riojaPlot: User Guide (Version 0.1-20)

Steve Juggins

2022-11-28

Contents

1. Introduction

`riojaPlot` plots a set of variables in a stratigraphic diagram. Diagrams can be plotted with lines, silhouettes, bars or symbols of any combination of these element. `riojaPlot` extends the older function `strat.plot` in several ways: variables can be grouped and displayed with different colours, a cumulative summary plot can be added automatically based on the grouping, a cluster dendrogram and resulting zones can be added automatically, a secondary y-axis can be added to plot both depth and age axes, and the figures margins are

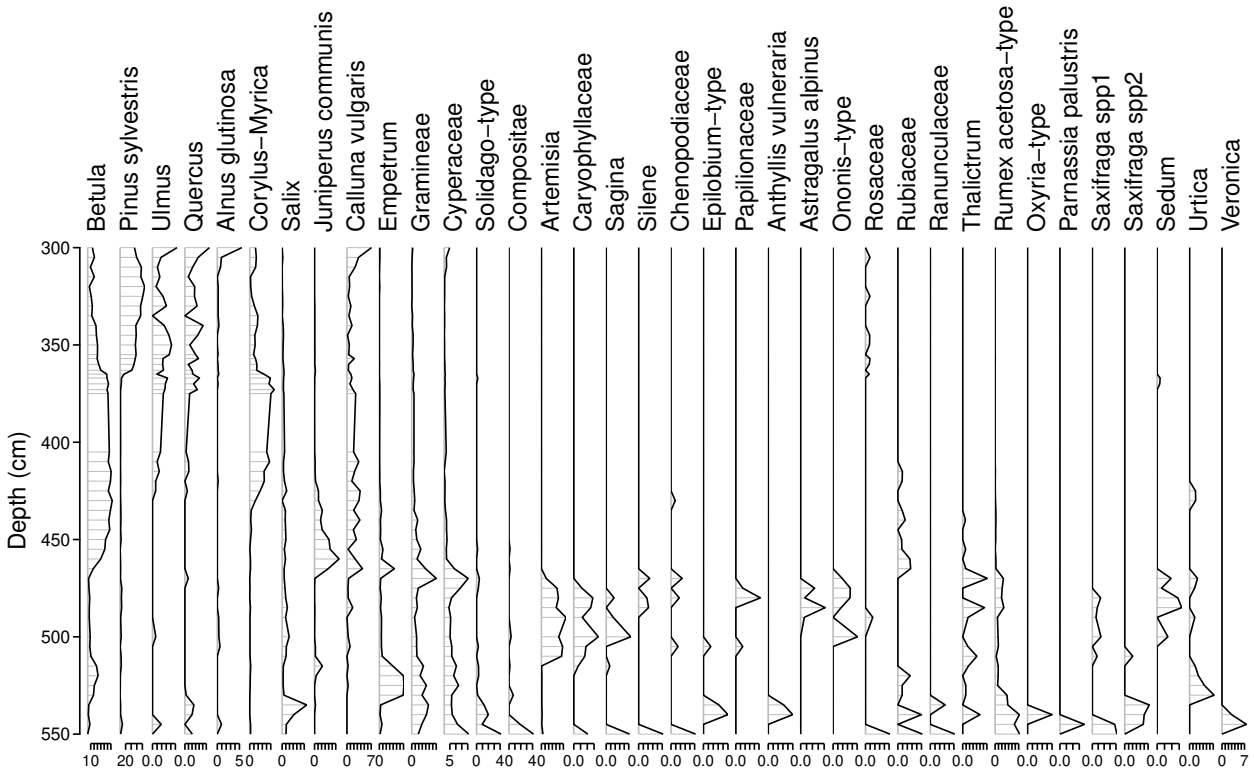
determined automatically depending on the size and length of labels. Arguments to `riojaPlot` that control the appearance of the plot are know as **styles**. Styles can specified on a plot-by-plot basis or can be saved and applied to multiple figures.

Please send questions, bug reports and suggestions for improvements and additions to Stephen.Juggins@ncl.ac.uk.

2. Basic diagram

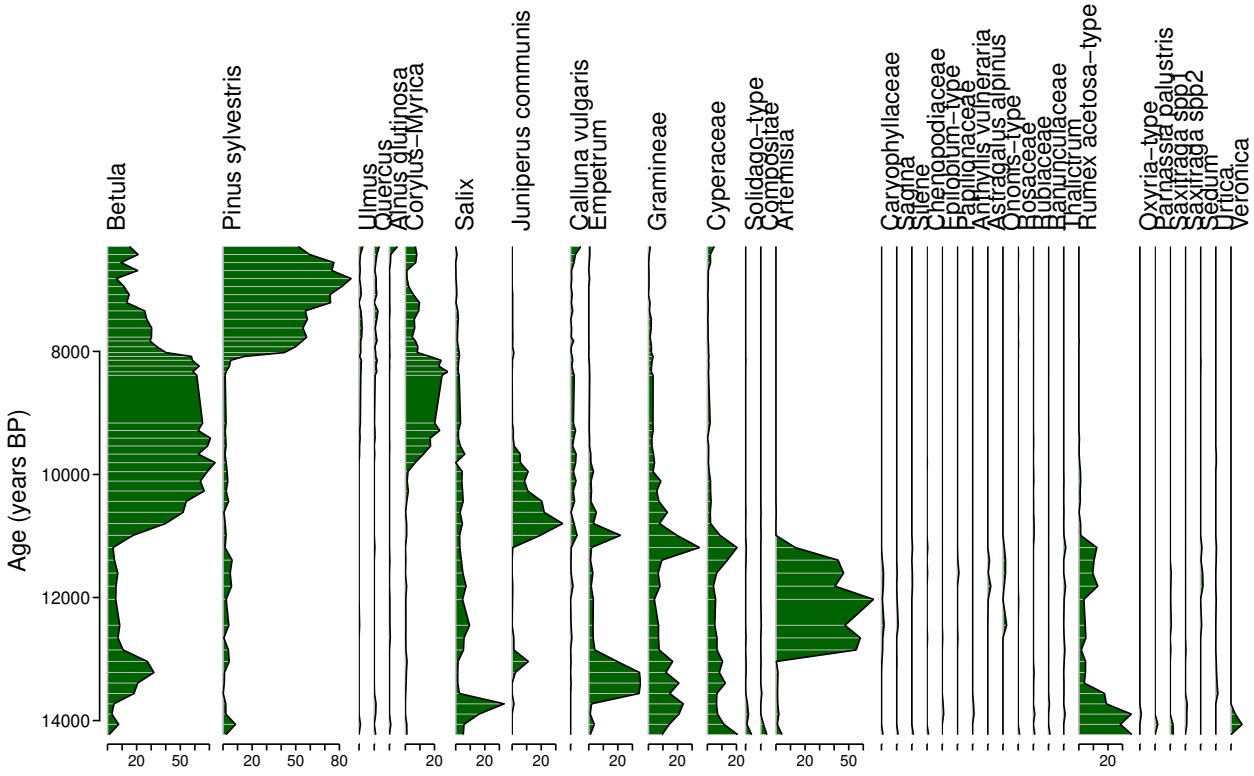
To create a basic stratigraphic diagram just pass `riojaPlot` a data frame of variables to plot and and a data frame with at least one column containing the y-axis variable (usually sample depth or age). If the y-axis variable name is not specified `riojaPlot` will use the first column and label the y-axis with the column name. The default plot is shown with lines and bars for non-percentage data and as filled silhouettes (filled curves) and bars for percentage daat (style `scale.percent=TRUE`. Turn silhouettes off and create plots with lines, bars or symbols by setting styles `plot.poly`, `plot.line`, `plot.bars` or `plot.symb` `TRUE` or `FALSE` (all may be overlain).

```
library(riojaPlot)
library(rioja)

# use built-in data from Abernethy Forest
# see ?rioja::aber for citation
data(aber)
# extract pollen percentages
aber.poll <- aber$spec
# replace species codes with full taxon names
colnames(aber.poll) <- aber$names$Name
aber.chron <- aber$ages

# plot on depth scale (depth is the first column in chron)
riojaPlot(aber.poll, aber.chron)
```



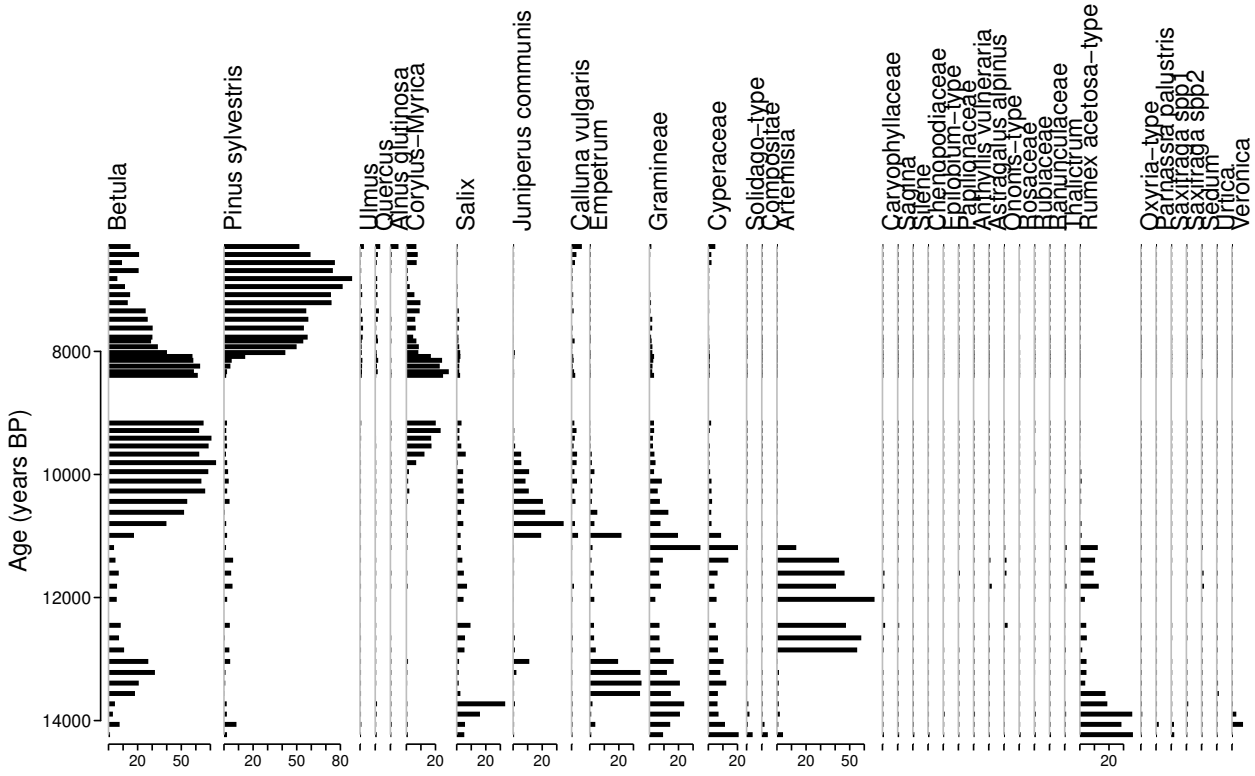
These are percentage data so we scale the diagram for percentages and plot on an age scale by passing the name of the appropriate column to style `yvar.name`. `riojaPlot` will take the y-axis from the first column of `aber.chron` by default.

```
# scale for percentage data
riojaPlot(aber.poll, aber.chron,
   yvar.name="Age (years BP)",
   scale.percent=TRUE)
```



