StratigrapheR

an R package for integrated stratigraphy

Wouters S. *[1], **Da Silva** A.-C. [1], **Boulvain** F. [1] & **Devleeschouwer** X. [2]

1. University of Liège 2. Royal Belgian Institute of Natural Sciences
*Corresponding author [sebastien.wouters@doct.uliege.be]

Poster available on ORBi

Link to poster

StratigrapheR (version **0.0.6**) is an open-source integrated stratigraphy package. It is available in the free software environment R (https://CRAN.R-project.org/package=StratigrapheR, or see QR code below) and is designed to manage the large amount of data needed to perform cyclostratigraphy.

As cyclostratigraphy can be carried out by visual analysis on lithological observations and by time-series analyses, StratigrapheR endeavours to link the two by allowing the semi-automated generation of lithologs, the processing of stratigraphical information, and the visualisation of any plot along the lithologs in the R environment.

Link to StratigrapheR

1. How to draw a simple, long and monotonous litholog

This applies if you want to draw a simple litholog with beds as rectangles. To avoid drawing them one by one in a vectorial drawing software you can generate them in R in line commands. This can serve as base for more complicated lithologs, and the result can be imported by any vectorial drawing software (Inkscape, Adobe Illustrator, CorelDRAW,…).

1. Create a table of every relevant information for each bed.

Bed ID	Lower Limit (cm, m, ...)	Upper Limit (cm, m, ...)	Hardness (arbitrary)	Colour	Lithology	Lithology Code
B1	0	1	3	Grey	Shale	S
B2	1	3	4	Grey	Limestone	L
B3	3	4	5	Black	Chert	C
B4	4	9	4	White	Limestone	L
B5	9	11	4	White	Limestone	L
B6	11	12	5	Dark Grey	Chert	C
B7	12	14	5	Black	Chert	C
B8	14	17	3	Black	Shale	S
B9	17	19	5	Brown	Chert	C
B10	19	20	3	Grey	Shale	S

Table 1: example of table with lithological information

2. Import the table into R. A large set of R functions exist to import tables in formats such as .txt (such as the functions **`read.table`** and **`read.fwf`**), .csv (**`read.csv`**) or even .xls or .xlsx (see for instance the **`xlsx`** package on the CRAN website).

```
# Find a package to import .xlsx files
library(xlsx)
# Set the file directory where the data file is, and
# where your output will be
setwd("C:/Users/Guy/Desktop/Working_File")
# Import the file
table <- read.xlsx("Table.xlsx", 1, stringsAsFactors = F)
```

3. Create a data table (what is known as a data frame in the R vocabulary) of the rectangles coordinates for your log using the **`litholog()`** function.

```
# You will need to import the StratigrapheR package
library(StratigrapheR)
# Create the coordinates for each rectangle
log1 <- litholog(l = table$upper.limit,
                 r = table$lower.limit,
                 h = table$hardness,
                 i = table$bed_id)
```

```
> litholog1
    i dt xy
1  B1  1  0
2  B1  1  3
3  B1  0  3
4  B1  0  0
5  B2  3  0
6  B2  3  4
7  B2  1  4
8  B2  1  0
```

Fig 1: example of data frame made with litholog()

The **`l`** and **`r`** arguments in **`litholog()`** stand in a general sense for the left and right boundary of intervals. Both can equally deal with upper and lower bed limit.

The data frame created by the **`litholog()`** function (Fig. 1) provides an id for each polygon (each bed), and their coordinates for hardness (or arbitrary amplitude, identified as **`xy`**) versus depth or time (**`dt`**).

