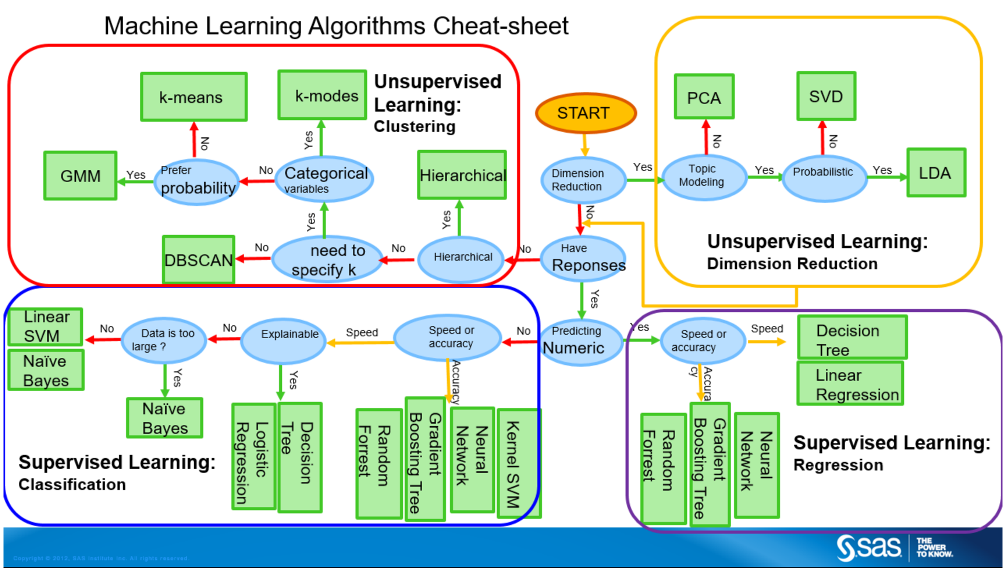
```
# shiny-51-quandl.R

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",
```

```r
                  "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') %>%
```

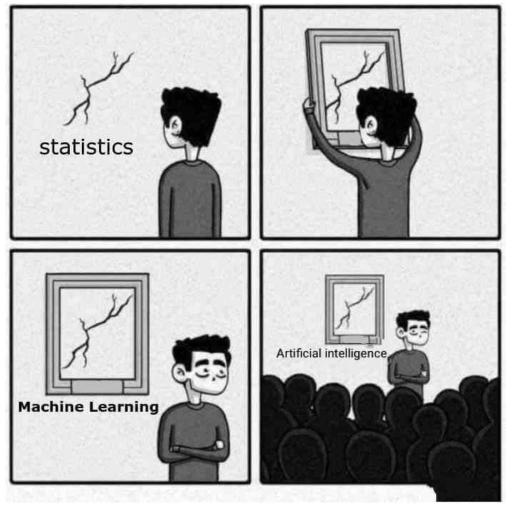
```
      dyOptions(axisLineWidth = 1.5, fillGraph = TRUE, drawGrid = TRUE,
                colors = RColorBrewer::brewer.pal(3, "Set1")) %>%
      dyRangeSelector()
  })

  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)
```

Lecture 11: Building Predictive Model

Machine Learning



Statistics and Machine Learning



Statistics and Machine Learning

- Statistics is a age-old year subject, with many developed theory.
- Machine learning is an algorithm that can learn from data without relying on rules-based.
- ML uses many statistical theories in application.
- ML emphasizes optimization and performance (accuracy) over inference (conclusion based on reasons and evidence) which is what statistics is concerned about.

  | ML professional: "The model is 85% accurate in predicting Y, given a, b and c." | Statistician: "The model is 85% accurate in predicting Y, given a, b and c; and I am 90% certain that you will obtain the same result."

