Trabajando con datos en profundidad, creando diagramas estratigráficos

Equipo UPWELL

28/10/2021

1. R basico

```
#install.packages(tidyverse)
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --
```

```
## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.5     v dplyr   1.0.7
## v tidyr   1.1.4     v stringr 1.4.0
## v readr   2.0.1     v forcats 0.5.1
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
#install.packages(here)
library(here)
```

```
## here() starts at /home/mat1506/Documents/workflows/Taller_git_github
```

```
#install.packages(patchwork)
library(patchwork)
#install.packages(ggplot2)
library(ggplot2)
#install.packages(remotes)
# remotes::install_github("paleolimbot/tidypaleo")
```

1.1 Directorio de trabajo

En este taller vamos a trabajar con el package `"here"` permite una fácil referencia de archivos mediante el uso del directorio de nivel superior de un proyecto de archivo para crear fácilmente rutas de archivo.

A modo de demostración, este artículo utiliza un proyecto de análisis de datos que vive en `r project_path`` en mi pc. Esta es la *raíz del proyecto*. Lo más probable es que la ruta sea diferente en su máquina, el paquete aquí ayuda a lidiar con esta situación.

Por ejemplo, el proyecto puede tener la siguiente estructura:

-Taller - carpeta data - carpeta analisis

Cuando se representa el archivo `report.Rmd` en nuestro proyecto, rmarkdown establece internamente el directorio de trabajo en `raíz del proyecto/análisis`.

Sin embargo, `misdatos.csv` todavía vive en el subdirectorio`data/` y el informe `report.Rmd` requiere que el archivo con `misdatos.csv` funcione.

1.2 El package "here" siempre usa rutas relativas al proyecto

Para representar el `report.Rmd`, debe asegurarse de que la ruta a `misdatos.csv` sea relativa al directorio `analisis/`, es decir, `../data/misdatos.csv`. Los trozos se unirían correctamente, pero no se podrían ejecutar en la consola ya que el directorio de trabajo en la consola *no es* `analisis/`.

El paquete `"here"` evita este problema al referirse siempre a la raíz del proyecto. Todos los archivos a los que se accede por `report.Rmd` deben ser referidos usando `here()`:

```
#here("data","misdatos.csv")
#here("data/misdatos.csv")
# en este caso el subdirectorio del proyecto es "Taller_git_github/data/"
here()
```

```
## [1] "/home/mat1506/Documents/workflows/Taller_git_github"
```

```
geodata <- read_csv(here::here("data/LEM_proxies.csv"))
```

```
## Rows: 501 Columns: 25
```

```
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (9): ID, Unit, Facie, Montmorillonite, Muscovite, Talc, Riebeckite, Cli...
## dbl (16): depth, MS, Den, FP, TS, TC, TIC, TOC, BioSi, TN, TOC/TN, d15N, d13...
```

```
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

2. Datos paleoambientales

Los datos paleoambientales hacen referencia a las entidades biológicas y geologícas inferidas a partir de los datos observables en el registro fósil. Algunos criterios `paleo` son útiles para identificar los cambios en los sustratos sedimentarios y otros registros del pasado para entender procesos a escalas de decadas a miles de años.

2.1 Miremos los datos geoquimicos de la laguna del Maule

```
mode(geodata)
class(geodata)
```

Para la primeras u últimas filas de los datos

```
head(geodata) #para la primeras filas de datos
tail(geodata) #para la ultimas filas de datos
```

Para ver la estructura de los datos

```
str(geodata)
```

3. Creando diagramas estratigráficos de la laguna del Maule

En este mini tutorial trabajaremos con los packages ggplot y tidypaleo “modificado del tutorial de Dewey Dunnington (2018)” para hacer una figura en profundidad de datos obtenidos por Romina en conchales.

```
library(ggplot2)
library(patchwork)
library(tidypaleo)
theme_set(theme_bw(8)) #este es el estilo del plot
```

Introduccion a Tidyverse

El tidyverse es una colección de paquetes R de código abierto introducidos por Hadley Wickham y su equipo que 'comparten una filosofía de diseño, una gramática y estructuras de datos subyacentes de datos ordenados (wikipedia).

Cómo añadir una columna con mutate

```
LEM <- mutate(geodata, ratio = TOC/TS)
```

Podemos ver que se agrega la nueva columna:

```
head(LEM)
```

Cómo crear subconjuntos con filter

```
lem <- filter(LEM, rate <= 0.5)
```

Cómo seleccionar columnas con select

```
lemplot <- select(LEM, MS, TOC, TIC,TN,BioSi,"TOC/TN")
filter(lemplot, TOC/TN <= 10)
```

El pipe: %>%

Con el paquete "dplyr", podemos realizar una serie de operaciones, por ejemplo select y entonces filter, enviando los resultados de una función a otra usando lo que se llama el pipe operator: %>%. Algunos detalles se incluyen a continuación. En "dplyr", podemos escribir código que se parece más a una descripción de lo que queremos hacer sin objetos intermedios:

```
LEM %>% select(MS, TOC, TIC,TN,BioSi,"TOC/TN") %>% filter(TOC< 2)
```

```
## # A tibble: 128 x 6
##       MS   TOC    TIC     TN BioSi `TOC/TN`
##    <dbl> <dbl>  <dbl>  <dbl> <dbl>    <dbl>
##  1  55.9  1.90 0      NA      18.8    NA
##  2  67.4  1.91 0.196  NA      20.2    NA
##  3  80.0  1.94 0.0277  0.21   19.2    10.8
##  4  NA    1.12 0.262  NA      19.4    NA
##  5  97.9  1.76 0.113  NA      20.4    NA
##  6  NA    1.89 0      NA      18.7    NA
##  7 114.   1.71 0.0132 NA      19.5    NA
##  8  NA    1.46 0       0.175  17.7     9.77
##  9 128.   1.26 0      NA      18.4    NA
## 10  NA    1.84 0      NA      19.8    NA
## # ... with 118 more rows
```

Conversión a datos 'ordenados'

Nos enfocaremos en `"tidyr"`, un paquete que provee un conjunto de herramientas que te ayudarán a ordenar datos desordenados. tidyr es parte del núcleo del tidyverse.

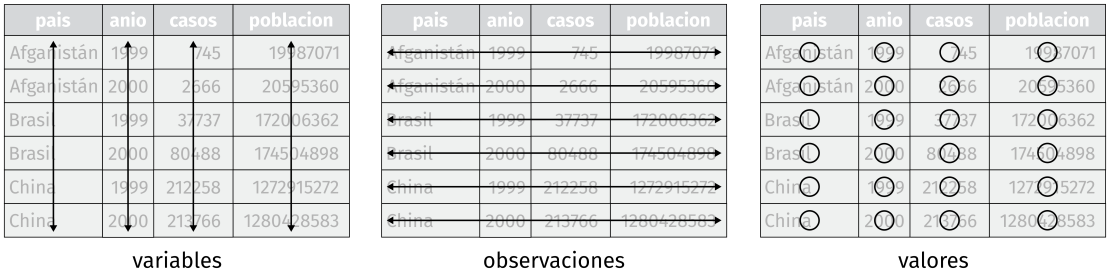
Veamos la serie de datos geoquímicos de la laguna del Maule

Pivotar

Los principios sobre datos ordenados parecen tan obvios que te podrías preguntar si alguna vez encontrarás un dataset que no esté ordenado. Desafortunadamente, gran parte de los datos que vas a encontrar están desordenados. Existen dos principales razones para esto:

- La mayoría de las personas no están familiarizadas con los principios de datos ordenados y es difícil derivarlos por cuenta propia a menos que pases mucho tiempo trabajando con datos.
- Los datos a menudo están organizados para facilitar tareas distintas del análisis. Por ejemplo, los datos se organizan para que su registro sea lo más sencillo posible.

```
knitr::include_graphics(here::here("tidy.png"))
```



Esto significa que para la mayoría de los análisis necesitarás hacer algún tipo de orden. El primer paso es entender siempre cuáles son las variables y las observaciones. Esto a veces es fácil; otras veces deberás consultar con quienes crearon el dataset. El segundo paso es resolver uno de los siguientes problemas frecuentes:

- Una variable se extiende por varias columnas
- Una observación está dispersa entre múltiples filas.

Datos "largos"

Veamos que tipo de datos son