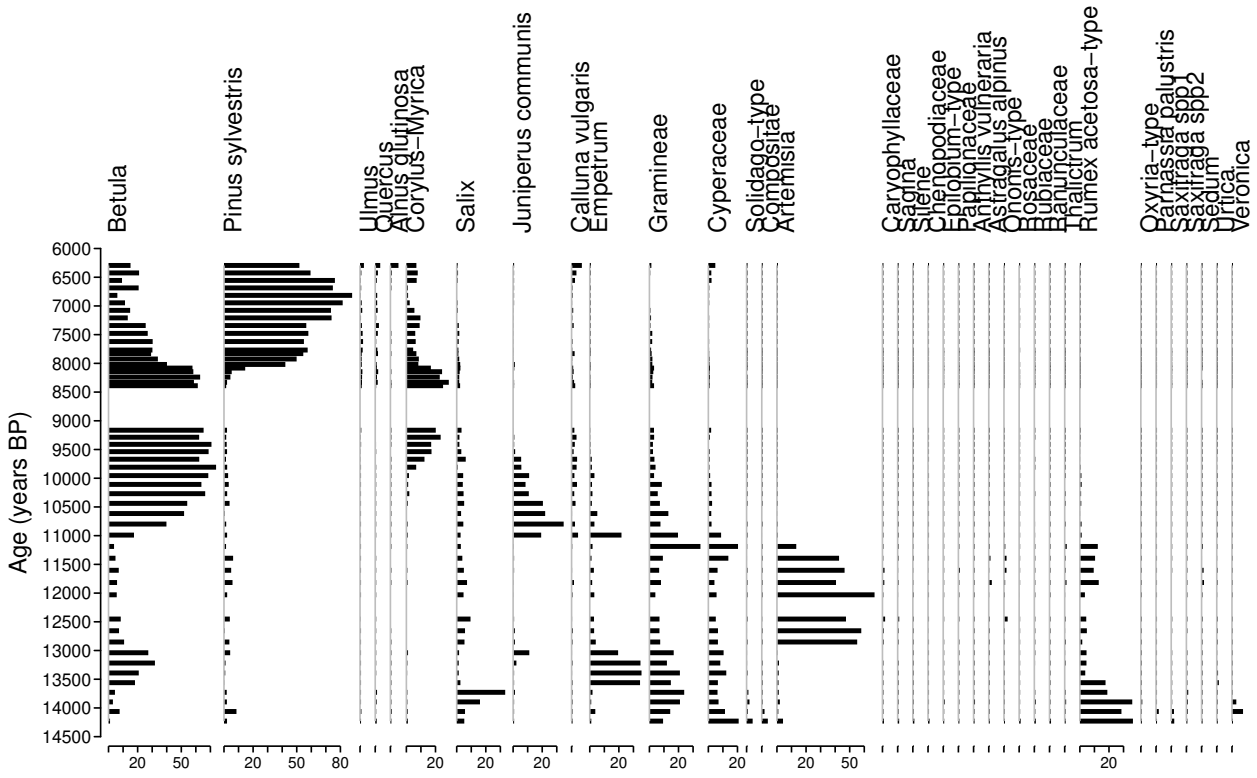
Or plot as bars.

```
# turn of silhouettes and lines, increase bar thickness
riojaPlot(aber.poll, aber.chron,
   yvar.name="Age (years BP)",
   scale.percent=TRUE,
   plot.poly=FALSE,
   plot.line=FALSE,
   lwd.bar=3,
   col.bar="black")
```



The y-axis doesn't cover the whole range so we set the upper and lower limits manually.

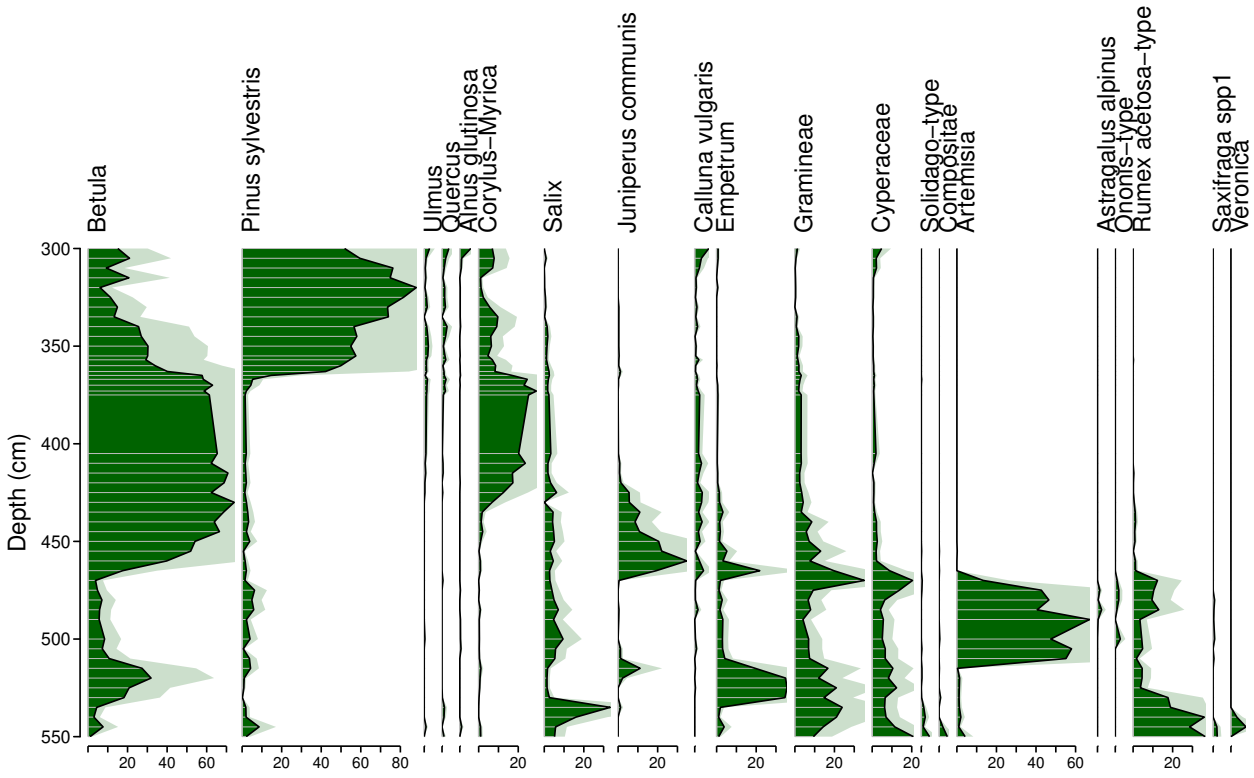
```
# Specify y-axis limits
sapply(aber.chron, range)
#>      Depth (cm) Age (14C years BP) Age (years BP)
#> [1,]        300                5515           6295
#> [2,]        550               12147          14229
riojaPlot(aber.poll, aber.chron,
   yvar.name="Age (years BP)",
   ymin=6000, ymax=14500, yinterval=500,
   scale.percent=TRUE,
   plot.poly=FALSE,
   plot.line=FALSE,
   lwd.bar=3,
   col.bar="black")
```



3. Selecting variables to plot

By default `riojaPlot` will plot all columns in the data frame. In this example there are many rare taxa: we could either remove these from the data frane prior to plotting or pass a character vector of column names to plot to the argument `selVars`. Here we only plot taxa with max abundance greater than 5%. We also add exaggeration curves. These are shown as silhouettes with a multiplication factor of 2 by default by can be changed to lines with different exaggeration using styles `exag.mult`, `col.exag`, and `col.line.exag`. `col.exag="auto" will use a transparent colour the same as the main curve.Use`exag.alpha‘ to change transparency.