4. Make a personalised symbology for each lithology (or any kind of feature characterising an entire bed).

```
# Import dplyr for table/data frame manipulations
library(dplyr)
# Create a table for each symbology, in R basic
# graphical arguments (can be imported from table data),
# providing colour, shading density and angle, etc...
legend <- data.frame(Lithology_Code = c("S", "L", "C"),
                     col = c("grey30", "grey90", "white"),
                     density = c(30, 0, 10),
                     angle = c(180, 0, 45),
                     stringsAsFactors = F)
# Join the bed table with the legend by the lithological
# codes to define the symbology to each bed separately.
bed_legend <- left_join(table, legend,
                        by = "Lithology_Code")
```

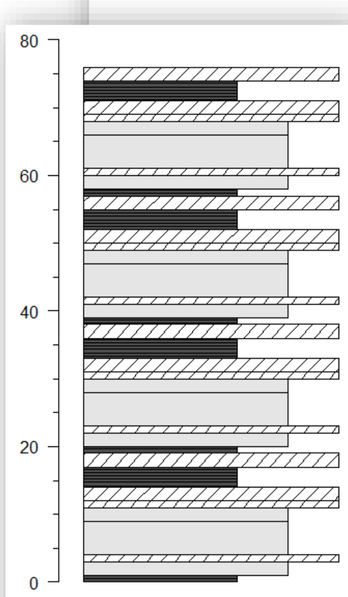

Fig 2: example of simple lithological log made by litholog() and multigons(). Note the three different type of beds: colour, shading density and angle are different for all three.

5. Plot the litholog (Fig. 2), using the **`multigons()`** function designed to plot multiple polygons.

```
# Import StratigrapheR if you
# have not done it already
library(StratigrapheR)
# Prepare the plotting environment
par(mar = c(1, 5, 1, 2))
plot.new()
plot.window(xlim = c(0, 5),
            ylim = c(-1, 77))
# Set a scale
minorAxis(side = 2, pos = -0.5,
          las = 1, n = 4)
# Plot the log with the adequate
# symbology for each bed (see Fig. 2)
multigons(i = log1$i,
          x = log1$xy,
          y = log1$dt,
          col =     bed_legend$col,
          density = bed_legend$density,
          angle =   bed_legend$angle)
```

2. How to add drawn elements (fossils, minerals, anything really…)

This allows to import in R simple drawings from graphical vectorial software (Inkscape, Adobe Illustrator, CorelDraw,…). They have to be saved as .svg files, a format that the all main graphical vectorial software can import and export with.

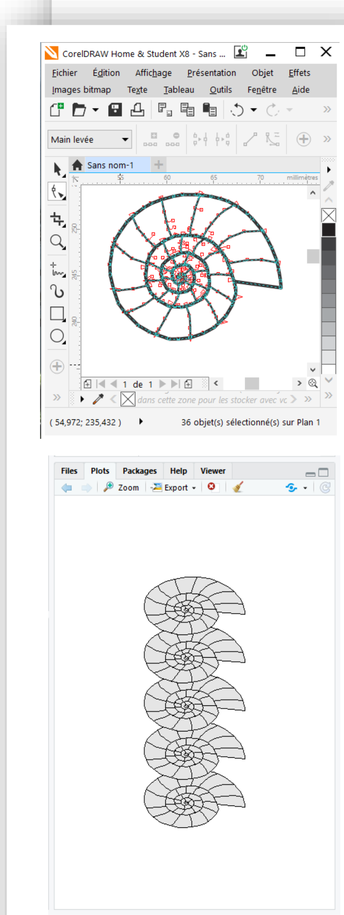

Fig 3: example of a simple drawing imported in R

1. Create a simple .svg made of lines, polylines, polygons and rectangles only (e.g. Fig. 3). These objects are made of nodes linked together by straight lines. Any other type of object will not be imported, as R graphics are not designed to accommodate them without deformations.

2. Import the .svg file using the **`pointsvg()`** function

```
library(StratigrapheR)
# Import an svg file (file.choose
# allows to select your file)
svg <- pointsvg(file.choose())
```

3. Draw it using the **`centresvg()`** or **`framesvg()`** functions, designed for objects made of polylines and polygons together (Fig. 3) .

```
# Prepare plot
par(xaxs = "i", yaxs = "i",
    mar = c(1, 1, 1, 1))
plot.new()
plot.window(xlim = c(-1, 1),
            ylim = c(-1,7))
# Draw svg, in several
# locations if needed
centresvg(svg, x = 0, y = c(1:5),
          xfac = 0.6, yfac = 0.6,
          col = "grey90")
```

