Documentacion R Script Dividendos

Resumen: el presente script realiza una extracción de información de la columna PROPORCION_O_IMPORTE que es un string de longitud muy variable. Se extrae información sobre el monto del dividendo, su divisa, su serie, cupones, importe y dividendo extraordinario.

Entrada: la tabla tb_derechos_decretados_bmv de Merc_Cap_DB; algúna parte de ella o los nuevos flujos de datos que la conforman.

Salida: la misma base de datos o flujo de datos de input pero con 19 columnas extras correspondientes a la extracción de información de la columna PROPORCION_O_IMPORTE. Las columnas extras son las siguientes:

Variable	Descripción
curr	referente a divisa
importe_brut	importe bruto
importe_net	importe neto
div1	primer dividendo mostrado en el string
serie1	serie del primer dividendo mostrado
cup_vig1	primer cupón vigente mostrado
cup_pag1	primer cupón de pago mostrado en el string
div2	segundo dividendo mostrado en el string
serie2	serie del segundo dividendo mostrado
cup_vig2	segundo cupón vigente mostrado
cup_pag2	segundo cupón de pago mostrado en el string
div3	tercer dividendo mostrado en el string
serie3	serie del tercer dividendo mostrado
cup_vig3	tercer cupón vigente mostrado
cup_pag3	tercer cupón de pago mostrado en el string
monto_extraord	Cantidad del monto extraordinario
monto_extraord2	Cantidad del segundo monto extraordinario

Consideraciones:

- Las 19 columnas solo podrán tener valores en aquellas observaciones que cumplan dos condiciones: pertenecer a algún dividendo de los siguientes: “0”,“1”,“1B”,“1E”,“3”,“41”,“CF”,“FE”,“FF”,“FH”;
- y haber otorgado un dividendo en efectivo, lo cuál se categoriza como tipo de derecho: DISTRIBUCION DE EFECTIVO, REEMBOLSO o DIVIDENDO EN EFECTIVO (o los valores “2”,“9”,“16”,“21” en la columna TipoDividendo)

Anexo: explicación a detalle de cada paso en el script

I. Ambiente

Se instalan 4 paquetes para la extracción.

```
requiredPackages  <- c("dplyr","stringr","stringi","strex")
ipak <- function(pkg){
  new.pkg <- pkg[!(pkg %in% installed.packages()[, "Package"])]
  if (length(new.pkg))
    install.packages(new.pkg, dependencies = TRUE)
  sapply(pkg, require, character.only = TRUE)
}
ipak(requiredPackages)
```

II. Datos

En esta parte se importan los datos. En el script se llaman **new_data** pero se refiere a cualquier base de datos de input. Posteriormente se recodifican las variables TipoValorStr y TipoDerecho para poder realizar los filtros previamente comentados y se agrega una variable del número de fila solo para mantener un control.

```
#Se importan los datos completos
new_data<-read.csv("new_data.csv", encoding="UTF-8")
new_data$TipoValorStr<-as.factor(new_data$TipoValorStr)
new_data$TipoDerecho<-as.factor(new_data$TipoDerecho)
new_data$id <- 1:nrow(new_data)
new_data<-new_data%>% select(id,PROPORCION_O_IMPORTE,TipoValorStr,TipoDerecho )

subset_div<-c("0",
          "1 ",
          "1B",
          "1E",
          "3 ",
          "41",
          "CF",
          "FE",
          "FF",
          "FH")

subset_derecho<-c("2","9","16","21")
dividendos<-new_data%>% filter(TipoValorStr %in% subset_div) %>% filter(TipoDerecho %in% subset_derecho)
```

III. Prelimpieza

En esta parte se preparan los datos para poder realizar las extracciones de manera exitosa. En primer creamos la variable cleaning al quitar algunas frases del string de la variable PROPORCION_O_IMPORTE. Asimismo, creamos una serie de variables auxiliares que nos ayudarán a la extracción como cantidad de signos $, si menciona la palabra “BRUTO”, si menciona la palabra “EXTRAORDINARIO”.

```
# Se quitan palabras ruidosas
limpieza<-dividendos %>%
  mutate(cleaning=sub("DE DONDE.*", "", PROPORCION_O_IMPORTE),
         cleaning=sub("EN DONDE.*", "", cleaning),
         cleaning=sub("DONDE.*", "", cleaning),
         cleaning=sub("DE LOS CUALES.*", "", cleaning),
         cleaning=sub("DIVIDI.*", "", cleaning),
```

```
         cleaning=sub("CONFOR.*", "", cleaning),
         cleaning=sub("CADA UNO.*", "", cleaning),
         cleaning=sub("COMPUEST.*", "", cleaning),
         cleaning=sub("LO QUE REP.*", "", cleaning),
         cleaning=sub("DI[ ]*VIDIDO.*", "", cleaning),
         cleaning=sub("POR CERT.*", "", cleaning),
         cleaning=sub("POR EL PER.*", "", cleaning),
         cleaning=sub("DISTRIBUC.*", "", cleaning),

         q_montos=str_count(cleaning,pattern ="\\$"),
         q_cupones=str_count(cleaning,pattern ="CUPON|CUPÓN"),
         q_series=str_count(cleaning,pattern ="SERIE"),
         extraord=str_count(cleaning,pattern ="EXTRAORD"),
         importe=str_count(cleaning,pattern ="BRUTO"),
         cleaning=str_remove_all(cleaning,pattern = "[ ]*-[ ]*PESO[ ]*MEXICANO"),
         cleaning=str_remove_all(cleaning,pattern = " DE"),
         cleaning=str_remove_all(cleaning,pattern = " - DÓLAR AMERICANO"),
         cleaning=str_remove_all(cleaning,pattern = " - EURO"))
limpieza$PROPORCION_O_IMPORTE<-NULL
```