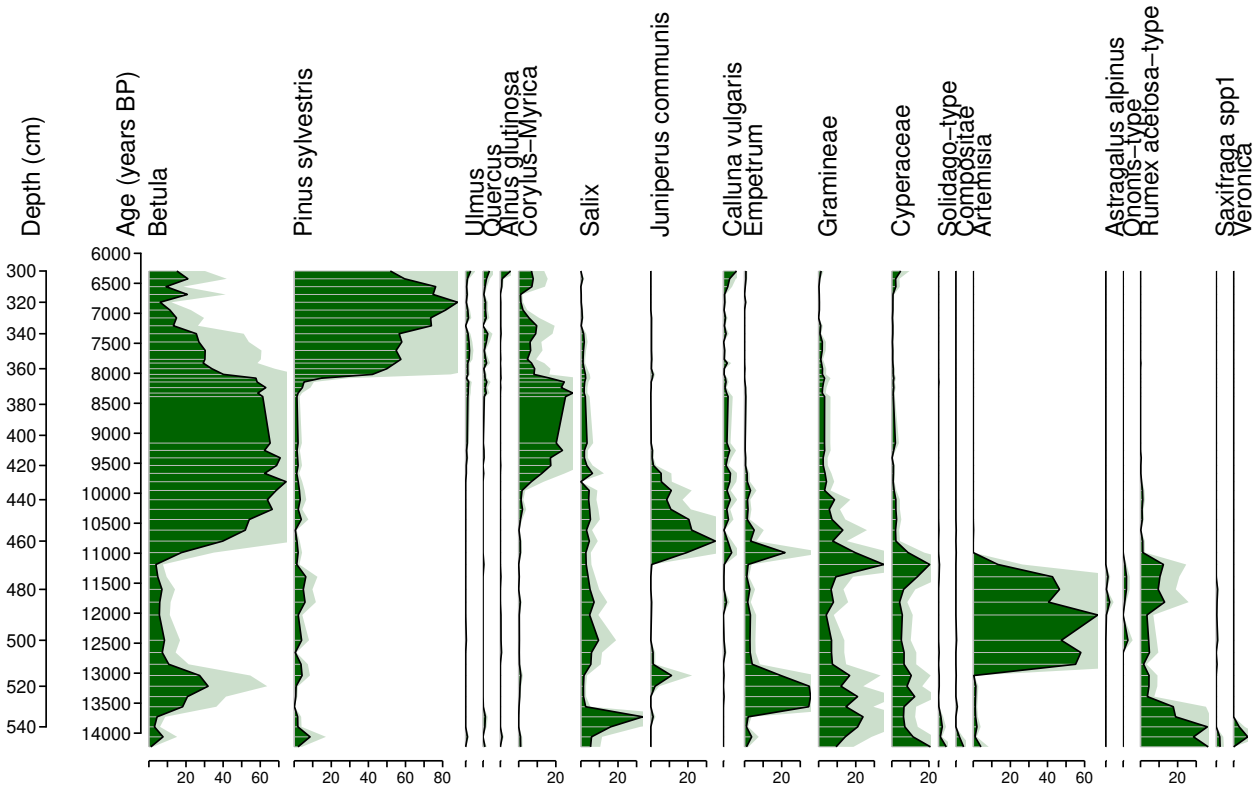
```
# reduce number of taxa and add exaggerations
# make a vector of taxon names with max abundance greater that 2 percent
mx <- apply(aber.poll, 2, max)
aber.sel <- names(mx[mx > 2])

riojaPlot(aber.poll, aber.chron, aber.sel,
   scale.percent=TRUE,
   plot.exag=TRUE)
```



4. Plotting a secondary y-axis

To show a secondary y-axis simply pass the column name of the secondary y-axis variable to style `sec.yvar.name`. In this example we use age (years BP) for the primary axis and depth for the secondary axis and set `plot.sec.axis` to TRUE. We also use `ymin`, `ymax` and `yinterval` to fine-tune the start, end, and increment of the y-axis.

```
riojaPlot(aber.poll, aber.chron, aber.sel,
   yvar.name="Age (years BP)",
   sec.yvar.name="Depth (cm)",
   plot.sec.axis=TRUE,
   ymin=6000,
   ymax=14300,
   yinterval=500,
   scale.percent=TRUE,
   plot.exag=TRUE)
```



5. Showing groups

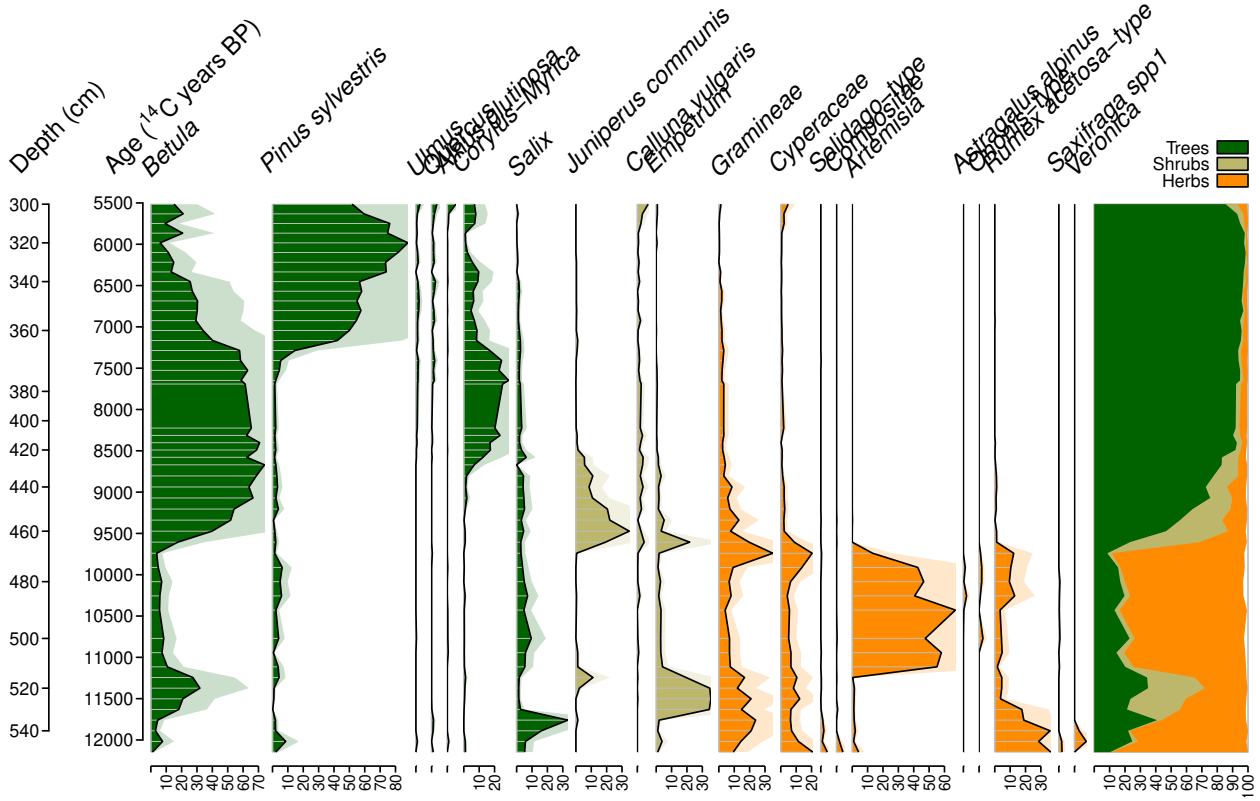
Individual can be assigned to a group and each group assigned a different colour. Cumulative plots can also be added showing the sum of each group.

To assign variables to groups pass a data frame of two columns, the first with the name of each variable (spelt exactly the same as in the data) and a column of group memberships. This should be a factor with the levels determining the order they appear in the legend. The names of the groups will be used in the cumulative plot legend so choose them wisely.

We can also italicise the names using the style `labels.italicise`, rotate them with `srt.xlabel`, and specify a label for the age variable to override the column name. Here we plot a 14C age scale and use an `expression` to create a label with superscriped 14C.

```
# group taxa by type and add cumulative graph
# extract types
aber.types <- aber$names[, -1]
# convert pollen types to a factor
aber.types$Group <- factor(aber.types$Group, levels=c("Trees", "Shrubs", "Herbs"))
ylab <- expression(Age~"("*""^{14}*C~years~BP*")")

riojaPlot(aber.poll, aber.chron, aber.sel, aber.types,
   yvar.name="Age (14C years BP)",
   sec.yvar.name="Depth (cm)",
   ylabel=ylab,
   plot.sec.axis=TRUE,
   ymin=5500,
   ymax=12200,
   yinterval=500,
   scale.percent=TRUE,
```

```
    plot.groups=TRUE,
    plot.cumul=TRUE,
    plot.exag=TRUE,
    labels.italicise = TRUE,
    srt.xlabel=45,
    las.xaxis=2,
    yBottom=0.05)
```