3. How to personalize beds

The basic rectangular beds generated by **`litholog()`** can be modified. If you want to replace one of these beds by a modified version, you first need to avoiding drawing it when you plot the other basic rectangular beds. That can be performed in the drawing function **`multigons()`**, and this functionality also exists for the functions **`multilines()`**, **`centresvg()`** and **`framesvg()`**. This is done by giving the name or index (the order in which it appears in the data) of the bed you want removed to the **`forget`** argument of these functions.

```
library(StratigrapheR)
# Forget the lower beds
multigons(i = log2$i, x = log2$xy, y = log2$dt,
          forget =c("B1“, "B2"), density = c(10, 0, 10, 0))
```

Modified versions of the beds can then be added. Beds can for instance be drawn with a side open (see Fig. 4A):

```
# Modify the lower bed so the bottom is open
open_bed <- log2[which(log2$i == "B1"),] # select bed
open_bed <- shift(open_bed, 1) # change order of points

# Plot the background for symbology, and the outline with one
 # side open. You have to provide the symbology manually.
multigons(i = open_bed$i, x = open_bed$xy, y = open_bed$dt,
          density = 10, border = NA)
multilines(i = open_bed$i, x = open_bed$xy, y = open_bed$dt)
```

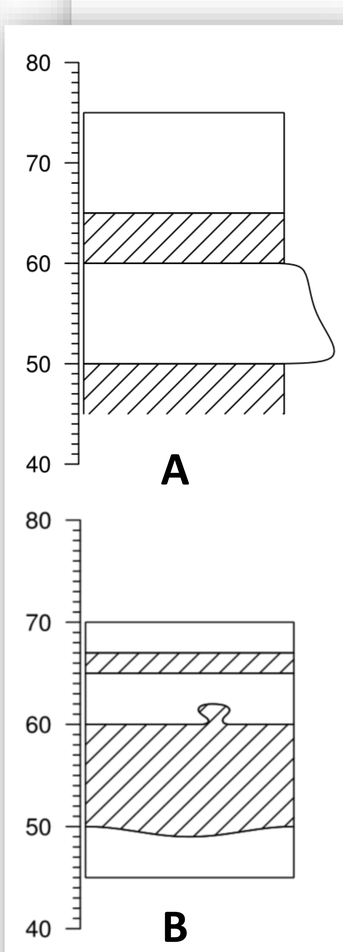

Fig 4: example of modifications of the basic lithological logs: [A] removing two beds, replacing them with a bed open on the lower side, and a bed imported from a .svg drawing, [B] modification of the bed boundaries

Or beds can be added as drawings imported by **`pointsvg()`** and positioned with pinpoint accuracy using **`framesvg()`** (see Fig. 4A):

```
# Add a bed drawn as a svg
framesvg(svg, 0, 5, 50, 60)
```

In a more automated way, bed boundaries can be modified directly on the data frame generated by **`litholog()`** using the **`weldlog()`** function. It allows to change the geometry of bed boundaries without altering data structure (Fig. 4B). It works by providing line segments to the function, and attributing them to specific bed boundaries. This way, you can provide the symbology of the beds as you would for a simple litholog (see part 1 of the poster: “1. How to draw a simple, long and monotonous litholog”).

```
# The functions here respectively
# import a .svg drawing and generate
# a sinusoid, both for a making a different
# segment to be used as bed boundary
l1 <- framesvg(svg, 0, 4, 0, 2,
               output = T, plot = F)
l2 <- sinpoint(4, 0, 1, phase = 0.5)
# Add them to the log, at the desired
# bed boundaries (see Fig. 4B)
log3 <- weldlog(log2, dt = c(60, 50),
                seg = list(seg1 = l1,
                           seg2 = l2))
# Plot
plot.new()
plot.window(xlim = c(-1,6),
            ylim = c(40, 80))
minorAxis(2, pos = -0.1, n = 10,las = 1)
multigons(log3$i, log3$xy, log3$dt,
          density = c(0, 10, 0, 10, 0))
```

4. How to visualise large R plots directly on your default PDF reader

Using the **`pdfDisplay()`** function, R plots of any size can directly be shown on any PDF reader (Fig. 5). Incremental padding of the names of the files (x_1.pdf, x_2.pdf, x_3.pdf, …) allows not having to close the reader at each modification. For Windows users, using the Sumatra PDF reader allows to modify a PDF file without closing the reader, removing the need to adapt/pad the file name at each change.

```
library(StratigrapheR)
# Put the graphical parts in function form
g1   <- function() { plot(1, 1) }
# Provide function to pdfDisplay
pdfDisplay(g1(),"Test")
```

