Zingtree Visualizations

MAO

27-06-2019

Setup

Librerías:

```
library(tidyverse)
library(jsonlite)
library(magrittr)
```

I. Funciones de Extracción de Datos desde Zingtree.

I.a. Detalle de sesiones (session_id) por rango de fechas.

- **Nombre de consulta API**: agent_sessions
- **Estructura de consulta API**: 'https://zingtree.com/api/agent_sessions/{{apikey}}/{{agent}}/{{dateini}}/{{datefi

Donde: apikey = Clave de acceso para realizar la consulta API agent = Agentes a considerar en la consulta (* son todos) dateini y datefin = fechas de inicio y término de la consulta

Función de extracción de sesiones para el período y agentes:

```
Agent_Sessions <- function (apikey, agent, dateini, datefin) {

  #Importando desde URL Zingtree, las variables "session_id", "start_time" (renombrándola "session_date
  agent_sessions <- fromJSON(txt = str_c("https://zingtree.com/api/agent_sessions/", apikey, "/", agent
    .$sessions %>%
    select(., session_id, session_date = start_time, agent) %>%
    as_tibble(.)
}
```

I.b. Detalle de datos para una sesión del período (id de cada sesión, de función I.a).

- **Nombre de consulta API**: get_form_data
- **Estructura de consulta API**: https://zingtree.com/api/session/{{session_id}}/get_form_data

Donde: session_id = Id de la sesión (sale de resultado de función Agent_Sessions: agent_sessions$session_id)

Función de extracción de datos para una sesión (id determinado):

```
Get_Form_Data <- function(session_id) {
  fromJSON(txt = str_c("https://zingtree.com/api/session/", session_id, "/get_form_data")) %>%
    as_tibble(.)
}
```

II. Funciones para creación de insumos con los cuales construir el detalle de una sesión del período.

II.a. Función Crit_Mov:

Función Crit_Mov, crea Tibble de movimientos críticos (varios a una), a partir de un registro de session_data:

```
Crit_Mov <- function(session_data) {

  # Parámetro para distinguir sesiones sin movimientos críticos:
  numcritmov <- session_data %>%
    colnames(.) %>%
    str_subset(., "CritMov") %>%
    length(.)

  # Función propiamente tal:
  if (numcritmov > 0) {
    crit_mov <- session_data %>%
      select(., -(starts_with("Sympt_Diagn_Sol") | starts_with("TG"))) %>%
      gather(., which(str_starts(colnames(.), "CritMov")), key = "CritMov", value = "CritMov_Value") %>%
      mutate(., Rama = str_sub(CritMov, start =9, end =11), .after = Session)
    return(crit_mov)
  } else {
    crit_mov <- session_data %>%
      mutate(., CritMov = "", CritMov_Value = "") %>%
      mutate(., Rama = "", .after = Session)
    return(crit_mov)
  }

}
```

II.b. Función Sympt_Diagn_Sol:

Función Sympt_Diagn_Sol. Idem Crit_MOv, con síntomas, diagnósticos y soluciones asociados a cada movimiento crítico de la sesión (1 o más):

```
Sympt_Diagn_Sol <- function (session_data) {

  # Parámetro para distinguir sesiones sin síntomas:
  numsympt <- session_data %>%
    colnames(.) %>%
    str_subset(., "Sympt_Diagn_Sol") %>%
    length(.)

  # Función propiamente tal:
```

```
  if (numsympt > 0) {
    sympt_diagn_sol <- session_data %>%
      select(., Session, starts_with("Sympt_Diagn_Sol")) %>%
      gather(., which(str_starts(colnames(.), "Sympt_Diagn_Sol")), key = "Sympt_Diagn_Sol", value = "Sy
      mutate(., Rama = str_sub(Sympt_Diagn_Sol, start = 17, end = 19), .after = Session)
    return(sympt_diagn_sol)
  } else {
    sympt_diagn_sol <- tibble(Session = session_data$Session[[1]], Sympt_Diagn_Sol = "", Sympt_Diagn_Sol
      mutate(., Rama = "", .after = Session)
    return(sympt_diagn_sol)
  }

}
```

II.c. Función Tech_Gest:

Función Tech_Gest y sus parámetros: Idem anteriores, para gestos técnicos, mediante vectores con detalle de gestos técnicos, anidados por zonas (ej de un vector: TG1_1_1, TG1_1_2, etc.). Última variable (TGs) es en base a listas de Tibbles (1 a varios gestos técnicos e intensidades). Se construye en base a vectores de gestos (detalle por sesión) y zonas (ramas del árbol por las que se pasea durante una sesión):

```
Tech_Gest <- function (session_data) {

  # Vector de gestos técnicos de misma rama, es decir, que contengan "TG" y terminen con un número (ej:
  gestos <- session_data %>%
    colnames(.) %>%
    str_subset(., "TG") %>%
    str_subset(., "[:digit:]$")

  # Vector con ramas, es decir, conjuntos de gestos técnicos para un diagnóstico (en base a primeros 5
  zonas <- gestos %>%
    str_sub(., start = 1, end = 5) %>%
    unique(.)

  # Parámetro para distinguir sesiones sin ramas ni, por tanto, sin gestos técnicos:
  numzonas <- length(zonas)

  # Función propiamente tal:
  if (numzonas > 0) {
    TG <- tibble()
    for(i in 1:length(zonas)) {
      TGloop <- session_data %>%
        select(., Session, str_subset(gestos, zonas[i])) %>%
        mutate(., Rama = str_sub(zonas[[i]], start = 3, end = 5), .after = Session) %>%
        pivot_longer(., cols = colnames(.[3:ncol(.)]), names_to = "TG", values_to = "TG_Val") %>%
        nest(., TGs = 3:4)
      TG <- bind_rows(TG, TGloop)
    }
    return(TG)
  } else {
    TG <- tibble(Session = session_data$Session[[1]], Rama = "", TGs = list(tibble(TG = "", TG_Val = ""
    return(TG)
  }
```

```
}
```