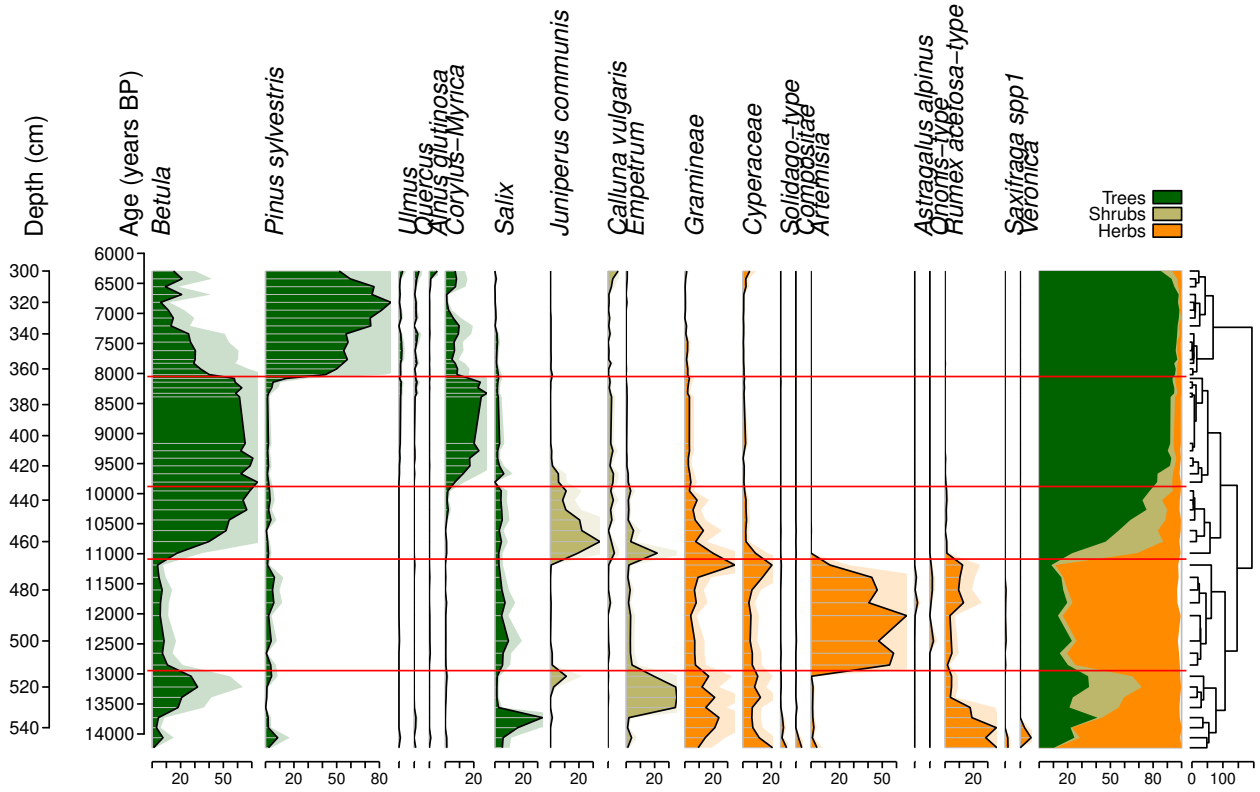


6. Add a zonation

We can add a zonation to the plot in 3 ways. The first is to set the styles `do.clust` and `plot.clust` and / or `plot.zones` to "auto" or the number of zones to show. This will perform a constrained clustering using `chclust` and either add a dendrogram and/or zone lines to the diagram. The number of zones can be estimated using a broken stick model (`plot.zones="auto"` see `?rioja::chclust` for details) or specified (e.g. `plot.zones=3`). Colour of the zone lines can be changed with `col.zones`. The zonation can be based on all columns in the data (style `clust.use.selected=FALSE`) or only the displayed columns if only a subset are selected using `selVars`. Data can optionally be transformed (`clust.data.trans="sqrt"`), scaled (`clust.data.trans="scale"`) prior to clustering or used as is (`clust.data.trans="none"`, the default). Change the width of the dendrogram using style `clust.width=N` where N is the required width as a fraction of the whole diagram width.

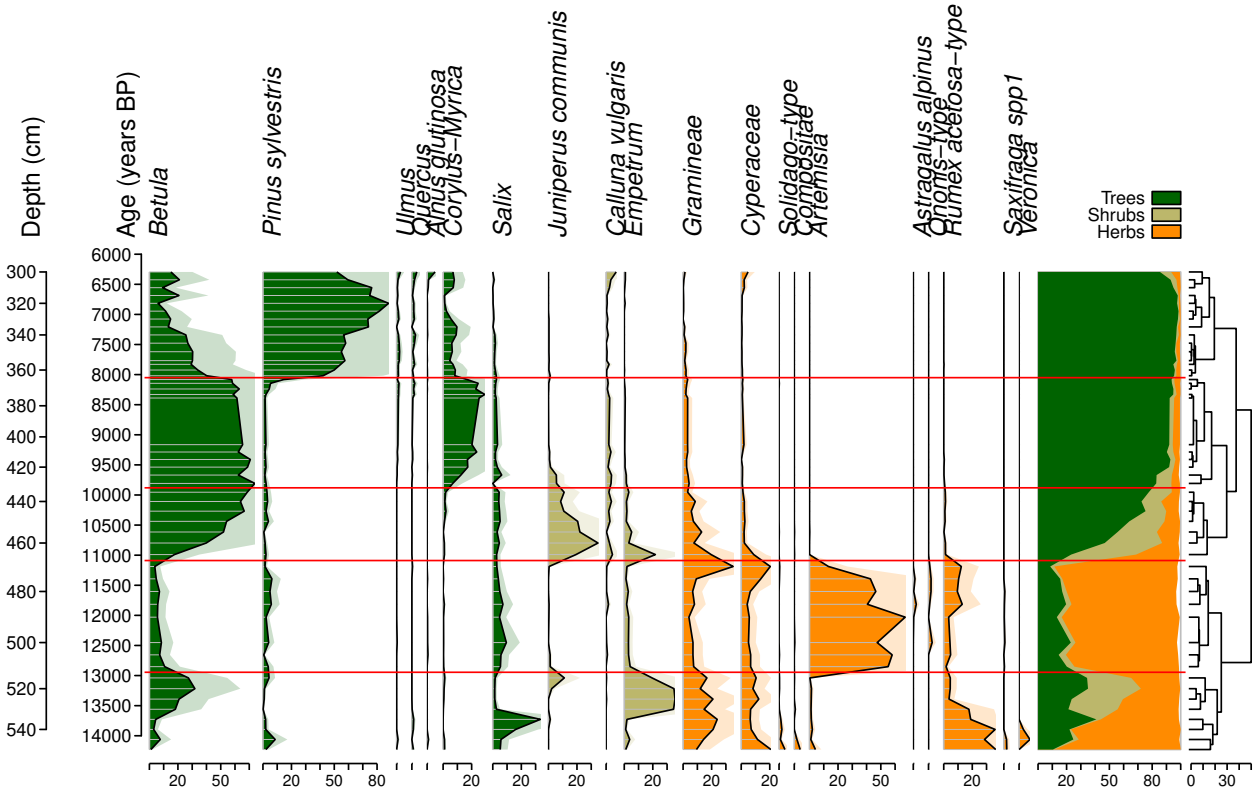
```
riojaPlot(aber.poll, aber.chron, aber.sel, aber.types,
    sec.yvar.name="Depth (cm)",
    yvar.name="Age (years BP)",
    ymin=6000,
    ymax=14300,
    yinterval=500,
    plot.sec.axis=TRUE,
    scale.percent=TRUE,
```

```
   plot.groups=TRUE,
   plot.cumul=TRUE,
   plot.exag=TRUE,
   do.clust=TRUE,
   clust.data.trans="sqrt",
   plot.clust=TRUE,
   plot.zones="auto",
   labels.italicise = TRUE)
```



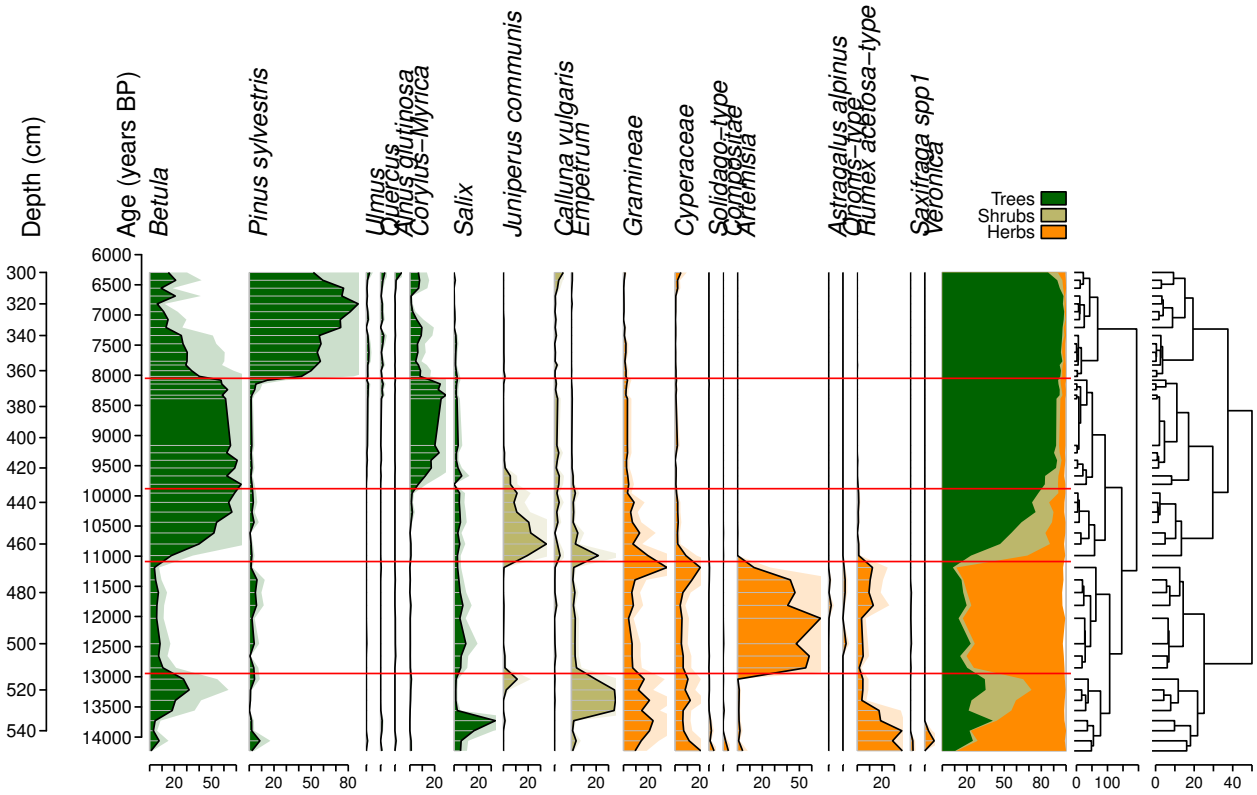
We can also generate a constrained cluster object and pass this to riojaPlot for plotting. Here we log10-transform the data prior to clustering.

```
aber.clust <- chclust(dist(log10(aber.poll+1)))
riojaPlot(aber.poll, aber.chron, aber.sel, aber.types, clust=aber.clust,
   sec.yvar.name="Depth (cm)",
   yvar.name="Age (years BP)",
   ymin=6000,
   ymax=14300,
   yinterval=500,
   plot.sec.axis=TRUE,
   scale.percent=TRUE,
   plot.groups=TRUE,
   plot.cumul=TRUE,
   plot.exag=TRUE,
   clust.data.trans="sqrt",
   plot.clust=TRUE,
   plot.zones="auto",
   labels.italicise = TRUE)
```



Finally, we can add a zonation at a different position in the diagram. Here we add the zonation based on log10-transformed data to the original diagram with a sqrt-transformed zonation for comparison. In this case we set the right-hand limit of the original diagram with `xRight` and add the additional zonation to this space. This is a bit of a contrived example but adding the zonation separately to the diagram can be useful if we are plotting multiple datasets and want the zonation on the far-right (see example in section 8).

