An R package for sclero- and dentrochronologists: *sclero* tutorial

Mikko Vihtakari

December 17, 2014

Contents

1 Short introduction to the package

The *sclero* package for R is developed primarily as a tool for the sclerochronology community, but the functions can also be applied for other image measuring tasks. The package is developed to interact with ImageJ, a free-to-use public domain software for image processing and analysis. As the *sclero* package is developed for open source software, it is free to use and the code is modifiable by anyone interested. If these modifications are distributed in other packages/software, references to the original source (type `citation("sclero")` to R) and author(s) of the particular function are required. If you use the package in a scientific publication, please cite the package, as it is written as a volunteer contribution. The package is still at a developmental stage and users should critically evaluate the results of any function before publishing them. Contributions, code-fixes and problem reports are welcomed and should be done on GitHub.

The package is currently available as a developmental version on GitHub, and can be installed using the *devtools* package:

```
library(devtools)
install_github("MikkoVihtakari/sclero")
```

Anything documented in this tutorial or in the package documentation might be subject to changes. Currently *sclero* package contains following work flows:

1. Aligning sample spots with growth lines

2 Aligning sample spots with growth lines

Sample spots acquired with laser-ablation inductively-coupled-plasma mass-spectrometry (LA-ICP-MS) or similar techniques often need to be aligned to growth lines in order to spatially synchronize the samples along a chronologically deposited material. *Sclero* package together with ImageJ enables a semi-automatic work-flow of aligning these sample spots from photographs of the sampled material. In order to align the samples, the user has to define a measurement axis (called 'main axis' in the packge) to which all sample spots and growth lines will be aligned (Figures 1 and 4). The work-flow is applicable for samples where the main axis is parallel to the growth lines (such as tree cores and cross-sections of bivalve shells) and for samples where the main axis crosses the growth lines (tree and coral cross-sections for instance).

2.1 Prerequisites for sample spot alignment

In order to run the sample spot alignment, a photograph of sufficient resolution to see the growth lines and sample spots is required (Figure 1). In addition, the user should take care that following points are considered:

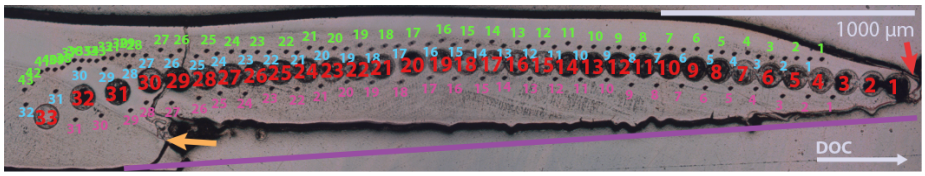


Figure 1: An example shell where the measurement axis (called 'main axis' in the package) is approximately parallel to the growth lines. Large holes marked with red numbers are the LA-ICP-MS samples, while the small holes marked with green, cyan and magenta numbers are the SIMS sample sequences to be aligned with growth lines. Red and orange arrows show the beginning and the end of the measurements, respectively. Purple line shows the measurement axis along which all the samples should be aligned. White arrow marked 'DOC' points towards the direction of growth.

Aspect ratio The aspect ratio of pixels in the photograph has to be 1:1, meaning that a certain length vertically equals to the same length horizontally. ImageJ has a 'Pixel Aspect Ratio' setting in 'Set Scale' option, but the output of this setting and how it affects the R input has not been tested by the package author.

File format The image file can be of any raster format compatible with ImageJ, but .tif images are generally recommended for editing purposes as this format is lossless meaning that it can be saved again without losing information. Please note that using vector image formats (.gif, .png, .esp, .pdf) is probably not a good idea when doing measurements from photographs.

Compiled images If you are using compiled images from a microscope software, pay a special attention in aligning the photographs correctly as this will affect the results. Make sure that all compiled images are taken with a same magnification.

