SURPASS-EcoPol 2019-2020 Database joining

Lican Martínez

25-07-2022

Resumen

Hay un archivo `.xlsx` correspondiente a cada colector. Cada archivo tiene diferentes hojas. Varias de estas hojas son homólogas entre sí, pero algunas son particulares del archivo de cada colector (ej. hoja `Todos los walking trails` en el archivo de MPA). Dentro de hojas homólogas (ej. `Recorrido_metadata`) pueden haber columnas particulares de cada colector, o columnas equivalentes pueden tener valores de diferentes *clases*. Un ejemplo de esto último (aunque encontré montones, por lo que solucionarlo manualmente sería bastante trabajo...) es la columna `T_(°C)` en la hoja `Recorrido_metadata`. Esta puede incluir únicamente valores numéricos en algunos archivos (ej. `VCP`) pero incluir texto en otros (ej. `MPA`). Como consecuencia, estas columnas se importan como de diferente *clase* (`numeric` y `character`) desde diferentes archivos.

El primer objetivo es obtener un excel que incluya todas las hojas existentes, y que aquellas hojas homólogas entre colectores se fusionen. En esta fusión se deberían: 1) apilar los registros de columnas homólogas y 2) agregar todas aquellas columnas que puedan ser particulares de sólo algunos colectores. Cada una de estas columnas "especiales" tendrá muchos valores faltantes (`NAs`) en las filas de la hoja de los colectores que no la consideraron. Las columnas compartidas entre hojas de distintos colectores, pero con diferentes clases, serán transformadas a `character`.

Procesado de datos

Importación

Lo primero que hago es importar todos los archivos. Como cada uno tiene diferentes hojas (tablas), armo una estructura anidada (jerárquica) de `data.frames` (DFs) en forma de `listas`. Cada hoja se importa a un DF, y todos los DFs de cada colector son agrupados en una lista (`sheets.ls.i`, donde `.i` refiere a que hay una por colector), y todas estas son agrupadas en una "meta-lista" (`excel.ls`) que contiene las listas de todos los colectores.

```
# archivos excel de entrada
input.filenames=c("Data_MoralesCL_Julio2022.xlsx",
                  "Data_ArbetmanMP25ene21.xlsx",
                  "Data_EEZ.xlsx",
                  "Data_Victoria_Campopiano.xlsx")

# códigos para los colectores
collector.codes=c('CLM', 'MPA', 'EEZ', 'VCR') # caro, maru, edu y vicky

# sheet names
sheet.names.ls=lapply(input.filenames, excel_sheets) %>%
    `names<-`(collector.codes)
```

```
# uinque sheet names
sheet.names=unlist(sheet.names.ls) %>% unique()


# para cada uno de los infputfiles (1 por colector) quiero una lista de tablas 'sheets.ls'
# cada una correspondiendo a una hoja del excel de ese colector
# estas listas estarán dentro de otra lista 'excel.ls'
excel.ls=list() # lista vacía a ir llenando

# loop a lo largo de los archivos de entrada (4 en este caso)
for (i in 1:length(input.filenames)) {

  # mensaje en consola para seguir la importación
  message(paste0('File:   ', input.filenames[i]))

  # Guarda los nombres de las hojas existentes en el excel i
  sheet.names.i=excel_sheets(input.filenames[i])


  # En el siguiente loop, cada una de estas hojas es imoporatda a un data.frame
  # y se guarda en una lista 'sheets.ls.i' (que luego será introducida en excel.ls).
  sheets.ls.i=list() # vacía para ir llenando

  for (j in 1:length(sheet.names.i)) {

    # lectura del archivo .xlsx del colector i, e importación de la hoja j como data.frame
    sheet.i=read_xlsx(input.filenames[i], sheet = sheet.names.i[j]) %>%
      as.data.frame()

    # guarda cada hoja en la lista de dataframes del colector i
    sheets.ls.i[[j]]=sheet.i
  }

  # asigna nombres a los elementos (data.frames en este caso) de la lista
  names(sheets.ls.i)=sheet.names.i

  # guarda la lsita de hojas del colector i en la meta-lista excel.ls
  excel.ls[[i]]=sheets.ls.i
}

# asgina nombres a los elementos de la meta lista excel.ls
names(excel.ls)=collector.codes
```