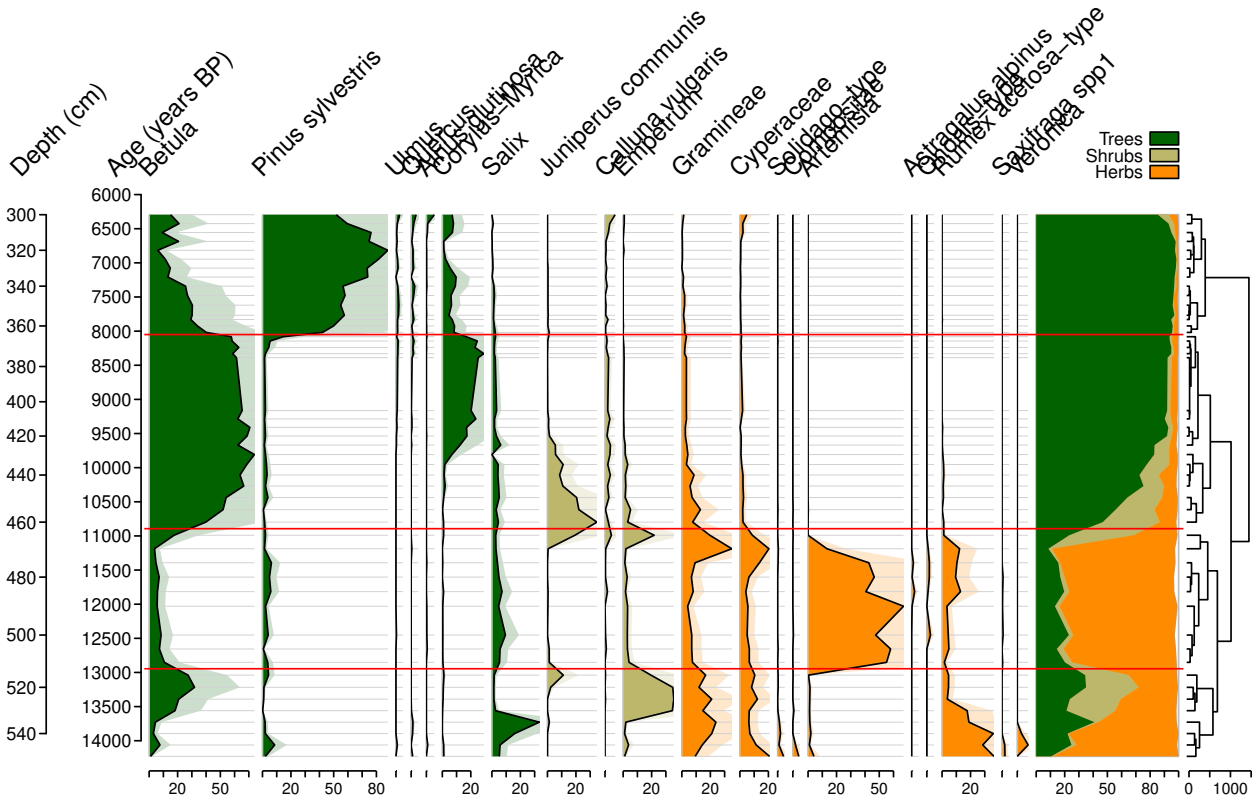
```
aber.clust <- chclust(dist(log10(aber.poll+1)))
rp <- riojaPlot(aber.poll, aber.chron, aber.sel, aber.types,
   sec.yvar.name="Depth (cm)",
   yvar.name="Age (years BP)",
   ymin=6000,
   ymax=14300,
   yinterval=500,
   plot.sec.axis=TRUE,
   scale.percent=TRUE,
   plot.groups=TRUE,
   plot.cumul=TRUE,
   plot.exag=TRUE,
   clust.data.trans="sqrt",
   do.clust=TRUE,
   plot.clust=TRUE,
   plot.zones="auto",
   labels.italicise = TRUE,
   xRight = 0.9)

addRPClust(rp, aber.clust, xLeft=0.91, xRight=0.99)
```



7. Reusing styles

Save settings as a style and apply to figure. Here we also full-width plot bars behind the curves.

```
mystyle <- makeStyles(
   plot.sec.axis=TRUE,
   scale.percent=TRUE,
   plot.groups=TRUE,
   plot.cumul=TRUE,
   plot.exag=TRUE,
   plot.poly=TRUE,
   plot.bar="full",
   bar.back=TRUE,
   lwd.bar=0.5,
   col.bar="lightgrey",
   do.clust=TRUE,
   plot.clust=TRUE,
   plot.zones="auto",
   srt.xlabel=45)

riojaPlot(aber.poll, aber.chron, aber.sel, aber.types, mystyle,
   sec.yvar.name="Depth (cm)",
   yvar.name="Age (years BP)",
   ymin=6000,
   ymax=14300,
   yinterval=500)
```



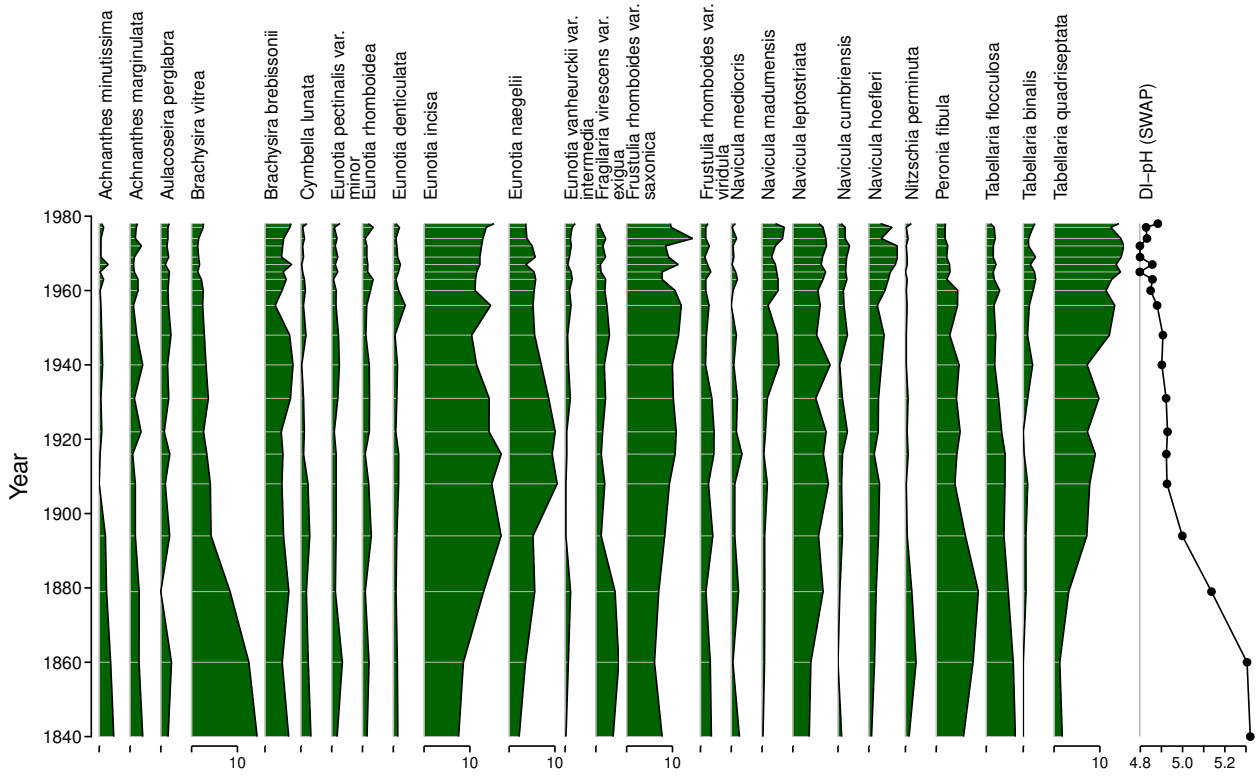
8. Combining different datasets in a single diagram

We may want to plot different datasets in the same figure. For example, we may want to plot one set of variables with percentage scaling and the other with normal scaling where each variable is the same width. Or we may have two or more datasets measured at different depths of ages.