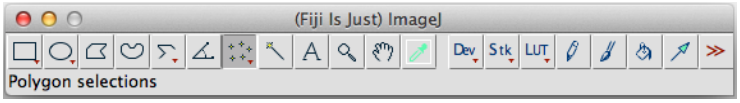
Scaling If you want real distances as output, the microscope has to be calibrated. If real distances are not important, knowing the microscope calibration is not essential as sample alignment is relative to one photograph. Currently there is no way to export the "Set Scale"

Cracks The sample material has to be chronologically deposited without spatial caps or cracks. If your sample material has large caps or cracks, they are bound to affect the output. One option is to edit the photograph in a photo editing program and remove these cracks.

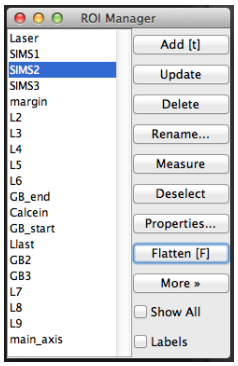
2.2 Mark sample spots and growth lines in ImageJ

The example shell has one sequence of samples taken with LA-ICP-MS (the large holes) and three separate sequences of Secondary Ion Mass Spectrometer (SIMS) samples (Figure 1). The workflow can also be applied for a single sequence of samples. First we have to mark the sample spots and growth lines using ImageJ. Click this link for instructions how to use ImageJ. Note that all marks with ImageJ should be done **against the direction of growth**. Otherwise the aligning functions might not work properly.

1. Start by using the 'Multi-point Tool' and mark LA-ICP-MS sample spots *against the direction of growth* (see Figure 2). The sample spots will be numbered in the order you mark the spots. In this example we begin from the margin (the red arrow in Figure 1).

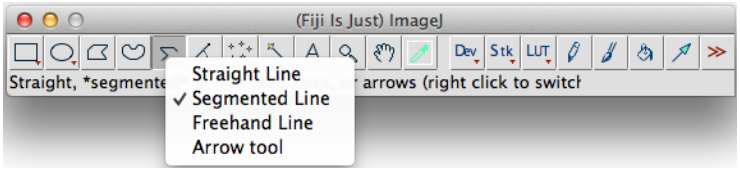


2. After this open the ROI manager and add the sequence using Add[t] button (the keyboard shortcut is *Ctrl* + *T* or *Cmd* + *T* depending on the OS). You can rename the ROI (Region Of Interest) to correspond the type of the object ("Laser") for your own reference. The ROI names can be assigned as object names in consequent R functions, although this behavior is not required nor default (see Section 2.3). If you decide to do this, use `_` or `.` as a separator marker. Do not use other special characters (`-` , `#`) or white space in ROI names, which you want to keep, because these will confuse the internal `grep` functions. ImageJ automatically generates ROI names containing `-` (0782-4756 for instance). These names are not valid `data.frame` names in R and will be regenerated by the functions in *sclero* package.



3. Then go on and mark the SIMS sequences and add each of them separately to the ROI manager.

4. After this mark the visible growth lines using 'Segmented Line' tool starting from the lower margin upwards *against the direction of growth*. Mark each line so that the sample spots are in between at least two lines, but it is not necessary to continue marking much further (Figure 2). Add each line separately to the ROI manager. The order of lines is not important as this can be changed later. Often it is easiest to start with the most clear growth lines and add less clear growth lines in between these lines as needed. Pay special attention that the growth lines do not cross each other. There must be at least one growth line before and after the first and last sampling spot. It might be helpful to rename the growth line ROIs, so that they are easier to associate later.



5. Once you are done with this, add the measurement axis using the Straight Line tool. It is very important to start the axis from where you want the measurements to begin as all distances will be scaled to this axis. The main axis should not cross any growth line, if the axis is intended to be approximately parallel to the growth lines (cores and shell sections). If your sample material is a cross-section of a tree or coral or an umbo region of a bivalve, the main axis has to cross all growth lines (see Section 2.4.1). Note that it is possible to have only one Straight Line per ImageJ .zip file as the internal functions recognize the straight line as the main axis automatically.