```
head(geodata)
```

```
## # A tibble: 6 x 25
##   ID         Unit  Facie depth    MS   Den     FP    TS    TC    TIC   TOC BioSi
##   <chr>      <chr> <chr> <dbl> <dbl> <dbl>  <dbl> <dbl> <dbl>  <dbl> <dbl> <dbl>
## 1 CHILL-LEM~ U1    D1        0  44.8  1.21  0.883 0.175  2.42 0       2.42  19.6
## 2 CHILL-LEM~ U1    D1        1  55.9  1.39  0.773 0.230  1.90 0       1.90  18.8
## 3 CHILL-LEM~ U1    D1        2  NA   NA    NA     0.229  2.03 0       2.03  19.2
## 4 CHILL-LEM~ U1    D1        3  67.4  1.41  0.758 0.267  2.11 0.196   1.91  20.2
## 5 CHILL-LEM~ U1    D1        4  NA   NA    NA     0.255  2.07 0       2.07  19.3
## 6 CHILL-LEM~ U1    D1        5  80.0  1.44  0.740 0.286  1.97 0.0277  1.94  19.2
## # ... with 13 more variables: TN <dbl>, TOC/TN <dbl>, d15N <dbl>, d13C <dbl>,
## #   Montmorillonite <chr>, Muscovite <chr>, Talc <chr>, Riebeckite <chr>,
## #   Clinochlore <chr>, Microcline <chr>, Quartz <dbl>, Albite <dbl>,
## #   Amorphous <dbl>
```

Para ordenar un dataset como este necesitamos pivotar las columnas que no cumplen en un nuevo par de variables. Para describir dicha operación necesitamos tres parámetros:

- El conjunto de columnas cuyos nombres son valores y no variables.
- El nombre de la variable cuyos valores forman los nombres de las columnas. Llamaremos a esto key(clave).
- El nombre de la variable cuyos valores están repartidos por las celdas.

Con estos parámetros podemos utilizar la función pivot_longer() (pivotar a lo largo):

Para esto usaremos de ayuda de los paquetes que se llaman "tidypaleo" y "tidyverse"

```
# install.packages("remotes")
remotes::install_github("paleolimbot/tidypaleo")
```

```
## Using github PAT from envvar GITHUB_TOKEN
```

```
## Skipping install of 'tidypaleo' from a github remote, the SHA1 (1c22061b) has not changed since last
##   Use `force = TRUE` to force installation
```

```
library(tidypaleo)
```

Las columnas a girar quedan seleccionadas siguiendo el estilo de notación de dplyr::select(). En este caso hay varias columnas, que las listamos manualmente.

```
geodata_longer <- geodata %>%
    mutate(TOC.TN = (TOC/12)/(TN/14)) %>%
    dplyr::select(depth,
                  TOC,
                  TIC,
                  TC,
                  TN,
                  TOC.TN,
                  BioSi) %>%
    rename(`TOC/TN`=TOC.TN) %>%
    pivot_longer(cols =TOC:BioSi,names_to = "values", values_to = "count") %>%
    filter(values %in% c("TOC","TIC","TC","TN","TOC/TN","BioSi")) %>%
    mutate(values = fct_relevel(values,"TIC","TOC","TN","TOC/TN","BioSi"))
```

Y vemos los datos generados

```
head(geodata_longer)
```

```
## # A tibble: 6 x 3
##   depth values  count
##   <dbl> <fct>   <dbl>
## 1     0 TOC     2.42
## 2     0 TIC     0
## 3     0 TC      2.42
## 4     0 TN      0.278
## 5     0 TOC/TN 10.2
## 6     0 BioSi  19.6
```

4. Creando diagramas estratigráficos

En este mini tutorial trabajaremos con los packages ggplot y tidypaleo "modificado del tutorial de Dewey Dunnington (2018)" para hacer una figura en profundidad de datos obtenidos por Romina en conchales.

```
library(ggplot2)
library(patchwork)
library(tidypaleo)
theme_set(theme_bw(8))
```

Cargando los datos que necesitaremos

```
zone_data <- tibble(ymin = c(38,140,240),
                    ymax = c(50,170,268),
                    xmin = -Inf, xmax = Inf)  ###### Stratiplot zone
```

Realizamos un Cluster plot

