Tarea 1 - Algoritmos de clasificación

Jordi Vanrell Forteza

26/5/2021

La tarea requiere escoger tres archivos de cada uno de los tres grupos provistos. Los grupos se corresponden con acciones de capitalización pequeña (nombrados *Stock20x.txt*), media (*Stock10x.txt*) y grande (*Stockx.txt*). Se descargan los ficheros y se meten en un mismo directorio. Se seleccionan aleatoriamente tres números entre 1 y 5 que marcarán la elección de las acciones en los tres grupos. Se leen con un bucle y se almacenan los nueve en una lista.

# Se escogen 3 acciones de cada grupo (al azar)  
set.seed(603) # semilla para completa reproducibilidad  
idx <- as.character(sample(1:5, size = 3, replace = F))  
idx <- c(paste0("", idx), paste0("10", idx), paste0("20", idx))  
  
Stock <- vector(mode = "list", length = length(idx))  
for (i in idx){ # Bucle de lectura de los archivos; se almacenan en lista  
 Stock[[which(idx==i)]] <- read.table(paste0("Stock", i, ".txt"), header = T)  
 rm(i)  
}

Dicha lista consiste en 9 tablas de datos ordenadas según el índice de posición del vector idx: 5, 3, 1, 105, 103, 101, 205, 203 y 201.

# (a) Programario para la aplicación de los algoritmos de clasificación de transacciones.

require(tidyverse) # Paquete para la manipulación de de data frames

Se procede a clasificar las transacciones según los algoritmos *Tick Rule*, *Quote Rule* y *Lee & Ready*. Para ello se arma un bucle que los implementará en cada una de las transacciones para cada activo. Debido a la propia idiosincrasia de los algoritmos resulta más adecuado aplicarlos a través de bucles sobre objetos de tipo vector que sobre datos estructurados en *data frames*. Por ello se construyen tres bucles internos, uno por algoritmo, dentro de los cuales se codifican las reglas de clasificación mediante funciones condicionales. En cada caso se transforman los vectores en columnas de los *data frames* originales.

for (d in 1:length(Stock)){  
 Stock[[d]] <- Stock[[d]] %>%  
 arrange(day, time) %>%   
 # se asegura que las órdenes están correctamente ordenadas  
 mutate(Q = round((ask+bid)/2, 4))   
 # se obtiene el valor como la media de ask y bid  
   
 day <- Stock[[d]]$day # se vectoriza day  
 price <- Stock[[d]]$price # se vectoriza price  
   
 # Tick Rule  
 TR = c(); TR[1] = NA # se inicializa el vector con valor NA  
 for (i in 2:length(price)){ # implementación del algoritmo  
 TR[i] <- ifelse(day[i] == day[i-1] & price[i] > price[i-1], 1,  
 ifelse(day[i] == day[i-1] & price[i] < price[i-1], -1,  
 ifelse(day[i] == day[i-1] & price[i] == price[i-1], TR[i-1], NA)))  
 }  
 Stock[[d]]$TR <- TR; rm(TR) # se agrega el vector como columna  
   
 # Quote Rule  
 Q <- Stock[[d]]$Q # se vectoriza el valor  
 QR = c() # se inicializa un vector vacío  
 for (i in 1:length(price)){ # se implementa el algoritmo  
 QR[i] <- ifelse(price[i] > Q[i], 1,  
 ifelse(price[i] < Q[i], -1, NA))  
 }  
 Stock[[d]]$QR <- QR; rm(QR) # se agrega el vector como columna  
   
 # Lee & Ready  
 LR = c(); LR[1] = ifelse(price[1] > Q[1], 1, # se inicializa el vector  
 ifelse(price[1] < Q[1], -1, NA))  
 for (i in 2:length(price)){  
 LR[i] <- ifelse(price[i] > Q[i], 1,  
 ifelse(price[i] < Q[i], -1,   
 ifelse(price[i] == Q[i] & day[i] == day[i-1],   
 LR[i-1], NA)))  
 rm(i)  
 }  
 Stock[[d]]$LR <- LR; rm(LR, day, price, Q, d)  
 # se agrega el vector como columna al dataset  
}