Supervised v.s. Unsupervised

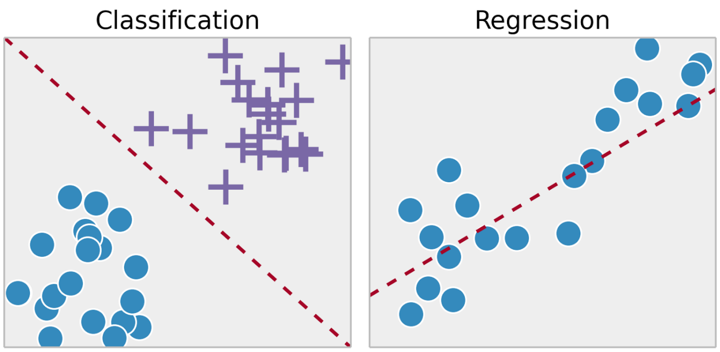
- Supervised learning: It is based on example input-output pairs. It infers a function from labeled training data consisting of a set of training examples
- Unsupervised learning: It is a type of self-organized learning that helps find previously unknown patterns in data set without pre-existing labels. It is also known as self-organization and allows modeling probability densities of given inputs.
- Reinforcement learning: ...

Machine Learning

- Regression
- Classification
- Ensemble
- Neutral network
  - Deep learning: multiple hidden layers.

Regression and Classification

Regression and classification are two main categories of machine learning algorithms under supervised learning.