III. Consulta Final.

III.a. Parámetros de la consulta:

Se definen la clave para consultar (apikey), los agentes a consultar (* es igual a todos) y el período (determinado por fechas dateini y datefin):

```
apikey <- "4c92e2fd3f24f25d02308f6b8d7dd53c"
agent <- "*"
dateini <- "2022-02-21"
datefin <- "2022-02-25"
```

III.b. Tabla con detalle de sesiones del período de la consulta:

Detalle de sesiones para el período dateini - datefin, en tabla resultante (agent_sessions$session_id):

```
agent_sessions <- Agent_Sessions (apikey, agent, dateini, datefin)

  #Eliminando hora y dejando sólo la fecha de la variable "session_date".  Luego cambiando a formato 'd
  agent_sessions$session_date <- str_sub(agent_sessions$session_date, 1, 10) %>%
    as.Date(.)

  #Tibble obtenido:
  head(agent_sessions, 5)
```

```
## # A tibble: 4 x 3
##   session_id                       session_date agent
##   <chr>                            <date>       <chr>
## 1 cefce2ef777c4457a0d908fed5eaf536 2022-02-21   Zingtree+Hosted
## 2 e887a1ddceb2415b8a30574ddf6319ae 2022-02-22   Zingtree+Hosted
## 3 65bcae59b79e4da8ad039677b48f0ade 2022-02-25   Zingtree+Hosted
## 4 e734d0ff2a72478084ec81defea4ef61 2022-02-25   Zingtree+Hosted
```

III.c. Resultado final (tibble_final), como tabla con detalle de sesiones del período, cada una con su detalle de datos:

Consulta en base a loop para la totalidad de sesiones del período (en vector agent_sessions$session_id, obtenido en III.b.):

```
#Número de sesiones del período:
numsessions <- length(agent_sessions$session_id)

#Tibble consolidado del período, vacío:
tibble_loop <- tibble(Session = "", Rama = "")

#Loop
for (reg in 1:numsessions) {
```

```
  #Tibble con detalle de cada sesión (reg):
  session_data <- Get_Form_Data(agent_sessions$session_id[reg]) %>%
    select(., order(colnames(.))) %>%
    mutate(., Session = agent_sessions$session_id[reg], .before = colnames(.)[1])

  #Tibble con movimientos críticos de cada sesión (reg):
  crit_mov <- Crit_Mov(session_data)

  #Tibble con síntomas, diagnósticos y soluciones de cada sesión (reg):
  sympt_diagn_sol <- Sympt_Diagn_Sol(session_data)

  #Tibble con gestos técnicos de cada sesión (reg):
  TG <- Tech_Gest(session_data)

  #Tibble de una sesión (reg), en base a inner_join de tablas crit_mov, sympt_diagn_sol y TG:
  tibble_session <- inner_join(crit_mov, sympt_diagn_sol, by = c("Session", "Rama")) %>%
    inner_join(., TG, by = c("Session", "Rama"))

  #Tibble resultante del loop, en base a la unión de los tibble_session:
  tibble_loop <- bind_rows(tibble_loop, tibble_session)

}

# Tibble final, limpio de NAs, variables 3 en adelante ordenadas alfabéticamente y sin filas que no con
tibble_final <- tibble_loop  %>%
  replace(., is.na(.), "") %>%
  select(., order(colnames(.))) %>%
  relocate(., c("Session", "Rama"), .before = colnames(.)[[1]]) %>%
  filter(., Session != "")

#Tibble obtenido:
head(tibble_final, 5)
```

```
## # A tibble: 5 x 92
##   Session  Rama  Age   Analgesic Analgesic_Txt AntiInflam AntiInflam_Txt Biotype
##   <chr>    <chr> <chr> <chr>     <chr>         <chr>      <chr>          <chr>
## 1 cefce2e~ "3_2" "41"  "No Apli~ ""            "Ketorola~ ""             "Mesom~
## 2 e887a1d~ "1_1" ""    ""        ""            ""         ""             ""
## 3 e887a1d~ "3_3" ""    ""        ""            ""         ""             ""
## 4 65bcae5~ ""    ""    ""        ""            ""         ""             "Ectom~
## 5 e734d0f~ "1_3" "15"  "Otro"    ""            "Otro"     ""             "Mesom~
## # ... with 84 more variables: Biotype_Score <chr>, CritMov <chr>,
## #   CritMov_Value <chr>, Diagnosis <chr>, ElectroRoutine <chr>,
## #   ElectroRoutine_Txt <chr>, ExtracurricActiv <chr>, ExtraFreq <chr>,
## #   ExtraFreq_Score <chr>, FatPerc <chr>, FatPercProxy <chr>,
## #   FatPercProxy_Sore <chr>, FinalComments_Txt <chr>, FormFill <chr>,
## #   Genre <chr>, Height <chr>, ID_Code <chr>, Labor_Sit <chr>,
## #   Labor_Stand <chr>, Labor_StandMobile <chr>, ...
```