# (b) Precisión diaria por activo y algoritmo

require(scales)

En primera instancia se requieren los datos de precisión de los algoritmos para cada activo y día del rango de los datos. Se define la precisión como la ratio de transacciones correctamente clasificadas entre el total de transacciones clasificadas (no se toman en cuenta las que el algoritmo no clasifica, en cada caso). Para obtener esta información es necesario filtrar por día en cada una de las tablas de la lista; así se obtienen tres datos por tabla y día, uno por algoritmo. Estos tres se agregan y se almacenan en un *data frame*. El proceso se repite con un bucle hasta que el *data frame* contiene los datos para todos los días. La tabla completa, con filas y columnas convenientemente nombradas, se almacena en una lista y todo el proceso anterior se repite en bucle hasta que se ha efectuado para todos los activos. El resultado es una lista de nueve *data frames* con los datos de precisión requeridos, uno por cada activo.

# se inicializa la lista de almacenamiento:  
tabla\_i\_list <- vector(mode = "list", length = length(Stock))  
for (s in 1:length(Stock)){  
 subtabla\_i <- data.frame() # se inicializa subtabla  
 for (d in unique(Stock[[s]]$day)){  
 stock\_parse <- Stock[[s]] %>% filter(day == d) # filtro de día  
 acc\_f = function(R){   
 # se define función de precisión (R es parámetro para algoritmo)  
 mean(stock\_parse[, R]==stock\_parse[, "buysell"], na.rm = T)  
 } # Aplicación de las funciones  
 acc\_TR = acc\_f("TR"); acc\_QR = acc\_f("QR"); acc\_LR = acc\_f("LR")  
 acc\_df = data.frame(acc\_TR, acc\_QR, acc\_LR) # se introducen en tabla  
 subtabla\_i <- rbind(subtabla\_i, acc\_df)   
 # se agrega lo anterior a subtabla\_i  
 rm(d, acc\_TR, acc\_QR, acc\_LR, acc\_df, acc\_f, stock\_parse)  
 }  
 tabla\_i\_list[[s]] <- subtabla\_i # se agrega subtabla\_i a tabla\_i\_list  
 var <- colnames(tabla\_i\_list[[s]]) # se extraen nombres de columnas  
 rownames(tabla\_i\_list[[s]]) <- paste0("day ", unique(Stock[[s]]$day))   
 # se renombran las filas  
 colnames(tabla\_i\_list[[s]]) <- paste0("Stock", idx[s], ", ", var)   
 # se añade nombre del stock a las columnas  
 rm(subtabla\_i, s, var)  
}

A modo de ejemplo, en la Tabla I se disponen los resultados para el activo Stock5. Las filas contienen la dimensión día y en las columnas los algoritmos. En caso de querer explorar los resultados de los demás activos puede hacerse mediante la instrucción tabla\_i\_list[[x]], sustituyendo la x por el índice del activo deseado. En estas los resultados están en tanto por uno.

tabla\_i <- tabla\_i\_list[[1]]  
var <- colnames(tabla\_i) # Se disponen los datos como % con 2 dec.  
tabla\_i[, var] <- lapply(tabla\_i[, var], percent\_format(accuracy = .01))  
rm(var)

# Por cuestiones de compilación la tabla (esta y las siguientes) se ha generado de forma separada, de acuerdo con el código que sigue. Este es un código que NO puede compilarse en Rmd.  
library(officer)  
library(flextable)  
library(magrittr)  
# Para exportar tablas a formato Word  
ft <- flextable(data = tabla\_i %>% add\_rownames()) %>%   
 theme\_zebra %>% autofit  
ft <- set\_caption(ft, caption = "Tabla I", style = "Table Caption")  
# Crea un archivo temp  
tmp <- tempfile(fileext = ".docx")  
# Crea un documento docx  
read\_docx() %>% body\_add\_flextable(ft) %>% print(target = tmp)  
browseURL(tmp) # abre el documento