Apilado de las tablas homólogas

Algo importante es que, para cumplir el objetivo 1 para aquellas columnas con diferentes clases entre colectores, los valores de estas columnas deben ser transformados a la fuerza (*coerced*) a formato de texto (*character*). Luego de eso recién pueden ser apiladas.

```
# lista vacía para llenar con:
# el apilado de las versiones de todos los colectores para cada de cada tipo de hoja
stacked.sheets.ls=list()
```

```r
# para cada hoja
for (i in 1:length(sheet.names)) {

  # guardamos el nombre de la hoja correspondiente
  sheet.i=sheet.names[i]

  # liita de esa hoja, con cada elemento viniendo de un colector diferente
  sheet.i.ls=list() # vacia para llenar

  # índice que vamos a usar despues (no es importante, es un truquito nomás)
  coll.idx=1

  # para cada colector
  for (j in 1:length(collector.codes)) {

    # tomamos la hoja i del colector j
    excel.j_sheet.i=excel.ls[[j]][[sheet.i]]

    # si la hoja i está en el archivo del colector j
    if (!is.null(excel.j_sheet.i)) {

      # la gusrdamos en un dataframe excel.j_sheet.i
      excel.j_sheet.i=excel.j_sheet.i %>%
        as.data.frame() %>%
        # y le agregamos un identificador del colector
        mutate(coll_id=collector.codes[j])

      # y la agregamos a la lista de esa hoja
      sheet.i.ls[[coll.idx]] = excel.j_sheet.i

      # nombramos el nuevo elemento de la lista con el código del colector correspondiente
      names(sheet.i.ls)[coll.idx]=collector.codes[j]

      # truquito no importante
      coll.idx=coll.idx+1
    }
  }

  # guardamos todos los nombres de las columnas existenes en alguna versión de la hoja i
  all.colnames_sheet.i=sheet.i.ls %>%
    lapply(colnames) %>%
    unlist() %>%
    unique()

  # generamso una tabla vacía
  # para ir llenando con la clase que que presenta cada columna en las hojas
  # de los diferentes colectores
  colm.class.m=matrix(NA, ncol=length(collector.codes),
                      nrow=length(all.colnames_sheet.i)) %>%
    `rownames<-`(all.colnames_sheet.i) %>%
    `colnames<-`(collector.codes) %>%
    as.data.frame()
```

```
# para cada columna de la hoja i
for (k in 1:length(all.colnames_sheet.i)) {

  # nos fijamos la clase que esta tiene en la verisón de cada colector j
  for (j in 1:length(collector.codes)) {

    # primero chequeamos que esa columna esté en la verisón de la hoja de cada colector
    # porque algunas son exclusivas de algunos
    if (all.colnames_sheet.i[k] %in%
        colnames(sheet.i.ls[[collector.codes[j]]]) ) {

      sheet_i.j_k.column.class=
        sheet.i.ls[[collector.codes[j]]][,all.colnames_sheet.i[k]] %>% class()

    }else{
      sheet_i.j_k.column.class=NA
    }

    # guardamos las clases en la amtriz previamente generada
    colm.class.m[all.colnames_sheet.i[k], collector.codes[j]]=sheet_i.j_k.column.class[1]
  }
}

# agregamos una columna a la tabla colm.class.m para identificar que columnas de la hoja i
# debieron ser cmabiadas a clase character a la fuerza
colm.class.m$`Coerced to`=NA

# para cada columna
for (k in 1:length(all.colnames_sheet.i)) {
  if (# si hay mas de una clase en esa columna entre colectores
    colm.class.m[k,-(length(collector.codes)+1)] %>%
      c() %>%
      unlist() %>%
      unique() %>%
      length() != 1
      ) {

    # transformar esa columna a character en todas las hojas j
    for (j in 1:length(sheet.i.ls)) {

      # primero chequear que esa columna esté en la versión de cada colector
      if (all.colnames_sheet.i[k] %in% colnames(sheet.i.ls[[j]])) {

        # si es así se hace la conversión
        sheet.i.ls[[j]][,all.colnames_sheet.i[k]] =
          as.character(sheet.i.ls[[j]][,all.colnames_sheet.i[k]])
      }
    }

    # y se deja registro en la tabla generada
    colm.class.m$`Coerced to`[k]='character'

  }else{
```

```
      # si np se deja vacía
      colm.class.m$`Coerced to`[k]=' '
    }
  }

  # se imprime la tabla en el documento
  colm.class.m %>%
    knitr::kable(caption = sheet.i) %>%
    print()

  # ahora que todas las versiones de la hoja i tienen la misma clase para todas las columnas
  # ya sea porque asi era desde siempre,como porque fueron convertidas a character
  # podemos apilarlas
  # (por las dudas previamente ya habiamos agregado una columna que identificaba al colector)
  stacked.sheet.i=rbindlist(sheet.i.ls, fill=T, use.names = T)

  # guardamos el apilado de la hoja i en una lista
  stacked.sheets.ls[[i]]=stacked.sheet.i
}
```