IV. Extracción

Importes y dividendos extraordinarios

Ahora comienza la extracción de información.

En primer lugar con la expresión regular arpoon_cifras extraemos un vector con todas las cifras que tienen punto decimal (esto para obtener toda la canitdad de montos de divisas e importes).

- Si la variable cleaning contiene la palabra EXTRAORDINARIO y un solo signo $: asignamosen la variable monto_extraord el primer elemento del vector de cifras (de lo contrario NA).
- Si la variable cleaning contiene la palabra EXTRAORDINARIO y dos o más signos $: asignamosen la variable monto_extraord el segundo elemento del vector de cifras (de lo contrario monto_extraord); y en la variable monto_extraord2, el tercer elemento (de lo contrario NA)

En segundo lugar se extrae la divisa:

- la variable curr extrae cualquier match que se tenga con la expresión regular arpooon_curr

En tercer lugar se vacio la información relativa a importes bruto y neto:

- si la variable cleaning contiene la palabra BRUTO, se le asigna a la variable imp_brut el primer elemento del vector de cifras, de lo contrario NA.
- si la variable cleaning contiene la palabra BRUTO, se le asigna a la variable imp_net el segundo elemento del vector de cifras, de lo contrario NA.
- finalmente si la variable cleaning contiene la palabra NETO y cuenta con NA en imp_brut, se le asigna el primer elemento del vector de cifras, de lo contrario imp_brut.

```
## MONTOS
### EXTRA
arpoon_cifras<- "[0-9]*[.][ ]*[0-9]{1,15}"
limpieza<-limpieza %>% mutate(vector_montos=(str_extract_all(cleaning,arpoon_cifras)))
arpooon_curr<- "MXN |EUR|USD |M.N |MXV|M.N."
limpieza<-limpieza %>% mutate(vector_curr=(str_extract_all(cleaning,arpooon_curr)))

limpieza<-limpieza %>% mutate(monto_extraord=ifelse(extraord==1 & q_montos==1,str_extract(cleaning,arpo
```

```
                                monto_extraord=ifelse(extraord==1 & q_montos==2,stri_extract_last(cleaning
                                monto_extraord=ifelse(extraord==1 & q_montos==3,
                                                      as.numeric(as.character(lapply(vector_montos, `[`, 1
                                                      ,monto_extraord),
                                monto_extraor2=ifelse(extraord==1 & q_montos==3,
                                                      as.numeric(as.character(lapply(vector_montos, `[`, 3
                                                      ,NA),# cifras extraordinarias
                                curr=str_extract(cleaning,arpooon_curr),# currencies
                                #importes
                                importe_brut=ifelse(str_detect(cleaning, "BRUTO"),as.numeric(as.character
                                importe_net=ifelse(str_detect(cleaning, "BRUTO"),as.numeric(as.character(
                                importe_net=ifelse(str_detect(cleaning, "NETO") & is.na(importe_net),as.nu
```

Cupones

En el caso de cupones, usamos la expresión regular arpoon_cupones para extraer un vector de números enteros. Lo primero que hacemos es vaciar el vector de manera general y luego realizamos una corrección.