**Tabla I**

|  | **Stock5, acc\_TR** | **Stock5, acc\_QR** | **Stock5, acc\_LR** |
| --- | --- | --- | --- |
| day 1 | 80.14% | 98.14% | 93.04% |
| day 2 | 77.91% | 93.80% | 85.53% |
| day 3 | 78.60% | 93.83% | 84.16% |
| day 6 | 78.59% | 78.81% | 71.59% |
| day 7 | 79.04% | 88.43% | 78.68% |
| day 8 | 82.39% | 94.96% | 85.94% |
| day 9 | 82.86% | 96.87% | 87.89% |
| day 10 | 82.94% | 92.96% | 85.77% |
| day 13 | 77.94% | 87.60% | 73.89% |
| day 14 | 83.01% | 97.10% | 90.04% |
| day 15 | 78.95% | 95.77% | 86.44% |
| day 16 | 77.57% | 95.16% | 84.82% |
| day 17 | 80.73% | 95.10% | 85.87% |
| day 20 | 81.51% | 84.90% | 79.62% |
| day 21 | 79.00% | 95.34% | 85.99% |
| day 22 | 80.30% | 95.88% | 89.60% |
| day 23 | 76.00% | 94.99% | 84.24% |
| day 24 | 81.48% | 94.74% | 84.50% |
| day 27 | 86.08% | 96.46% | 87.87% |
| day 28 | 84.63% | 97.00% | 87.60% |
| day 29 | 78.21% | 92.05% | 77.96% |
| day 30 | 80.12% | 96.43% | 87.11% |

En segundo lugar, se requiere la media y la desviación típica de las precisiones diarias por activo y algoritmo. Para ello se usa como fuente de partida la lista de tablas a partir de la cual se ha generado la Tabla I. Para cada una de ellas, y tras definir funciones para la media y la desviación típica, se calculan ambos indicadores para los tres algoritmos. Los resultados se disponen en la fila de un *data frame* donde van agregándose en bucle los de los nueve activos.

tabla\_ii <- data.frame() # Se inicializa tabla\_ii como tabla vacía  
for (s in 1:length(tabla\_i\_list)){  
 var <- colnames(tabla\_i\_list[[s]])   
 # Se toman los nombres de las columnas de la subtabla  
 mean\_f <- function(pos){ # Se define una función general para la media  
 mean(tabla\_i\_list[[s]][, var[pos]], na.rm = T)  
 }  
 sd\_f <- function(pos){ # Se define una función para la desv típica  
 sd(tabla\_i\_list[[s]][, var[pos]], na.rm = T)  
 }  
 mean\_acc\_TR <- mean\_f(1); sd\_acc\_TR <- sd\_f(1) # Aplicación de TR  
 mean\_acc\_QR <- mean\_f(2); sd\_acc\_QR <- sd\_f(2) # Aplicación de QR  
 mean\_acc\_LR <- mean\_f(3); sd\_acc\_LR <- sd\_f(3) # Aplicación de LR  
 subtabla\_ii <- data.frame(mean\_acc\_TR, sd\_acc\_TR, mean\_acc\_QR,   
 sd\_acc\_QR, mean\_acc\_LR, sd\_acc\_LR)  
 tabla\_ii <- rbind(tabla\_ii, subtabla\_ii)   
 # Introducción en data frame y agregación  
 rm(var, subtabla\_ii, s, mean\_f, sd\_f, mean\_acc\_TR, sd\_acc\_TR, mean\_acc\_QR, sd\_acc\_QR, mean\_acc\_LR, sd\_acc\_LR)  
}  
rownames(tabla\_ii) <- paste0("Stock", idx) # Se renombran las filas  
var <- colnames(tabla\_ii) # Se disponen los datos como % con 2 dec  
tabla\_ii[, var] <- lapply(tabla\_ii[, var], percent\_format(accuracy = .01))  
rm(var)

En la Tabla II se presentan los resultados. Cada fila representa un activo y en las columnas se representan la media y la desviación típica en porcentaje.