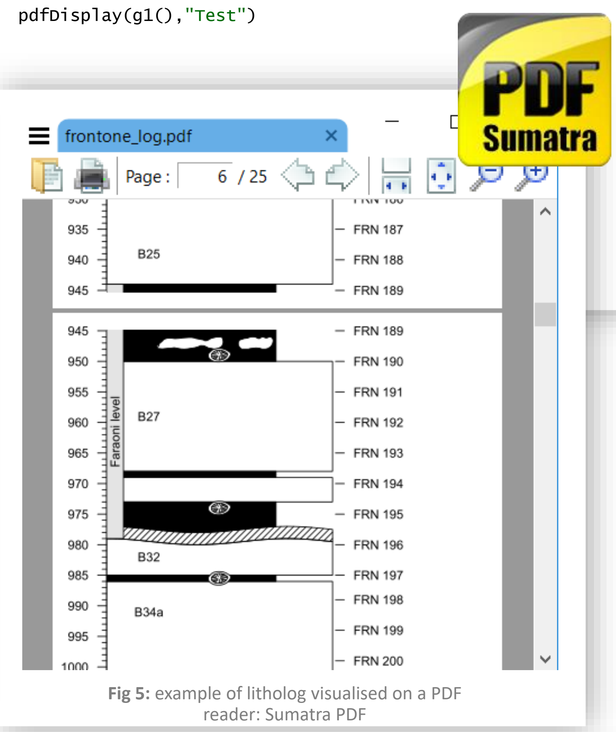

Fig 5: example of litholog visualised on a PDF reader: Sumatra PDF

Additional features

The StratigrapheR package furthermore allows basic visualisation (Fig. 6) and processing of oriented data used for magnetostratigraphy:

- Stereographic projections
- Zijderveld plots
- Conversion between data conventions
- Reorientation (sample correction, bedding correction, rotation)

It also provides a set of functions to deal with selected stratigraphic intervals (for instance in the [0,1[form): they allow simplification, merging, inversion and visualisation of intervals, as well as identifying the samples included in any given intervals, and characterising the relation of the intervals with each other (overlap, neighbouring, etc.).

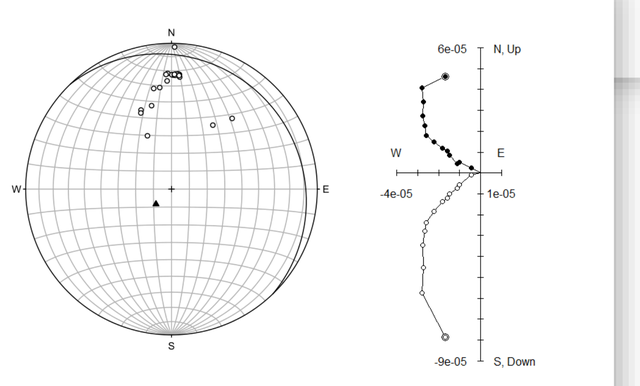

Fig 6: example of a spherical projection and a Zijderveld plot of palaeomagnetic data

Possibilities

- Automating litholog generation

- Plotting logs in parallel with other R figures (proxies, filtering results,…)

- Centralizing all relevant stratigraphical information in R

Prospects for the future

Adapting evolutive time-series analyses (EHA, wavelet,…) already existing as functions in R to be able to visualize their output in parallel with stratigraphical information and lithologs (complete or synthetic).

R can be an ideal environment to deal with all kind of stratigraphic data. Palaeomagnetic data of any format (MagIC, Utrecht, Rennes, Puffinplot,…) could be imported into a preformatted R convention for palaeomagnetic data (in a scheme known as a S3 object). This would allow to automate palaeomagnetic functions and data checking in R, and to allow direct translation from one data format to another.

The current R graphic system used for StratigrapheR can be upgraded (from R base graphics to Grid graphics) to allow for more features allowing among others better automatization.