- la variable cup_vig1 toma el valor el primer elemento del vector de enteros, de lo contrario NA.
- la variable cup_vig2 toma el valor el tercer elemento del vector de enteros, de lo contrario NA.
- la variable cup_vig3 toma el valor el quinto elemento del vector de enteros, de lo contrario NA.
- la variable cup_pag1 toma el valor el segundo elemento del vector de enteros, de lo contrario NA.
- la variable cup_pag2 toma el valor el cuarto elemento del vector de enteros, de lo contrario NA.
- la variable cup_pag3 toma el valor el sexto elemento del vector de enteros, de lo contrario NA.
- Posteriormente se hace una corrección: si existen valores en cup_vig1 pero no en cup_pag1, cambiar los valores de tal manera que cup_pag1 ahora no tenga NA pero cup_vig1 si.

```
arpoon_cupones<- "[A-Z][ ]{1,}[0-9]{1,3} |[A-Z][ ]{1,}[0-9]{1,3}\\.$|[A-Z][ ]{1,}[0-9]{1,3}\\.[ ]{1,}|[A
arpoon_integer<-"[0-9]{1,3}"
limpieza<-limpieza %>% mutate(
  vector_cupones=(str_extract_all(cleaning,arpoon_cupones)))


limpieza<-limpieza%>% mutate(
  cup_vig1= ifelse(q_cupones>0,as.numeric(str_extract(as.character(lapply(vector_cupones, `[`, 1)),arpoo
  cup_vig2= ifelse(q_cupones>0,as.numeric(str_extract(as.character(lapply(vector_cupones, `[`, 3)),arpoo
  cup_vig3= ifelse(q_cupones>0,as.numeric(str_extract(as.character(lapply(vector_cupones, `[`, 5)),arpoo
  cup_pag1= ifelse(q_cupones>0,as.numeric(str_extract(as.character(lapply(vector_cupones, `[`, 2)),arpoo
  cup_pag2= ifelse(q_cupones>0,as.numeric(str_extract(as.character(lapply(vector_cupones, `[`, 4)),arpoo
  cup_pag3= ifelse(q_cupones>0,as.numeric(str_extract(as.character(lapply(vector_cupones, `[`, 6)),arpoo
  cup_aux=cup_vig1,
  cup_aux2=cup_pag1,
  cup_vig1=ifelse(!is.na(cup_aux)&is.na(cup_aux2),NA,cup_vig1),
  cup_pag1=ifelse(!is.na(cup_aux)&is.na(cup_aux2),cup_aux,cup_pag1) # aquellas obs con un solo cuponse
)
```

SERIES

Para la serie es muy sencillo, extraemos el vector de series con la expresión regular arpoon_series y vaciamos el primer elemento a serie1, el segundo a serie2 y el tercero a serie3, de lo contrario NA para cada caso.

```
arpoon_series <- "['].{1,5}[']|[\"].{1,5}[\"]"
arpoon_string<-".{1,5}"
limpieza<-limpieza %>% mutate(
  vector_series=(str_extract_all(cleaning,arpoon_series)))

limpieza<-limpieza%>% mutate(
  serie1= (str_extract(as.character(lapply(vector_series, `[`, 1)),arpoon_string)),
  serie2= (str_extract(as.character(lapply(vector_series, `[`, 2)),arpoon_string)),
  serie3= (str_extract(as.character(lapply(vector_series, `[`, 3)),arpoon_string))
)
```

Dividendos

Finalmente vaciamos la información de dividendos aprovechando el vector de montos con numero décimal que teniamos anteriormente.

- Solamente para las filas que tengan NA en importe_net o monto_extraord, vaciar el primer elemento del vector en div1; el segundo en div2 y el tercer en div3.

En este caso el código dice monto1, monto2 y monto3 pero se cambia el nombre después.

```
limpieza<-limpieza%>% mutate(
  monto1= ifelse( is.na(importe_net) | is.na(monto_extraord), as.numeric(as.character(lapply(vector_mon
  monto2= ifelse( is.na(importe_net) | is.na(monto_extraord), as.numeric(as.character(lapply(vector_mon
  monto3= ifelse( is.na(importe_net) | is.na(monto_extraord), as.numeric(as.character(lapply(vector_mon
```

VI. CORRECCIONES

Finalmente vienen unas correcciones ad hoc.

- La primera es que si la serie1 tiene NA y se detecta “POR CPO” en la fila, se le asigna “CPO” en serie 1, de lo contrario serie1.
- La segunda menciona que si serie 1 contiene el string “CPO” y existe NA en serie2 y no hay NA en monto2, monto1 toma el valor de monto2, de lo contrario se queda con su mismo valor monto1.
- Bajo las mismas condiciones que el punto anterior, monto2 toma el valor de NA, de lo contrario conserva su valor monto2.
- Finalmente, si el TipoDerecho no es igual al 2 y existe un NA en curr, asignarle “MXN”, de lo contrario que mantenga su valor curr.

```
## CORRECCION CPO (si menciona explicitamente por cpo poner como serie 1 y quitar el monto 1 por el que
limpieza<-limpieza %>% mutate(serie1=ifelse(is.na(serie1) & str_detect(cleaning,"POR CPO"),"'CPO'",seri

  monto1=ifelse((serie1== "'CPO'" |serie1=="\"CPO\"") & is.na(serie2) & !is.na(monto2),
                              monto2,monto1),
                              monto2=ifelse((serie1== "'CPO'" |serie1=="\"CPO\"") & is.na(serie2) & !is
                                            NA,monto2))
##SUPUESTO MXN (si no se ha dicho una currency si no es del tipo de derecho 1E se le asginan pesos mexi
limpieza<-limpieza %>% mutate(curr=ifelse(TipoDerecho!=2 & is.na(curr),"MXN",curr))
summary(factor(limpieza$serie1))
```