# Para exportar tablas a formato Word  
ft <- flextable(data = tabla\_ii %>% add\_rownames()) %>%   
 theme\_zebra %>% autofit  
ft <- set\_caption(ft, caption = "Tabla II", style = "Table Caption")  
# Crea un archivo temp  
tmp <- tempfile(fileext = ".docx")  
# Crea un documento docx  
read\_docx() %>% body\_add\_flextable(ft) %>% print(target = tmp)  
browseURL(tmp) # abre el documento

**Tabla II**

|  | **TR** | | **QR** | | **LR** | |
| --- | --- | --- | --- | --- | --- | --- |
|  | **mean\_acc** | **sd\_acc** | **mean\_acc** | **sd\_acc** | **mean\_acc** | **sd\_acc** |
| Stock5 | 80.36% | 2.52% | 93.47% | 4.65% | 84.46% | 5.17% |
| Stock3 | 77.78% | 2.17% | 80.64% | 8.00% | 76.57% | 7.62% |
| Stock1 | 83.64% | 2.69% | 95.60% | 1.85% | 86.53% | 2.81% |
| Stock105 | 75.98% | 4.22% | 84.82% | 5.15% | 80.77% | 4.95% |
| Stock103 | 76.33% | 3.40% | 76.79% | 8.93% | 73.53% | 8.74% |
| Stock101 | 87.89% | 3.01% | 91.55% | 4.93% | 85.25% | 4.95% |
| Stock205 | 77.75% | 3.66% | 76.96% | 8.66% | 74.41% | 8.31% |
| Stock203 | 80.33% | 5.63% | 89.79% | 5.54% | 86.52% | 6.44% |
| Stock201 | 75.30% | 8.48% | 81.85% | 7.02% | 79.72% | 7.25% |

# (c) Precisión sobre el volumen por activo y algoritmo para las diferentes franjas temporales

require(lubridate)

A continuación, se requieren los datos de precisión de los algoritmos para cada activo y franja temporal de media hora medidos sobre el volumen (y no sobre el número de transacciones). En este caso la precisión se mide como la suma de los mínimos de los volúmenes reales y estimados de compra y de venta ponderados sobre el volumen total intercambiado. Esta aproximación sobreestima la precisión real de los algoritmos en la medida en que los errores de identificación en las compras se compensan con los errores en las ventas, y viceversa. Sin embargo, vienen a ilustrar la capacidad de estos para predecir la proporción de compras y ventas en una franja temporal determinada. Del mismo modo que en (b), los resultados se almacenan en una lista de *data frames*, una tabla por activo.

Una vez más los resultados se obtienen a través de un triple bucle anidado. Dado un activo, para asignar cada operación a su franja temporal correspondiente se usan la variable time, los datos de fecha y la referencia de 34200 segundos correspondiente con la apertura del mercado a las 9:30. El formato fecha del paquete lubridate permite funciones de redondeo de fecha y hora a los 30 minutos. De esta información puede extraerse y agregarse la información sobre horas y minutos. El proceso únicamente requiere una corrección adicional: en ocasiones hay transacciones registradas a las 9:30:00 y 0 -ésimas (y que se redondearían a las 9:30), por lo que, por practicidad, se asignan al intervalo de las 10H 0M.