Table 1: Recorrido_metadata

	CLM	MPA	EEZ	VCR	Coerced to
Date	POSIXct	POSIXct	POSIXct	POSIXct	
Site	character	character	character	character	
Walking_trail_ID	character	character	character	character	
Country	character	character	character	character	
Observer	character	character	character	character	
T_(°C) (estimated)	numeric	NA	NA	NA	character
Windspeed	character	character	character	character	
Humidity_(%)	numeric	character	numeric	numeric	character
Humidity_(description)	character	character	character	character	
Weather_(description)	character	character	character	character	
Time_start	character	character	POSIXct	POSIXct	character
Time_end	character	POSIXct	POSIXct	POSIXct	character
Total_time_(min)	character	character	character	numeric	character
Total_length_(km)	character	numeric	character	numeric	character
Habitat_Description	character	character	character	character	
Observation	character	NA	NA	NA	character
coll_id	character	character	character	character	
T_(°C)	NA	character	numeric	numeric	character
...16	NA	logical	NA	NA	character
...17	NA	logical	NA	NA	character
...18	NA	character	NA	NA	character

Table 2: Recorrido_plant data

	CLM	MPA	EEZ	VCR	Coerced to
Date	POSIXct	POSIXct	POSIXct	POSIXct	
Site	character	character	character	character	
Walking_trail_ID	character	character	character	character	

	CLM	MPA	EEZ	VCR	Coerced to
Plant sp	character	character	NA	NA	character
Abundance	numeric	character	numeric	numeric	character
Voucher_ID	character	logical	character	logical	character
Picture	character	character	character	character	
Observations	character	character	character	character	
...9	character	NA	NA	NA	character
coll_id	character	character	character	character	
Other pictures	NA	character	NA	NA	character
...10	NA	character	NA	NA	character
Suyai	NA	character	NA	NA	character
Caracteres clave	NA	character	NA	NA	character
...13	NA	logical	NA	NA	character
...14	NA	logical	NA	NA	character
...15	NA	character	NA	NA	character
Plant_sp	NA	NA	character	character	character

Table 3: Recorrido_Bombus data

	CLM	MPA	EEZ	VCR	Coerced to
Date	POSIXct	POSIXct	POSIXct	POSIXct	
Site	character	character	character	character	
Walking_trail ID:	character	character	NA	NA	character
Bombus_sp	character	character	character	character	
cast	character	character	character	character	
Plant visited	character	NA	NA	NA	character
Bombus_Collected Sample_ID	character	character	character	character	
Plant_voucher (when_needed)	character	character	character	logical	character
Habitat description	character	character	character	character	
Observation	character	NA	NA	NA	character
...11	logical	NA	NA	NA	character
coll_id	character	character	character	character	
Plant_visited	NA	character	character	character	character
...10	NA	character	NA	NA	character
Walking_trail_ID	NA	NA	character	character	character
Observations	NA	NA	character	character	character

Table 4: Transecta_metadata

	CLM	MPA	EEZ	VCR	Coerced to
Date	POSIXct	POSIXct	POSIXct	NA	character
Site	character	character	character	NA	character
Transect_ID	character	character	character	NA	character
Observer	character	character	character	NA	character
Time	POSIXct	POSIXct	POSIXct	NA	character
lat	logical	logical	logical	NA	character
long	logical	logical	logical	NA	character
alt	logical	logical	logical	NA	character
Time_period	character	character	character	NA	character
Transect_area	character	character	character	NA	character

	CLM	MPA	EEZ	VCR	Coerced to
Habitat_description	character	character	character	NA	character
coll_id	character	character	character	NA	character