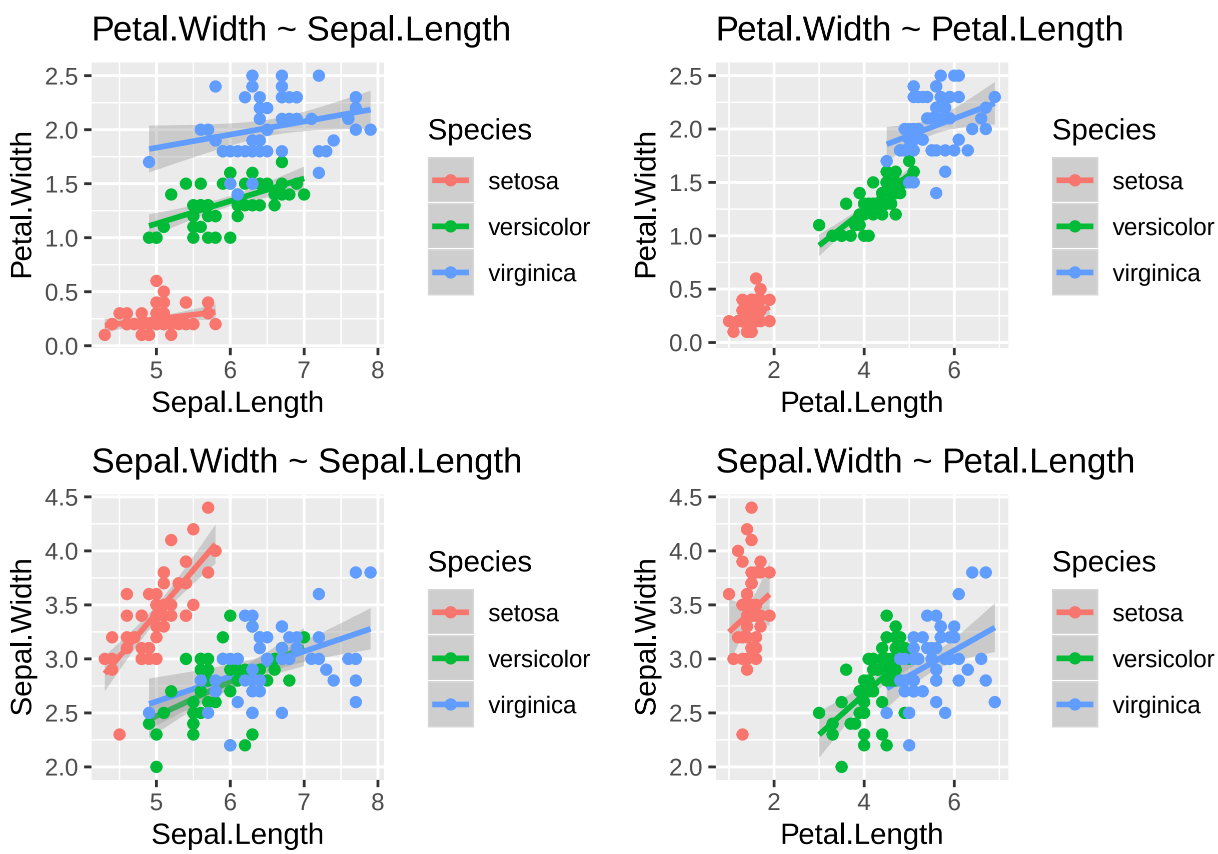


Regression

We can use many linear regression methods.

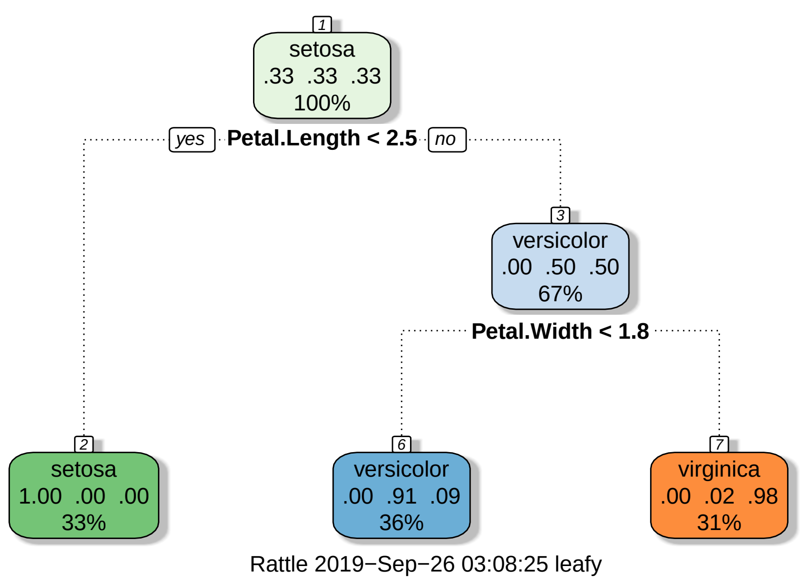
```
p1 <- ggplot(iris, aes(x = Sepal.Length, y = Petal.Width, color = Species)) +
  geom_smooth(method = "lm") +
  geom_point() +
  labs(title = "Petal.Width ~ Sepal.Length")

p2 <- ggplot(iris, aes(x = Petal.Length, y = Petal.Width, color = Species)) +
  geom_smooth(method = "lm") +
  geom_point() +
  labs(title = "Petal.Width ~ Petal.Length")

p3 <- ggplot(iris, aes(x = Sepal.Length, y = Sepal.Width, color = Species)) +
  geom_smooth(method = "lm") +
  geom_point() +
  labs(title = "Sepal.Width ~ Sepal.Length")

p4 <- ggplot(iris, aes(x = Petal.Length, y = Sepal.Width, color = Species)) +
  geom_smooth(method = "lm") +
  geom_point() +
  labs(title = "Sepal.Width ~ Petal.Length")
```

Regression



Classification

Decision Tree



```
predictions <- predict(iris_rp, iris, type = "class")
which(iris$Species != predictions)
## [1]  71 107 120 130 134 135
# caret::confusionMatrix(predictions, iris$Species)
```

Confusion Matrix

- Shows how model performs

```
Actual Target |          0         |  1  |
Model Output  |                    |     |
          0   |         90         | 10  |
          1   |         10         | 90  |
```

Machine Learning workflow

1. Setting
2. Exploratory Data Analysis
3. Feature Engineering
4. Data Preparation
5. Modelling
6. Conclusion

Machine Learning

- Data preparation:

Split into different groups. For simple data, we may split using 75/25 or 80/20. For complex data, we need to ensure selected 75 has coverage for different kinds, to avoid bias. For example, we are to predict a infrequent event (occuring 20%), shall our selected sample contain 20% or 50% or 75%?