VII FINALIZACION

Finalmente, mantenemos solo las variables: importe_brut,importe_net,monto1,serie1,cup_vig1,cup_pag1, monto2,serie2,cup_vig2,cup_pag2, monto3,serie3,cup_vig3,cup_pag3,monto_extraord,monto_extraord2 y

las podemos pegar a la base de datos original completa o filtrada. En este caso se hizo el merge con base en el número de fila pero se puede usar cualquier método. Como paso final se crea como output la base deseada con las nuevas variables.

```
## Selecciono las variables finales
final_clean<-limpieza %>% select(id,curr, importe_brut,importe_net,
                                 monto1,serie1,cup_vig1,cup_pag1,
                                 monto2,serie2,cup_vig2,cup_pag2,
                                 monto3,serie3,cup_vig3,cup_pag3,

                                 monto_extraord,monto_extraord2)

## Renombro
final_clean<-final_clean%>%rename(div1=monto1,div2=monto2,div3=monto3)

## REGRESO BASES DE DATOS FILTRADA Y COMPLETA
new_data<-read.csv("new_data.csv", encoding="UTF-8")
new_data$TipoValorStr<-as.factor(new_data$TipoValorStr)
new_data$TipoDerecho<-as.factor(new_data$TipoDerecho)
new_data$id <- 1:nrow(new_data)
dividendos<-new_data%>% filter(TipoValorStr %in% subset_div) %>% filter(TipoDerecho %in% subset_derecho)

subset_data<-merge(dividendos,final_clean, by="id")
subset_data$id<-NULL
subset_data$X.U.FEFF.tm<-NULL

write.csv(subset_data,"subset_data.csv")

complete_data<-merge(new_data,final_clean, by="id", all.x = T)
write.csv(complete_data,"complete_data.csv")
```

Anexo 2: Estructura de la variable de PROPORCION_O_IMPORTE

La variable tiene una estructura particular. En general se noto que una vez aplicado los dos filtros el string está en alguna de estas categorías:

- cuenta con informaciòn relacionada con un importe bruto y/o neto (uno o dos montos)
- cuenta con información relacionada con dividendo ordinario o extraordinario (uno, dos y hasta tres montos)
- cuenta con información relacionada con un dividendo, su cupón y serie (uno, dos y hasta tres montos)

En cuanto a los montos, se refieren a cualquier cifra con punto decimal. Por ello si se extraen todas las cifras con punto decimal para vaciar en las variables creadas se debe tener en consideración las categorías anteriores pues dependiendo ella es el órden de los montos en el vector de montos. Por ejemplo, si es un string con info de dividendos, cupones y series pues el primer elemento del vector del montos se refiere al dividendo1, el segundo al dividendo2 y el tercero al dividendo3. Otro ejemplo esque si es un string de importe, si hay dos montos pues el primero es el bruto y el segundo es el neto pero si solo hay un importe, este se refiere al neto. Finalmente, en cuanto al caso de dividendos extraordinarios, el primer elemento es el div1 y el resto son dividendos extraordinarios.

En cuanto a la moneda, esta se encuentra si se hace un match con los caractéres de la moneda, por ejemplo "MXN" o "EUR". Asimismo, aquí se hace un supuesto: para todos los dividendos que no pertenezcan al tipo 1E, se les imputa la moneda MXN.

En cuanto a la serie, en general vienen en orden y vienen entre comillas ' '.

En cuanto a los cupones, son siempre números enteros identificables porque tienen un espacio antes y después (" 19 "). Además en general vienen en pares y para cada cupón vigente viene uno de pago. Por lo que los elementos nones del vector de cupones (1,3,5) son los cupones vigentes 1,2 y 3; y los pares son los de pago respectivos. Solamente hay algunos pocos casos donde solo hay un cupon, en caso de que solo haya uno, este siempre es el de pago por lo que hay que tener cuidado aqui pues rompe el orden anterior.

Finalmente se realiza una corrección en cuanto a las series: en los casos donde explicitamente se mencione que el dividendo es "POR CPO", la serie se le pone "CPO" y si quedan dos montos pero solo una serie, entonces el monto2 se le asigna al monto1 (por lo que el monto 1 original se pierte) y se le asgina NA al monto 2.