Table 5: Transecta plant_data

	CLM	MPA	EEZ	VCR	Coerced to
Date	POSIXct	POSIXct	POSIXct	NA	character
Site	character	character	character	NA	character
Transect_ID	character	character	character	NA	character
Plant species	character	NA	NA	NA	character
N flowers per branch or per flowering patch	numeric	numeric	numeric	NA	character
N branches or iflorescences per plant or patches	numeric	numeric	numeric	NA	character
N plants or patches	numeric	numeric	numeric	NA	character
N_total_flowers	numeric	numeric	numeric	NA	character
Voucher_ID	character	character	character	NA	character
Observarciones	character	character	character	NA	character
coll_id	character	character	character	NA	character
Plant_species	NA	character	character	NA	character

Table 6: Transecta_Bombus data

	CLM	MPA	EEZ	VCR	Coerced to
Date	POSIXct	POSIXct	POSIXct	NA	character
Site	character	character	character	NA	character
Transect_ID	character	character	character	NA	character
Bumblebee_sp	character	character	character	NA	character
Cast	character	character	character	NA	character
Plant_species	character	character	character	NA	character
Observation	character	character	character	NA	character
...8	character	NA	NA	NA	character
coll_id	character	character	character	NA	character

Table 7: Samples_Bombus

	CLM	MPA	EEZ	VCR	Coerced to
Sample Number	numeric	numeric	numeric	numeric	
SampleCode	character	character	character	character	
Collector	character	character	character	character	
Species	character	character	character	character	
Date	POSIXct	character	POSIXct	POSIXct	character
Site	character	character	character	character	
Capture Method	character	character	character	character	
Preservation	character	character	character	character	
Country	character	character	character	character	
...10	character	NA	NA	NA	character
Plant_species	character	NA	NA	NA	character
...12	character	NA	NA	NA	character

	CLM	MPA	EEZ	VCR	Coerced to
coll_id	character	character	character	character	

Table 8: Voucher_plants

	CLM	MPA	EEZ	VCR	Coerced to
Voucher_ID	character	NA	NA	NA	character
Site	character	NA	NA	NA	character
Date	POSIXct	NA	NA	NA	character
Plant family	character	NA	NA	NA	character
Plant Genus	character	NA	NA	NA	character
Plant species	character	NA	NA	NA	character
Picture	character	NA	NA	NA	character
Descripcion	character	NA	NA	NA	character
...9	logical	NA	NA	NA	character
...10	logical	NA	NA	NA	character
...11	logical	NA	NA	NA	character
...12	logical	NA	NA	NA	character
...13	character	NA	NA	NA	character
coll_id	character	character	character	character	

Table 9: Todos los walking trails

	CLM	MPA	EEZ	VCR	Coerced to
#	NA	numeric	NA	NA	character
Date	NA	character	NA	NA	character
Sitio	NA	character	NA	NA	character
Código sitio	NA	character	NA	NA	character
Walking	NA	character	NA	NA	character
Transect	NA	character	NA	NA	character
Colect	NA	character	NA	NA	character
# BT/BR collected	NA	numeric	NA	NA	character
Walking trail	NA	character	NA	NA	character
Collector	NA	character	NA	NA	character
Lat Decimales	NA	character	NA	NA	character
Lon decimal	NA	character	NA	NA	character
...13	NA	character	NA	NA	character
...14	NA	character	NA	NA	character
Lat sexa	NA	character	NA	NA	character
Long sexa	NA	character	NA	NA	character
en Planilla	NA	character	NA	NA	character
...18	NA	character	NA	NA	character
coll_id	NA	character	NA	NA	character

Table 10: Habitat Types IUCN

	CLM	MPA	EEZ	VCR	Coerced to
1. Forest	NA	character	character	NA	character
coll_id	NA	character	character	NA	character

Table 11: Recorridos

	CLM	MPA	EEZ	VCR	Coerced to
Campaña	NA	NA	character	character	character
Tramo	NA	NA	numeric	numeric	character
Origen	NA	NA	character	character	character
Destino	NA	NA	character	character	character
Km	NA	NA	numeric	numeric	character
Notes	NA	NA	character	character	character
coll_id	NA	NA	character	character	character

```
# le ponemos nombres a esa lista
names(stacked.sheets.ls)=sheet.names
```

Exportación

```
# exportamos cada lista a una tabla que va a estar un una hoja diferente de un mismo .xlsx
# primero borramos versiones anteriores
unlink('./Stacked_excel.xlsx', recursive = T)

for (i in 1:length(stacked.sheets.ls)) {

  write.xlsx(stacked.sheets.ls[[i]],
             './Stacked_excel.xlsx',
             sheetName =names(stacked.sheets.ls)[i],
             append = T, row.names = F)
}
```