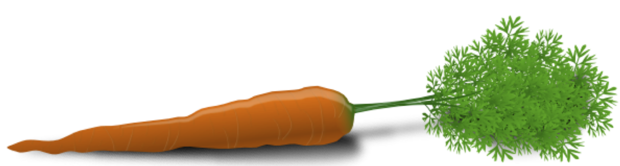
- Modeling:

Choose a few models and tune the hyperparametes. Tuning of hyperparameters is model-specific. You shall learn it in-depth with each model.

Measure the performacne and optimize it. Confusion matrix, AUC/ROC, model-specific output…

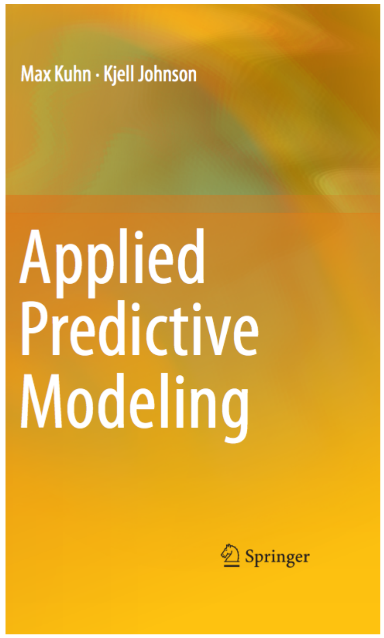
`Caret` Package

Caret is short for _C_lassification _A_nd _RE_gression _T_raining.



Main author is Max Kuhn. He has a book "Applied Predictive Modeling", By Max Kuhn and Kjell Johnson.

Max is now in RStudio, working on *parsnip* in tidymodel.



Caret Package

Links to 200 over (238 as of this version) https://topepo.github.io/caret/available-models.html

```
## Loading required package: lattice
##
## Attaching package: 'caret'
## The following object is masked from 'package:purrr':
##
##     lift
## [1] "ada,  AdaBag,  AdaBoost.M1,  adaboost,  amdai,  ANFIS,  avNNet,  awnb,  awtan,  bag,  bagEarth,
```

Project Iris

Use the petal/sepal width/length to determine which species it is.

```r
library(caret)

set.seed(123)
trainIndex <- createDataPartition(iris$Species, p = .8,
 list = FALSE,
 times = 1)

train <- iris[ trainIndex,]
test  <- iris[-trainIndex,]

train_x <- select(train, -Species)
train_y <- train$Species

test_x <- select(test, -Species)
test_y <- test$Species

# Cross validation
```

```
fitControl <- trainControl(
 method = "repeatedcv",
 number = 10,
 repeats = 5)

# first run may need to do package installation. Caret
# install.packages("e1071")
# recursive partition = decision tree
dt_fit <- train(Species ~ ., data = train,
 method = "rpart",
 trControl = fitControl,
 preProcess=c("center", "scale"))

dt_fit
plot(dt_fit)

predictions <- predict(dt_fit, test)
confusionMatrix(predictions, test$Species)

# random Forests
rf_fit <- train(Species ~ .,
                data = train,
                method = "ranger")

rf_fit
plot(rf_fit)

predictions <- predict(rf_fit, test)
confusionMatrix(predictions, test$Species)
which(y_test != predictions)
y_test[which(y_test != predictions)]
```

Project Bank

We previous whether our telemarketing is successful.

The output is binary - classification for binary output is very common.

For model simplicity, we use logistic regression. For logistic regression, we need to use one-hot encoding.

One-hot Encoding is implemented with dummy variables in R.