Los bucles interiores se hacen depender de los valores vectorizados de los algoritmos y las franjas temporales de media hora. Para cada combinación de los valores de dichos vectores se agregan los volúmenes en una matriz de confusión interna a partir de la cual se calcula la precisión tal y como se ha definido en este apartado. Los resultados se agregan primero por franja horaria para un algoritmo dado; a continuación, se yuxtaponen las columnas correspondientes a los diferentes algoritmos para un mismo activo y por último se agregan las tablas, con filas y columnas renombradas, a la lista principal.

alg <- c("TR", "QR", "LR")  
for (d in 1:length(Stock)){  
 for (a in alg){ # se reemplazan todos los NA por 0  
 Stock[[d]][,a][is.na(Stock[[d]][,a])] <- 0  
 }  
}

tabla\_iii\_list <- vector(mode = "list", length = length(Stock))  
for (d in 1:length(Stock)){  
 this\_1 <- Stock[[d]] %>%  
 mutate(time\_stamp = ymd(paste0("2011-06-", day)) + hms("09:30:00") + time-34200,  
 # 34200 son los segundos pasados la medianoche;   
 # time\_stamp está redondeado a la baja en las -ésimas de segundo.  
 ceil\_t = ceiling\_date(time\_stamp, unit = "30 minutes"),   
 # se redondea al alza en intervalos de 30 minutos.  
 ceil\_halfh = hm(paste0(hour(ceil\_t),":",minute(ceil\_t)))) %>%   
 # se extren las horas y minutos  
 select(-ceil\_t)  
 # Puede que haya operaciones a las 9:30:00 y 0 -ésimas de s (no antes)   
 # así que, por practicidad, se asignan a las 10:00:00:  
 this\_1$ceil\_halfh[this\_1$ceil\_halfh==hms("9H 30M 0S")] <- hms("10H 0M 0S")  
 # Se construye un vector de texto con las franjas de tiempo:  
 time <- unique(as.character(this\_1$ceil\_halfh))  
 alg <- c("TR", "QR", "LR") # Se hace lo mismo con los algoritmos  
 acc\_at <- data.frame(aux = rep(0, length(time)))   
 # se inicializa el data.frame con una columna auxiliar  
 for (a in alg){  
 acc\_t = data.frame() # se inicializa un data frame  
 group\_cols <- c("ceil\_halfh", "buysell", a)   
 # criterio de agrupación dependiente de a  
 for (t in time){ # bucle para la agregación de vol según franja de t  
 this\_2 <- this\_1 %>%  
 group\_by\_at(group\_cols) %>%  
 summarise(vol = sum(vol)) %>%  
 filter(ceil\_halfh == t)  
 this\_2 <- this\_2[this\_2[,a]!=0,]  
 # Precisión de un algoritmo dado para una franja de tiempo dada  
 acc = (min(sum(this\_2$vol[this\_2[, "buysell"]==1]),   
 sum(this\_2$vol[this\_2[, a]==1])) +  
 min(sum(this\_2$vol[this\_2[, "buysell"]==-1]),   
 sum(this\_2$vol[this\_2[, a]==-1])))/  
 (sum(this\_2$vol[this\_2[, "buysell"]==1]) +   
 sum(this\_2$vol[this\_2[, "buysell"]==-1]))  
 acc = data.frame(acc) # conversión de data.frame  
 acc\_t = rbind(acc\_t, acc)   
 # agregación de resultados de t para un algoritmo dado  
 }  
 acc\_at <- cbind(acc\_at, acc\_t) # agreg de algoritmos para un activo  
 }  
 acc\_at <- acc\_at[,2:4] # se desecha la columna auxiliar  
 colnames(acc\_at) <- paste0("Stock", idx[d], ", ", alg)   
 # se renombran las columnas con el nombre del activo  
 rownames(acc\_at) <- time # se renombran las filas con las franjas de t  
 tabla\_iii\_list[[d]] <- acc\_at # se agrega subtabla a la lista de tablas  
 rm(a, acc, acc\_at, acc\_t, alg, d, group\_cols, t, this\_1, this\_2, time)  
}

En la Tabla III se disponen los resultados medios del total de las franjas de 30 minutos. Cada fila es un activo y en las columnas se disponen las precisiones medias de cada algoritmo.