6. After this you can save the ROI collection as a zip file (More, Save...) naming it 'ijdata.zip'.

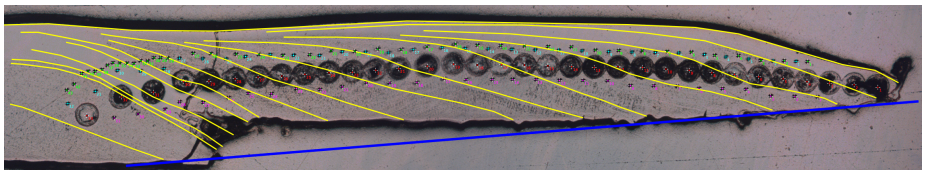


Figure 2: ImageJ annotated shell section. Small red, green, cyan and magenta numbers indicate the marked sample spots in different sequences. Yellow lines show the marked growth lines and the blue line shows the measurement (main) axis. The main axis should not cross any growth line when `type` is 'along'. If `type` is 'cross', the main axis has to cross all growth lines (see Section 2.4.1).

2.3 Reading ImageJ zip files containing ROI objects

Next we can open R and load *sclero* package.

```
library(sclero)
```

`read.ijdata`

After this we use `read.ijdata` function to read and process the ImageJ .zip file containing ROIs. The code below is possible to run without having saved 'ijdata.zip' file. If you followed the example above, replace 'path' with the location of your ImageJ .zip file.

```
path <- file.path(system.file("extdata", package = "sclero"), "ijdata.zip")
dat <- read.ijdata(path) # Replace 'path' with 'your_ImageJ_file.zip'
```

The function returns a list of class 'IJDATA' containing information about ROIs.

```
summary(dat)
```

```
               Length Class      Mode
spots.x         4     data.frame list
spots.y         4     data.frame list
gbs.x          15     data.frame list
gbs.y          15     data.frame list
main.x          1     data.frame list
main.y          1     data.frame list
sample.name     1     -none-     character
scaling.factor  1     -none-     numeric
unit            0     -none-     NULL
```

`order.ijdata`

At this point it can be of interest to reorder some of the ROIs. The order of ROIs does not make a difference in calculations of following `spot.dist` function, but the output will be given in the order the of IJDATA object. `order.ijdata(..., print.order = TRUE)` function can be used to check the order of ROIs.

```
order.ijdata(dat, print.order = TRUE)
```

```
$spots
           Laser d18O_1 d18O_2 d18O_3
col.number     1      2      3      4

$gbs
           margin L2 L3 L4 L5 L6 GB_end Calcein GB_start Llast GB2 GB3 L7 L8 L9
col.number      1  2  3  4  5  6      7       8        9    10  11  12 13 14 15
```

The same function can be used to reorder and subset ROIs within IJDATA objects. Subsetting can be done by leaving out the elements you do not want to include in the IJDATA object. In the case of subsetting the function prints a warning to make sure that the user did not forget to specify all elements within the object. Say that we want to reorder the growth lines:

```
dat <- order.ijdata(dat, gbs = c(1:6,13,14,15,7,11,12,8,9,10))
order.ijdata(dat, print.order = TRUE)
```

```
$spots
           Laser d18O_1 d18O_2 d18O_3
col.number     1      2      3      4

$gbs
           margin L2 L3 L4 L5 L6 L7 L8 L9 GB_end GB2 GB3 Calcein GB_start Llast
col.number      1  2  3  4  5  6  7  8  9     10  11  12      13       14    15
```

`convert.ijdata`

The information in IJDATA objects is meant to be user modifiable. The object behaves as any `list` in R only that the name of elements (`$spots.x`, `$spots.y`, `...`) should not be changed to ensure that subsequent functions behave correctly. In order to align the sample spots along the main axis, we need to convert the information to *spatstat* objects using `convert.ijdata` function.

```
shell <- convert.ijdata(dat)
```