To combine different datasets we plot the first with function `riojaPlot` and the others with function `riojaPlot2`. We also need to specify the right-hand position of each plot in fractions of the page width.

```
# Diatom data the Round Loch of Glenhead, Galloway, SW Scotland
# Shows recent acidification and small recovery
data(RLGH)
RLGH.names <- RLGH$names
RLGH.diat <- RLGH$spec

# plot only common taxa
mx <- apply(RLGH.diat, 2, max)
RLGH.sel <- colnames(RLGH.diat)[mx > 2]
RLGH.chron <- RLGH$depths
# data has 210Pb age based on years before coring (1980)
# Add new column with Years (CE)
RLGH.chron$Year <- 1980 - RLGH.chron$Age

# do a pH reconstruction using SWAP modern dataset (see ?rioja::SWAP)
data(SWAP)
# generate a WA model
SWAP.wa <- WA(SWAP$spec, SWAP$pH)
RLGH.pH <- predict(SWAP.wa, RLGH$spec)
# convert to data frame with single column
```

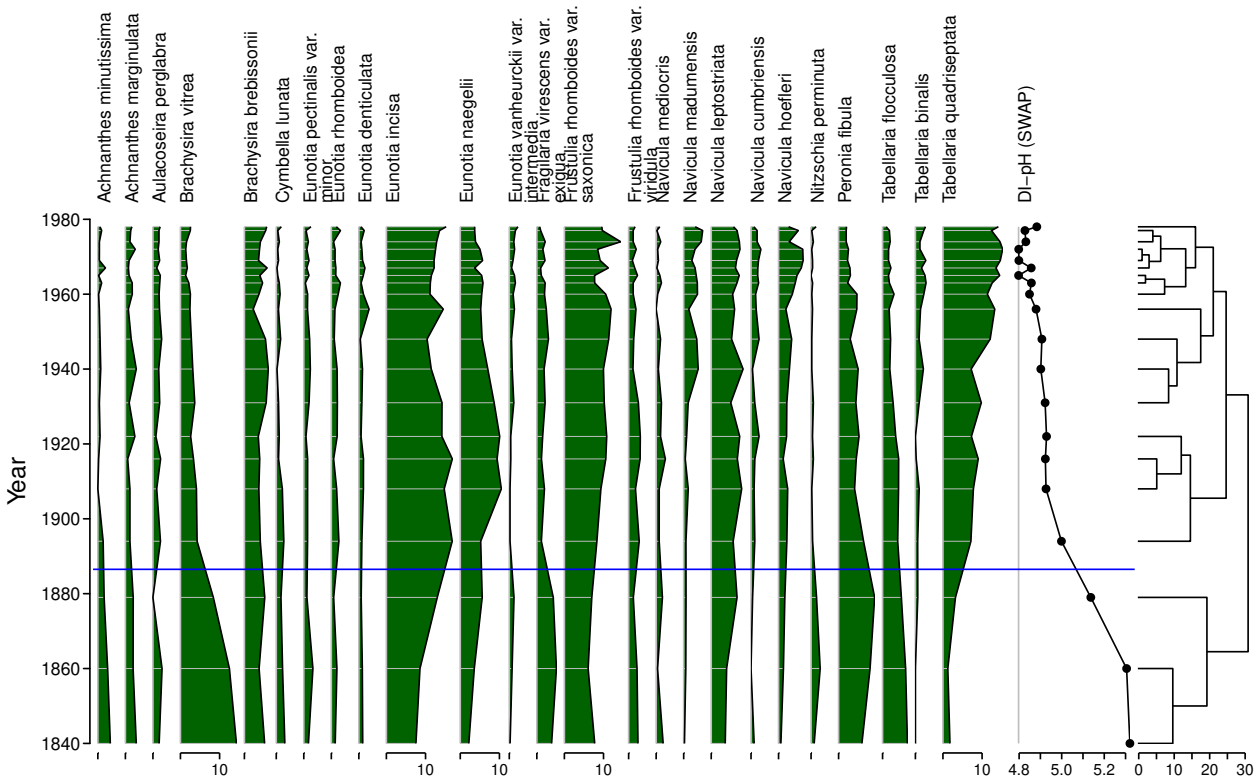
```
RLGH.pH <- data.frame(`DI-pH (SWAP)`=RLGH.pH$fit[, 1], check.names=FALSE)
rp <- riojaPlot(RLGH.diat, RLGH.chron,
          yvar.name="Year",
          scale.percent=TRUE,
          y.rev=FALSE,
          selVars=RLGH.sel,
          ymax=1980,
          xlabels=RLGH$names$TaxonName,
          cex.xlabel=0.7,
          labels.break.n=25,
          xRight=0.9)

riojaPlot(RLGH.pH, RLGH.chron[, "Year", drop=FALSE],
           riojaPlot=rp,
           scale.minmax=FALSE,
           plot.bar=FALSE,
           plot.symb=TRUE,
           symb.cex=0.6)
```



Here we add a zonation to the right of the above figure. There are only two significant zones in these data.

```
RLGH.clust <- chclust(dist(sqrt(RLGH.diat)))
rp <- riojaPlot(RLGH.diat, RLGH.chron,
          yvar.name="Year",
          scale.percent=TRUE,
          y.rev=FALSE,
          selVars=RLGH.sel,
          ymax=1980,
          xlabels=RLGH$names$TaxonName,
          cex.xlabel=0.7,
```

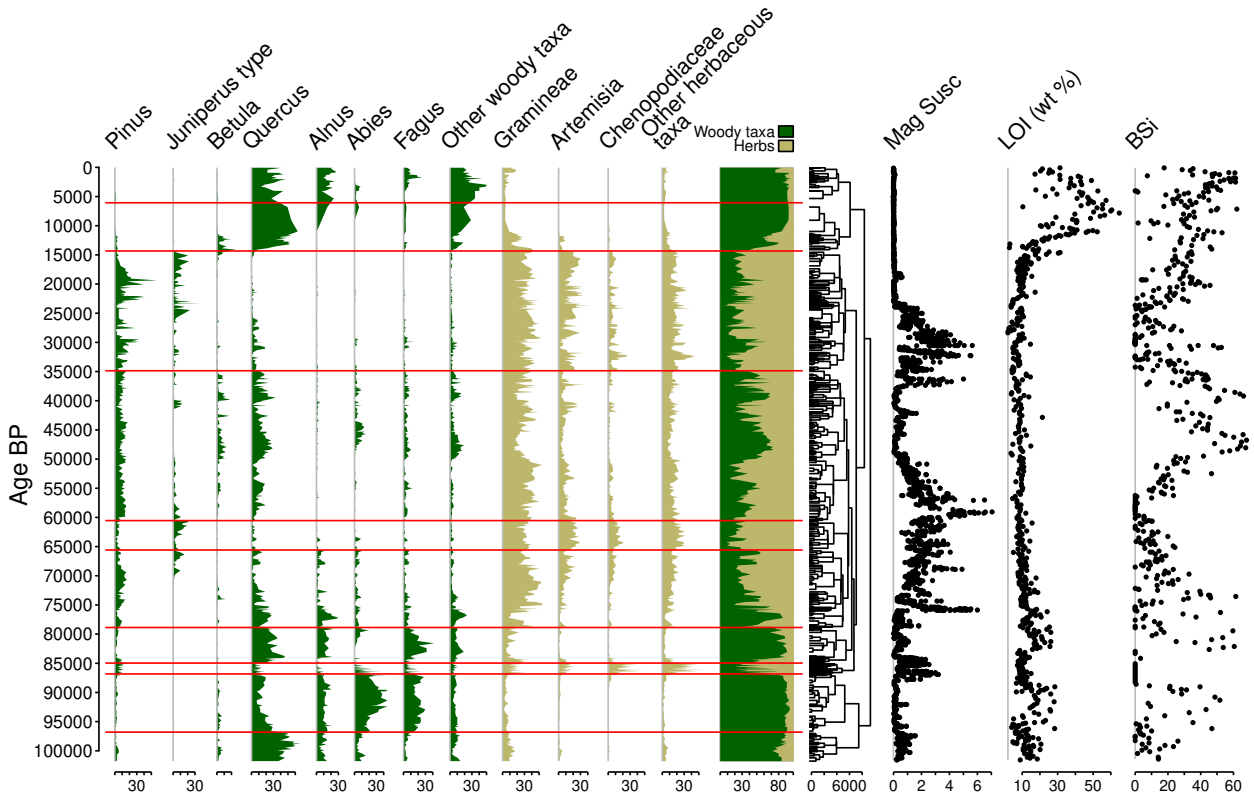
```
          labels.break.n=25,
          xRight=0.8)

rp2 <- riojaPlot(RLGH.pH, RLGH.chron[, "Year", drop=FALSE],
           riojaPlot=rp,
           xRight=0.9,
           scale.minmax=FALSE,
           plot.bar=FALSE,
           plot.symb=TRUE,
           symb.cex=0.6)

rp3 <- addRPClustZone(rp2, RLGH.clust, col="blue")
addRPClust(rp3, RLGH.clust)
```



In the next example we combine pollen, magnetic susceptibility, LOI and biogenic silica data recorded in a core from Lago Grande di Monticchio and published by Allen et al. (1999). The core spans the last c. 100 kyr and the four datasets are measured at different depth intervals and resolutions. The data are downloaded from the NOAA paleoclimatology data archive (https://www.ncei.noaa.gov/products/paleoclimatology).

```
library(readxl)
library(dplyr)
# Import the data, assign pollen types to groups
fpath <- system.file("extdata/allen1999.xlsx", package="riojaPlot")
pollen <- read_excel(fpath, sheet="Pollen data", skip=2)
pollen.chron <- pollen %>% select(1)
pollen <- pollen %>% select(Pinus:`Other herbaceous taxa`)
types <- data.frame(Name=colnames(pollen), Group="Woody taxa")
types$Group[9:12] <- "Herbs"
types$Group <- factor(types$Group, levels=c("Woody taxa", "Herbs"))
mag <- read_excel(fpath, sheet="Magnetic susceptibility", skip=2)
```

```
mag.chron <- mag %>% select(1)
mag <- mag %>% select(`Mag Susc`) %>% mutate(`Mag Susc` = `Mag Susc` / 1000)
loi <- read_excel(fpath, sheet="Loss on ignition", skip=2)
loi.chron <- loi %>% select(1)
loi <- loi %>% select("LOI (wt %)")
BSi <- read_excel(fpath, sheet="Biogenic silica", skip=2)
BSi.chron <- BSi %>% select(1)
BSi <- BSi %>% select("BSi")

rp1 <- riojaPlot(pollen, pollen.chron, groups=types,
          yinterval = 5000,
          ymin = 0,
          ymax=102000,
          yvar.name = "Age BP",
          scale.percent=TRUE,
          plot.groups=TRUE,
          do.clust = TRUE,
          plot.zones = "auto",
          plot.clust=TRUE,
          plot.cumul=TRUE,
          cex.cumul=0.6,
          srt.xlabel=45,
          xSpace = 0.01,
          plot.bar=FALSE,
          tcl=-0.1,
          cex.yaxis=0.7,
          cex.xlabel=0.8,
          xRight = 0.7,
          plot.line=FALSE
          )

rp2 <- riojaPlot(mag, mag.chron[, "Age BP", drop=FALSE],
           riojaPlot=rp1, xGap = 0.01,
           xRight=0.8, scale.minmax=FALSE,
           plot.bar=FALSE, plot.line=F,
           plot.symb=TRUE, symb.cex=0.3)
rp3 <- riojaPlot(loi, loi.chron[, "Age BP", drop=FALSE],
           riojaPlot=rp2,
           xRight=0.9,
           scale.minmax=FALSE, plot.bar=F,
          plot.line=F, plot.symb=TRUE, symb.cex=0.3)
riojaPlot(BSi, BSi.chron[, "Age BP", drop=FALSE],
           riojaPlot=rp3,
           xRight=0.99,
           scale.minmax=FALSE, plot.bar=FALSE,
           plot.line=FALSE, plot.symb=TRUE, symb.cex=0.3)
```