If one column contains more-than-one value, e.g. "admin", "engineer", "manager", we replace it with 2 = 3 - 1 columns

```
| job      |        | job_admin | job_engineer |
| admin    |   ==>  |     1     |       0      |
| engineer |        |     0     |       1      |
| manager  |        |     0     |       0      |
```

Project Bank - Load

```
bank <- read.csv("https://goo.gl/PBQnBt", sep = ";")
# We only work on following fields.
```

```r
bank_fit <- bank %>% select(y,
                loan,
                default,
                housing,
                poutcome,
                job,
                marital) %>%
  mutate_if(is.factor, as.character) %>%
  mutate(y = ifelse(y == "yes", "y", "n"))

str(bank_fit)
```

Project Bank - Dummy Variables

```r
# create dummy variables
dummies <- dummyVars("y ~ loan + default + housing + poutcome + job + marital",
                     data = bank_fit, fullRank = TRUE)
# generate data frame of dummy variables
bank_new <- data.frame(predict(dummies, newdata = bank_fit))
# add back y variable to data
bank_new <- bind_cols(bank_fit["y"], bank_new) %>% mutate(y = factor(y))

summary(bank_new)
```

Project Bank - Train/Test Data

```r
# library(caret)
set.seed(1234)
trainIndex <- createDataPartition(bank_new$y, p = .8,
 list = FALSE,
 times = 1)

bank_train <- bank_new[ trainIndex,]
bank_test  <- bank_new[-trainIndex,]

featurePlot(x = bank_new[-1],
            y = bank_new$y,
            plot = "box",
            strip=strip.custom(par.strip.text=list(cex=.7)),
            scales = list(x = list(relation="free"),
                          y = list(relation="free")))
```

Project Bank - Feature Plot

```r
featurePlot(x = bank_new[-1],
            y = bank_new$y,
            plot = "density",
            strip=strip.custom(par.strip.text=list(cex=.7)),
            scales = list(x = list(relation="free"),
                          y = list(relation="free")))
```

Project Bank - Train

```r
train_control <- trainControl(
    method = 'repeatedcv',                    # k-fold cross validation
    number = 5,                       # number of folds
    savePredictions = 'final',        # saves predictions for optimal tuning parameter
    classProbs = TRUE,                   # should class probabilities be returned
)

if (FALSE) {
  # Running time is too long. Skip running.
  adaboost_fit <- train(y ~ .,
                    data = bank_train,
                   method='adaboost',
                   tuneLength=2,
                   trControl = train_control)
  adaboost_fit

  predictions <- predict(adaboost_fit, newdata = bank_train)
  confusionMatrix(predictions, bank_train$y)

  predictions <- predict(adaboost_fit, bank_test)
  confusionMatrix(predictions, bank_test$y)
}

# Logistic regression
log_fit <- train(y ~ .,
                 data = bank_train,
                 method = "glm",
                 family = binomial,
                 trControl = train_control)

predictions <- predict(log_fit, newdata = bank_train)
confusionMatrix(predictions, bank_train$y)

predictions <- predict(log_fit, bank_test)
confusionMatrix(predictions, bank_test$y)
# which(test$y != predictions)
```

Project Bank - Train with Decision Tree

```r
# Recursive Partitioning and Regression Trees
rpart_fit <- train(y ~ .,
                   data = bank_train,
                   method="rpart",
                   trControl = train_control)
predictions <- predict(rpart_fit, bank_train)
confusionMatrix(predictions, bank_train$y)

predictions <- predict(rpart_fit, bank_test)
```

```r
confusionMatrix(predictions, bank_test$y)
```

```r
rattle::ggVarImp(rpart_fit$finalModel, log=TRUE)
rattle::fancyRpartPlot(rpart_fit$finalModel)
```

Project Bank - Model comparison

```r
models_compare <- resamples(list(RP = rpart_fit, GLM = log_fit))

# Summary of the models performances
summary(models_compare)
```

Project Australia Weather

- A complete project with some feature engineering.

https://www.dropbox.com/s/p73mdxcrx05mbwb/aus_weather_predict.Rmd?dl=1