The function returns a list of class 'rawDist', which can be plotted using the generic plotting function. The plot will be a representation of the ROIs where the coordinate system is flipped vertically. This is because of coordinate system differences between Java (ImageJ) and R.

```
plot(shell)
```

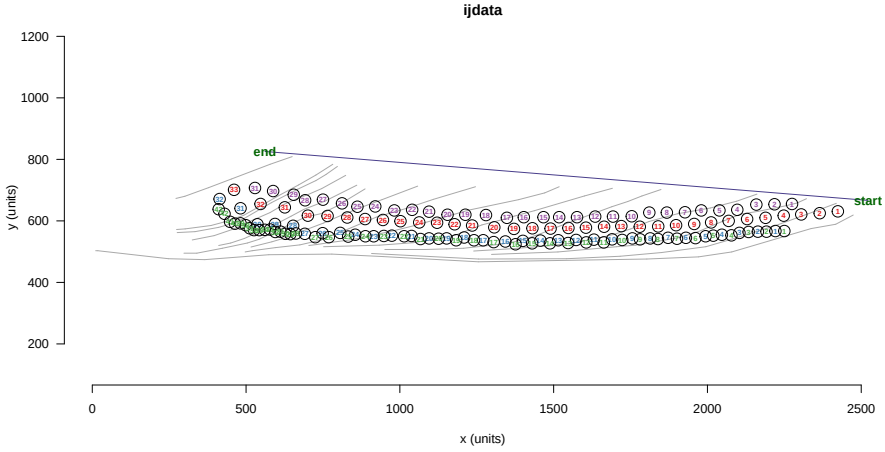


Figure 3: The digitalized representation of the shell section. Red, green, blue and purple numbers show the location of the sample spots. Grey lines represent the marked growth lines and the purple line the main axis to which the sample spots will be aligned.

2.4 Aligning sample spots

`spot.dist`

An object of class 'rawDist' can be readily processed using `spot.dist` function. First, the function projects the beginning of each growth line to the measurement axis (L_1 and L_2 in Figure 4). Then, the sample spots are aligned along the main axis in relation to the adjacent growth lines on both sides of a sample spot such that $d_1/d_2 = d_{L_1}/d_{L_2}$.

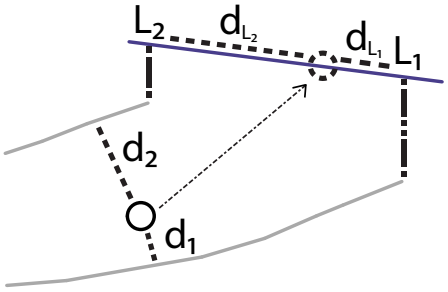


Figure 4: Alignment of sample spots along the measurement axis. Grey lines represent marked growth lines, solid circle a sample spot and dashed circle the aligned sample spot. Sample spots are aligned such that $d_1/d_2 = d_{L_1}/d_{L_2}$.

```
aligned <- spot.dist(shell)
```

The function returns a list of class 'spotDist' containing new information of the aligned sample spots and the shell sequence, which was already included in the 'rawDist' object. Also 'spotDist' objects can be plotted using the generic plotting command.

```
plot(aligned)
```