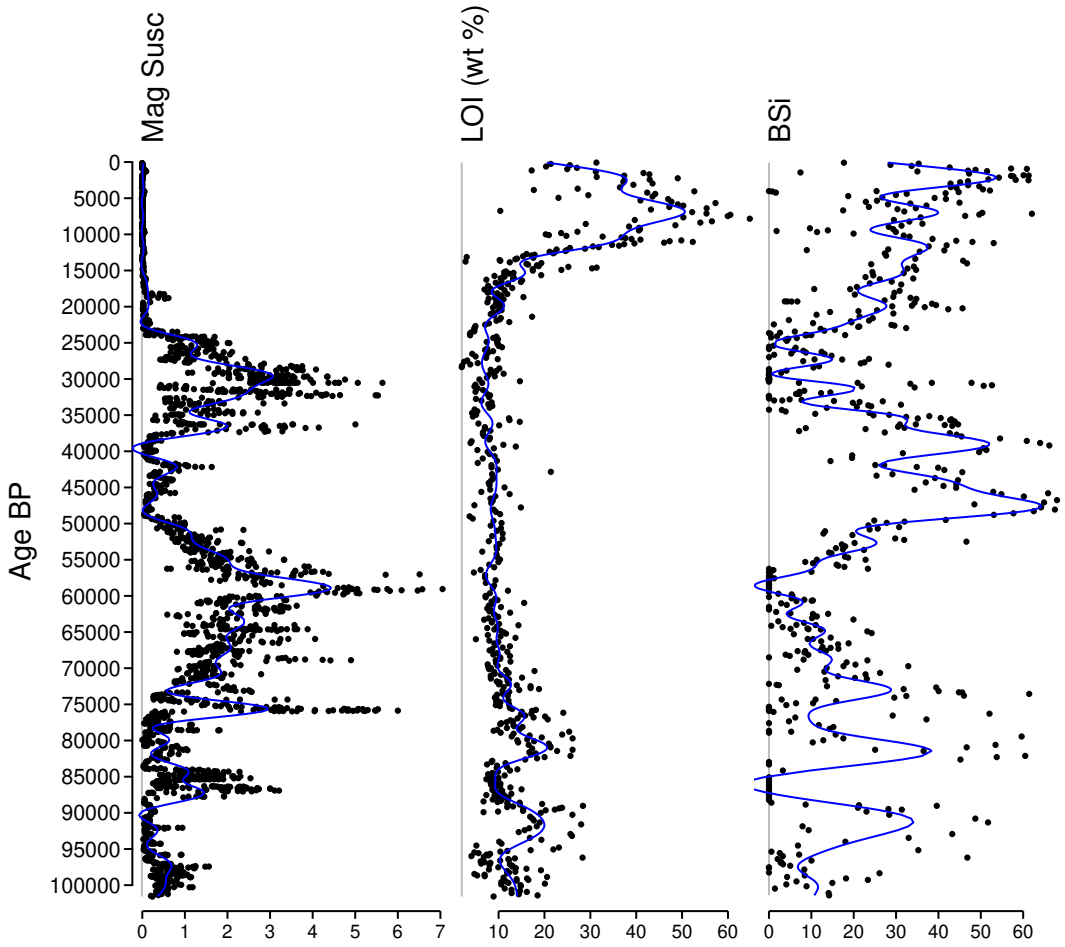


9. Adding custom plotting functions

In the above example the physical / chemical variables are measured on a large number of samples and we plot with symbols because these display the variability in the data more clearly than lines. It would be useful to fit a smooth through these data to highlight the trends. Here we plot the three phys/chem variables and fit a gam to each curve using a custom function to add the smooth to each plot.

```
fun.gam <- function(x, y, i, nm, style) {
  tmp <- data.frame(x=y, y=x)
  gam <- mgcv::gam(y ~ s(x, k=50), data=tmp)
  x2 <- predict(gam, type="response")
  lines(x2, y, col="blue", lwd=1)
}

rp <- riojaPlot(mag, mag.chron[, "Age BP", drop=FALSE],
          yinterval = 5000,
          ymin = 0,
          ymax=102000,
          xRight=0.3,
          scale.minmax=FALSE,
          plot.bar=FALSE,
          plot.line=F,
          plot.symb=TRUE,
          plot.poly=FALSE,
          symb.cex=0.3,
          fun.xfront=fun.gam)

rp1 <- riojaPlot(loi, loi.chron[, "Age BP", drop=FALSE],
           riojaPlot=rp,
```

```
            xRight=0.5,
            scale.minmax=FALSE, plot.bar=F, plot.line=F, plot.symb=TRUE,
            symb.cex=0.3, fun.xfront=fun.gam)
riojaPlot(BSi, BSi.chron[, "Age BP", drop=FALSE],
            riojaPlot=rp1,
            xRight=0.7,
            scale.minmax=FALSE, plot.bar=FALSE, plot.line=FALSE, plot.symb=TRUE,
            symb.cex=0.3, fun.xfront=fun.gam)
```

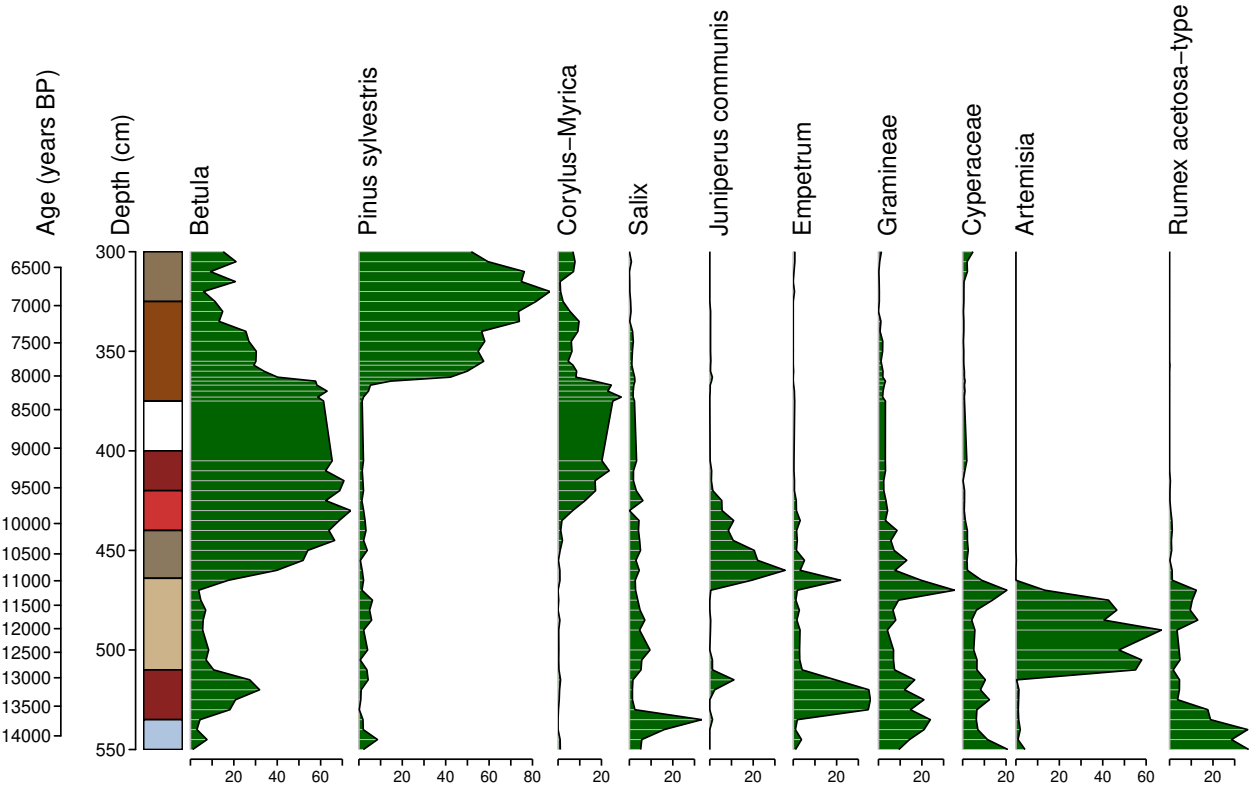


10. Adding a column showing lithology

Adding a column showing lithogy is experimental at the moment but it is possible to show basic lithology with different colours and / or pattern fills. To do this create a data frame with at least 3 columns - the top and bottom of each lithological unit, and the fill colour for that unit. You also have a define a custom function that takes this data frame and draws the lithology. Here we create a function that uses apply to loop through each row in the lithology dataframe and plot a filled rectangle in the lithology column.

```
library(purrr)
lithology <- data.frame(
  top=   c(300, 325, 375, 400, 420, 440, 464, 510, 535),
  bottom=c(325, 375, 400, 420, 440, 464, 510, 535, 550),
  lithology=c("Peat", "Mud", "NS", "Mud", "Mud2", "SiltyMud", "SiltyMud2", "Mud3", "Silt"),
  colour=c("burlywood4", "saddlebrown", "NA", "brown4", "brown3", "navajowhite4",
           "navajowhite3", "brown4", "lightsteelblue")
)
```

```
myfun2 <- function(x, style) {
  x %>% pwalk(~rect(0, ..1, 1, ..2, col=..4))
}

mx <- apply(aber.poll, 2, max)
mx5 <- mx[mx>10]

rp <- riojaPlot(aber.poll, aber.chron, selVars=names(mx5), lithology=lithology,
          sec.yvar.name="Age (years BP)",
          sec.ymin=6000, sec.ymax=14000, sec.yinterval=500,
          yvar.name="Depth (cm)",
          plot.sec.axis = TRUE,
          scale.percent=TRUE,
          fun.lithology=myfun2)
```



