ML no supervisado al Índice de Commpetitividad Global

Guillermo Bustos-Pérez

5/4/2020

Índice de Competitividad Global. Introducción y tidy de los datos

Introducción

El **Índice de Competitividad Global** es publicado anualmente por el Foro Económico Mundial para evaluar la competitividad de los diferentes países. Para cada país se evalúan 12 pilares. Para cada pilar se evalúan una serie de sub-pilares y sus correspondientes indicadores. Los pilares van desde el *Desempeño de las instituciones* (incluyendo como indicadores la seguridad, independencia del poder judiacial, incidencia del terrorismo, etc.) a la *Capacidad de innovación* (donde se evaluan como indicadores la diversificación de los trabajadores, publicacines científicas, patentes, etc.).
Los Pilares, Sub-pilares e indicadores **están formualdos de forma cuantitativa**.
Los datos aquí empeados pueden accederse en formato .xlsx:
http://reports.weforum.org/global-competitiveness-report-2019/downloads/?doing_wp_cron=1586016888.5028719902038574218750

Limpieza y procesamiento de los datos para el análisis

Aunque el .xlsx proporcionado por el Foro Económico Mundial presenta un buen aspecto, es necesario un procesado ya que **viola una buena cantidad de los principios del data tidy** (Wickham, 2014). Por ejemplo, la pertenencia de un "Sub-pillar" o un "Indicador" a determinado Pillar no aparece como una columna aparte, sino insertado en la misma columna.
En este apartado se muestra como corregir las violaciones del Data Tidy usando R.

Empezamos por cargar los datos. "Datos no disponibles", "no aplicables", etc. aparecen bajo diferentes denominaciones. Con el siguiente código son introducidos como Na's:

```
# Leer los datos
Comp_2019 <- read_excel("Data/WEF_GCI_4.0_2019_Dataset.xlsx",
                    sheet = "Data", skip = 3,
                    na = c("n/a", "N/Appl.", "not assessed", "Not assessed"))
```

Ahora podemos ir a las columnas y variables que queremos: la edición (2019), los atributos ("SCORE" y "VALUE"). La selección de estos dos atributos es importante ya que varios pilares no se reportan bajo el indicador SCORE, y en lugar de ello se emplea el indicador "VALUE". Por ejemplo, en el caso del *"Skillset of secondary-education graduates"* no aparece reportado como SCORE, sino como VALUE. A su vez la mayoría de las columnas de países en filas de VALUE aparecen vacías. **Lidiaremos con este problema a continuación**.
Pero antes también podemos aprovechar para librarnos de columnas que no nos interesan o que son redundantes.

```r
# Filtar los datos de interés y librarse de columnas redundantes/de no interés
Comp_2019 <- Comp_2019 %>% filter(
  Edition == 2019 &
    Attribute == "SCORE" | Attribute == "VALUE") %>%
  select(-c("Index", "Series Global ID", "Freeze date",
            "Series code (if applicable)", "Series order"
            ))
```

El data frame presenta los siguients problemas:

- Cuando el "Attribute" es igual a VALUE la mayoría de las columnas están vacías
- En algunos casos cuando Attribute es igual a SCORE la fila está vacía ya que los valores de ese mismo Pilar aparecen como VALUE
- Los Pilares están duplicados, ya que se duplican al aparecer como SCORE o VALUE

En resumen, debemos identificar filas que para las columnas de los países solo figuran Na's. En este caso basta con crear un vector booleano indicando si faltan todos los datos a partir de la columna 6 (primer país) a la última columna, y filtrar usando este booleano.
En el caso de los duplicados, se eliminan conservando prioritariamente aquellos en los que el atributo es SCORE. En caso de no haber duplicado conservamos el "Series Name" cuyo attribute sea VALUE.

```r
# Vector booleano indcando si faltan todos los datos de la fila desde
# la columna 6 hasta el final
ind <- apply(Comp_2019[, 6:ncol(Comp_2019)], 1, function(x) all(is.na(x)))

# Filtrar los datos usando el vector booleano
Comp_2019 <- Comp_2019[ !ind, ]

# Retirar duplicados de "Series Name",
# conservando prioritariamente aquellos en los que Attribute = SCORE
Comp_2019 <-
  setDT(Comp_2019)[, .SD[which.min(factor(Attribute, levels = c("SCORE","VALUE")))],
                               by=.(`Series name`)]
```

Ahora tenemos un formato de datos *aceptable*, pero sigue presentando una serie de deficiencias. Las puntuaciones de cada país son leidas como factores y las columnas están en un orden poco apropiado (a esto se suma que los países aparecen como columna y que los indicadores deberían figurar como columnas (de eso nos encargaremos más adelante).
Esto **se soluciona en tres pasos**:

1) Guardamos los paíeses en un data frame aparte que vaya desde Angola (primer país) hhasta la útlima columna y transformamos este data frame en numérico
2) Aprovechamos para cambiar el orden de las columnas según queramos
3) Volvmos a unir ambos data frames

```r
# Indicadores de cada país en un data frame nuevo y hacerlo numérico
Countries <- Comp_2019 %>% select(c(Angola: last_col()))
Countries[] <- lapply(Countries , function(x) as.numeric(as.character(x)))
```

```r
# Reordenar columnas
Comp_2019 <- Comp_2019 %>%  select(c("Edition", "Series type",
               "Series name", "Series units", "Attribute"))

# Unir indicadores numéricos de cada país con el orden apropiado de columnas
Comp_2019 <- cbind(Comp_2019, Countries)
```

Ya que estamos, podemos guardar los promedios de regions y el promedio general en un data frame diferente (puede resultar un indicador interesante para más adelante). Lo eliminamos del data frame que contiene los países.

```r
# Nuevo data frame con los promedios de cada región
Averages <- Comp_2019 %>% select(`Series name`, `Series type`,
                            `Europe and North America`:`Sample average`)

# Eliminar los promedios regionales del data frame con los países
Comp_2019 <- Comp_2019 %>%
  select(-c(`Europe and North America`:`Sample average`))
```

Ahora eliminamos dos filas que han quedado y que no son útiles. Estas son el "Index", que representa el promedio de todos los índices (se emplean para computar la posición final del ranking), y la etiqueta (no se empela en los cálculos):

```r
# Filtrar casos no apropiados (ruido)
Comp_2019 <- Comp_2019 %>% filter(`Series type` != "Label (does not enter calculation)" &
                                  `Series type` != "Index")
```

Ahora necesitamos crear una **columna que represente el "Pillar"** al que pertenece cada indicador. Nuevamente esto es sencillo. Basta con crear una columna duplicada de "Series Name", asignar Na a todos aquellos que no contengan "pillar", y hacer un autocompletado hacia adelante (**na.locf()**) del paquete **zoo**.

```r
# Crear columna Pillar como copia de Series Name
Comp_2019$Pillar <- Comp_2019$`Series name`

# Asignar Na si no contiene "pillar"
Comp_2019$Pillar <- ifelse(grepl("pillar" , Comp_2019$Pillar), Comp_2019$Pillar, NA)

# Autocompletado haciia adelante
library(zoo)
Comp_2019$Pillar <- na.locf(Comp_2019$Pillar)

# Reordenar columnas
Comp_2019 <- Comp_2019 %>% select("Edition", "Series type",
            "Series name", "Pillar", "Series units", Angola: last_col())
```

Una vez realizados todos estos pasos podemos reordemnar definitivamente las columnas y proceder con el análisis de datos.