```
coniss <- geodata_longer %>% ###### Cluster plot
    nested_data(qualifiers = depth, key = values, value = count, trans = scale) %>%
    nested_chclust_coniss()
```

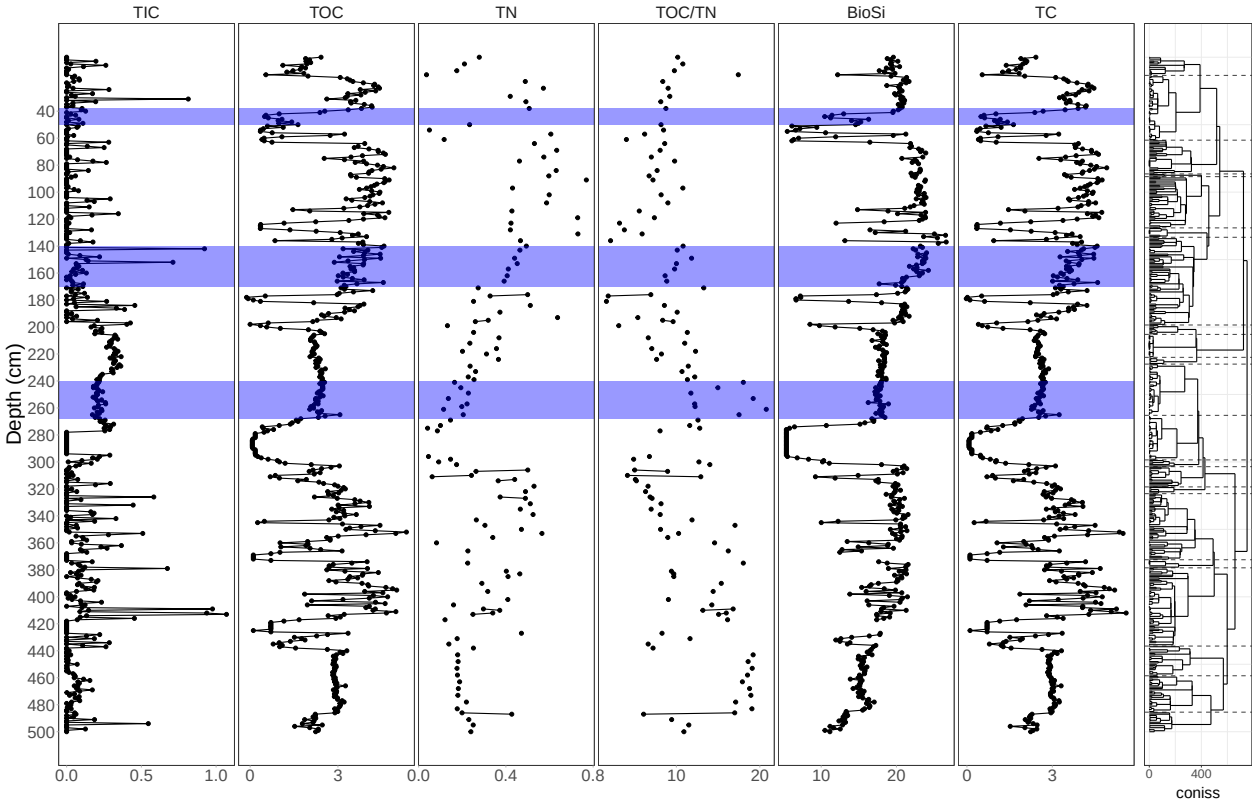
Realizamos el stratigraphic plots

```
alta_plot <- ggplot(geodata_longer,aes(x = count, y = depth)) +
    geom_lineh() +
    geom_point() +
    geom_rect(mapping = aes(ymin = ymin, ymax = ymax, xmin = xmin, xmax = xmax),
              data = zone_data,
              alpha = 0.4,
              fill = "blue",inherit.aes = FALSE ) +
    scale_y_reverse(breaks = c(40,60,80,100,120,140,
                               160,180,200,220,240,260,280,300,320,340,360,
                               380,400,420,440,460,480,500)) +
    facet_geochem_gridh(
      vars(values),
      default_units = "%") +
    labs(x = NULL, y = "Depth (cm)") +
    scale_x_continuous(breaks = scales::breaks_extended(n = 3)) +
    theme_set(theme_paleo(8)) +
    theme(
      text = element_text(size=20),
      axis.ticks = element_line(colour = "grey70", size = 0.3),
      panel.grid.major =element_blank(),
      panel.grid.minor = element_blank(),
      panel.background = element_rect(fill = "white", colour = "grey50")
    )

  wrap_plots(
    alta_plot +
      theme(strip.background = element_blank(), strip.text.y = element_blank()),
    ggplot() +
      layer_zone_boundaries(coniss, aes(y = depth)) +
      layer_dendrogram(coniss, aes(y = depth), param = "CONISS") +
      scale_x_continuous(breaks = scales::breaks_extended(n = 3)) +
      theme(axis.text.y.left = element_blank(),
            axis.ticks.y.left = element_blank(),
```

```
            text = element_text(size=15),
            panel.background = element_rect(fill = "white", colour = "grey50"))+
      labs(x = "coniss", y = NULL),
    nrow = 1,
    widths = c(8, 0.8)
  )
```

```
## Warning: Removed 788 rows containing missing values (geom_point).
```



Y finalmente exportamos

```
#png_out <- here::here("analysis/figures","Fig2.png")
#ggsave(png_out,  width = 25,height = 12, units = 'cm',device = "png")
```