tabla\_iii <- data.frame() # Se inicializa tabla\_ii como tabla vacía  
for (s in 1:length(tabla\_iii\_list)){  
 var <- colnames(tabla\_iii\_list[[s]]) # Se toman los nombres de las col  
 mean\_f <- function(pos){ # Se define una función general para la media  
 mean(tabla\_iii\_list[[s]][, var[pos]], na.rm = T)  
 }  
 mean\_acc\_TR <- mean\_f(1) # Aplicación de TR  
 mean\_acc\_QR <- mean\_f(2) # Aplicación de QR  
 mean\_acc\_LR <- mean\_f(3) # Aplicación de LR  
 subtabla\_iii <- data.frame(mean\_acc\_TR, mean\_acc\_QR, mean\_acc\_LR)  
 tabla\_iii <- rbind(tabla\_iii, subtabla\_iii) # Agregación en data frame  
 rm(var, subtabla\_iii, s, mean\_f, mean\_acc\_TR, mean\_acc\_QR, mean\_acc\_LR)  
}  
rownames(tabla\_iii) <- paste0("Stock", idx) # Se renombran las filas  
var <- colnames(tabla\_iii) # Se disponen los datos como % con 2 decimales  
tabla\_iii[, var] <- lapply(tabla\_iii[, var], percent\_format(accuracy = .01))  
rm(var)

# Para exportar tablas a formato Word  
ft <- flextable(data = tabla\_iii %>% add\_rownames()) %>%   
 theme\_zebra %>% autofit  
ft <- set\_caption(ft, caption = "Tabla III", style = "Table Caption")  
# Crea un archivo temp  
tmp <- tempfile(fileext = ".docx")  
# Crea un documento docx  
read\_docx() %>% body\_add\_flextable(ft) %>% print(target = tmp)  
browseURL(tmp) # abre el documento

**Tabla III**

|  | **mean\_acc\_TR** | **mean\_acc\_QR** | **mean\_acc\_LR** |
| --- | --- | --- | --- |
| Stock5 | 98.67% | 99.48% | 98.29% |
| Stock3 | 98.56% | 98.83% | 98.69% |
| Stock1 | 98.90% | 99.21% | 98.34% |
| Stock105 | 96.40% | 97.23% | 96.43% |
| Stock103 | 95.98% | 97.34% | 97.33% |
| Stock101 | 98.11% | 97.85% | 97.41% |
| Stock205 | 98.42% | 97.85% | 97.07% |
| Stock203 | 96.52% | 97.91% | 97.24% |
| Stock201 | 94.50% | 96.42% | 96.75% |

# (d) Conclusiones

La precisión en la clasificación de las transacciones individuales varía entre activos y de un día a otro (Tabla I), pero puede decirse, a grandes rasgos, que *Quote Rule* clasifica mejor que *Tick Rule* cuando únicamente se tienen en cuenta las transacciones que consiguen clasificar (Tabla II). Sin embargo, la cantidad de transacciones que *Quote Rule* deja sin clasificar es considerable. Por ejemplo, para el activo Stock5, el 31.73% de las transacciones quedan sin clasificar por *Quote Rule*, pero solamente es el 0.02% por *Tick Rule*. El algoritmo de *Lee & Ready* intenta subsanar esta limitación aplicando *Tick Rule* donde *Quote Rule* no clasifica. El resultado es un empeoramiento de la precisión de *Quote Rule* a cambio de una proporción mayor de transacciones clasificadas. Comparadas *Tick Rule* y *Lee & Ready*, que clasifican una proporción similar de transacciones, este último supone una mejora media en la precisión del 1.4%. En cualquier caso, esta mejora va acompañada de desviaciones típicas superiores a esa diferencia, con lo que resulta difícil aventurar cuál de los dos clasificaría mejor una serie cualquiera.

Por otro lado, se obtienen resultados de precisión sobre el volumen generalmente cercanos al 100% cuando se consideran volúmenes agregados y no transacción por transacción (Tabla III). Estos resultados dejan patente la capacidad de los tres algoritmos de predecir con un grado de acierto muy alto la proporción de compras y ventas producidas en un periodo de tiempo dado.