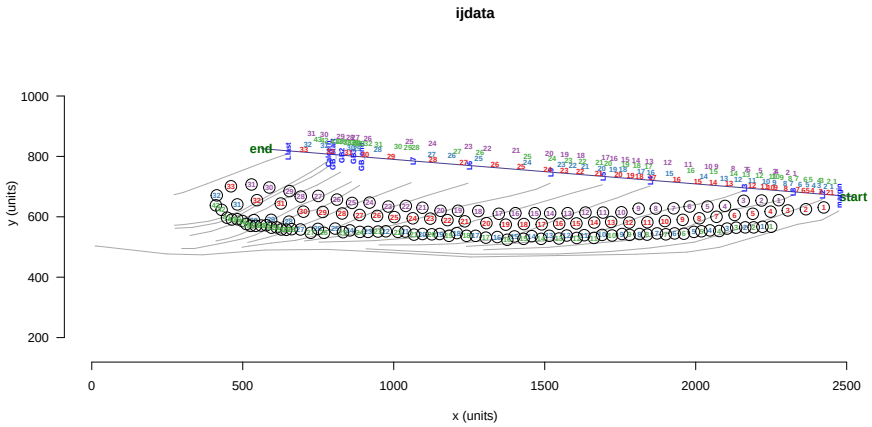


Figure 5: Aligned sample spots along the measurement axis together with the digitilized presentation from Figure 3.

```
summary(aligned)
```

	Length	Class	Mode
spots	4	-none-	list
gbs	5	psp	list
main	5	psp	list
window	4	owin	list
start.main	6	ppp	list
end.main	6	ppp	list
sample.name	1	-none-	character
scaling.factor	1	-none-	numeric
unit	0	-none-	NULL
main.type	1	-none-	character
gb.projections	6	ppp	list
gb.start	6	ppp	list
gb.end	6	ppp	list
gb.projections.dist	4	data.frame	list
mid.point.type	1	-none-	character

```
det.dat             1      -none-     list
output              4      -none-     list
```

The alignment information with sample spot numbers is stored as a sublist called `output` and can be extracted to a data.frame (see below). Otherwise the object behaves as any list in R. Relevant data can be subsetted as needed. Detailed data containing information of the alignment process is stored in a sublist called `det.data`.

```
aligned$output ## Results not shown to save space
aligned$det.data
```

2.4.1 `spot.dist` settings

The alignment function `spot.dist` can either project growth lines on the measurement axis or the crossing points can be chosen. do two types of alignment. The measurement axis type is automatically selected by the following criteria:

Cross Approximately round cross-sections: samples where the growth lines continue through the entire width of the sample (such as tree, coral or calcareous algae cross-sections and umbo-regions of bivalves). This type is selected by making the **measurement axis to cross each individual marked growth line**. The location of each growth line along the measurement axis is equal to the crossing point between the measurement axis and the growth line.

Along Samples with cut-off growth lines such as bivalve margin cross-sections and tree, sediment or ice-cores. This option is selected by placing the measurement axis such that **it does not cross any of the marked growth lines**. The location of each growth line is projected along the measurement axis from the beginning of the growth line (the point where you started marking the growth line in ImageJ).

These criteria are set due to the need of defining a location for each marked growth line along the measurement axis. The choice is rigid, to simplify calculations and to avoid bias in results by allowing two different methods for growth line locations. The easiest way to test which `type` suits a particular sample best is to save two sets of ImageJ zip files by moving the measurement axis.

2.4.2 `spot.dist` troubleshooting

Apart from the common issue of placing `main` axis a wrong way around (because where you start drawing the line is considered as the `start` point), problems with `spot.dist` failing to find a location for each sample spot are often connected with the lack of marked growth lines on both sides of each sample spot. If this is the case, try drawing more growth lines so that each sample spot is surrounded by them. If the nature of your sample material does not allow this, `spot.dist` function has an `alignment` option. Changing the value to `'traverse'` aligns

the sample spots as described in Fig X and is sometimes helpful especially if you try to align samples along an entire cross-section of a curved material, such as a bivalve shell. However, changing the value will change the way your sample spots are aligned and should be done consistently. The `'closest'` option for `alignment` works best for applications described in this tutorial.

All functions in this package are still at an experimental stage. They do work for the applications needed by the author, but might not work for other applications. They are also likely to contain bugs, which can be fixed. Please contact the package author, if you encounter unexpected behavior or clear errors in the functions.

2.5 Estimating spatial averaging error of sample spots

Implemented to the package. To be added here very soon.