11. Frequently asked questions

1. How do I change the order of the variables in the diagram?

There are several ways to do this. Variables are plotted in the order they appear in the data, so rearrange the order of variables in the original data. `dplyr::select` offers a convenient way to do this:

```
data.reordered <- data %>% select(var3, var1, var5) # and so on...
```

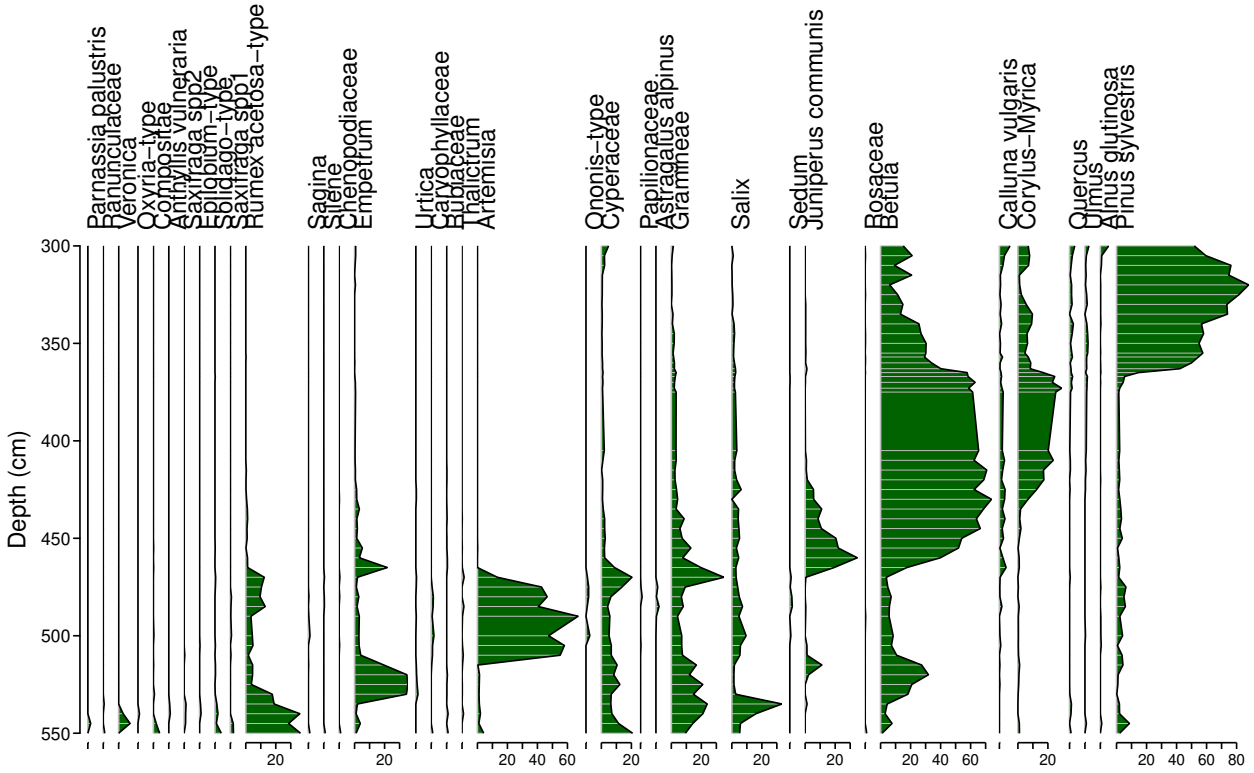
Another way is to create a character vector wi the variables in the required order and pass this to `selVars`:

```
myorder <- c("var3", "var1", "var5") # and so on...
riojaPlot(spec, chron, selVArs=myorder)
```

A quick way to sort variables accorging to their position in the diagram is to set the style

`wa.order="bottomleft"` (or use `bottomright` for the opposite). This will sort variables from those common at the base of the core on the left to those common at the top on the right.

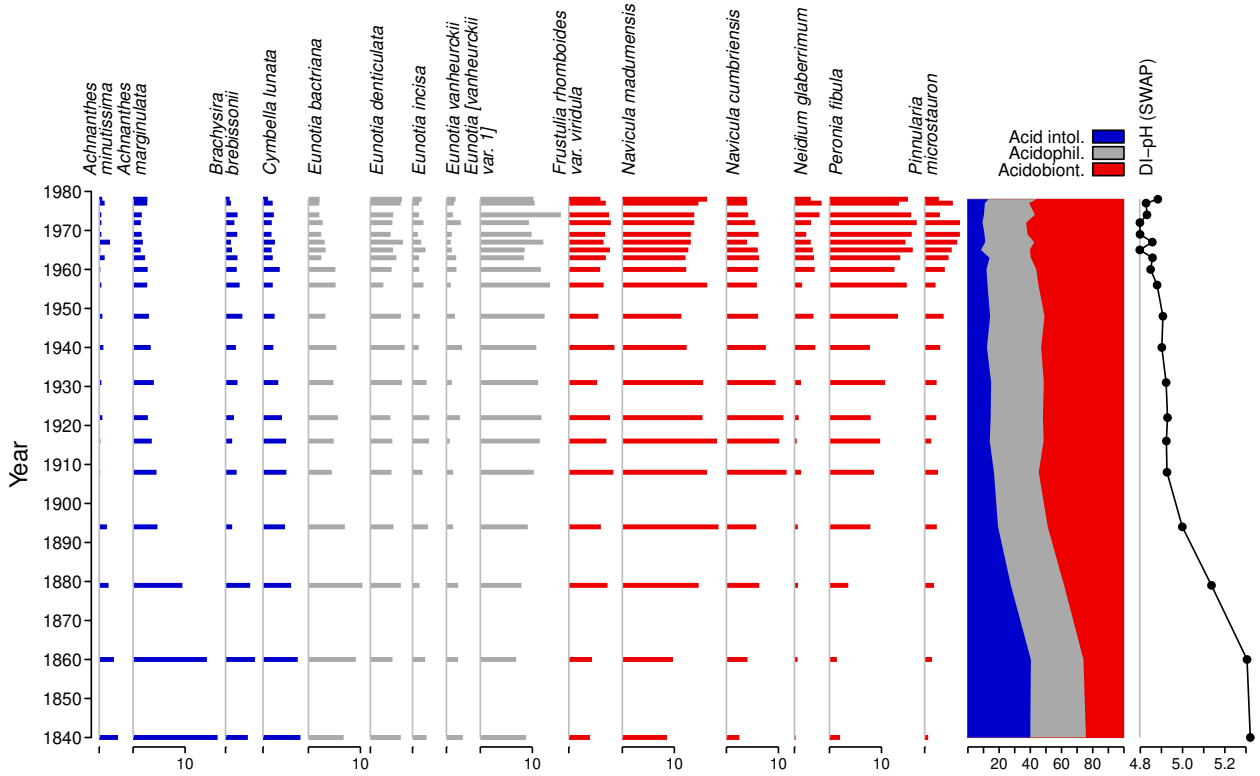
```
riojaPlot(aber.poll, aber.chron,
          scale.percent=TRUE,
          wa.order="bottomleft")
```



To have complete control you probably need to use option 1 - change the order in the original data. Here is an extended example in which we sort the taxa in the RLGH diatom data according to their pH optima. We also group the taxa into three pH classes based on their pH optima and display the groups and cumulative plot.

```
# these data only contain 41 most common taxa, so recalculate abundances to sum to 100 percent
RLGH.diat <- RLGH$spec
RLGH.diat <- vegan::decostand(RLGH.diat, method="total") * 100

optima <- SWAP.wa$coefficients %>%
  data.frame() %>%
  tibble::rownames_to_column(var="CODE")

optima2 <- data.frame(CODE=colnames(RLGH.diat)) %>%
  dplyr::left_join(optima, by="CODE") %>%
  mutate(Group=case_when(Optima <= 5.1 ~ "Acidobiont.",
                         Optima > 5.1 & Optima < 5.4 ~ "Acidophil.",
                         TRUE ~ "Acid intol."),
         Group=factor(Group, levels=c("Acid intol.", "Acidophil.", "Acidobiont."))) %>%
  mutate(Optima=ifelse(is.na(Optima), 5.5, Optima)) %>%
  arrange(desc(Optima)) %>%
  select(CODE, Group)

RLGH.diat <- RLGH.diat %>% select(optima2$CODE)
```

```
mx <- apply(RLGH.diat, 2, max)
RLGH.sel <- colnames(RLGH.diat)[mx > 3]

rp <- riojaPlot(RLGH.diat, RLGH.chron, groups=optima2,
          yvar.name="Year",
          scale.percent=TRUE,
          y.rev=FALSE,
          selVars=RLGH.sel,
          ymax=1980,
          yinterval=10,
          xlabels=RLGH$names$TaxonName,
          labels.italicise=TRUE,
          cex.xlabel=0.7,
          plot.poly=FALSE,
          plot.line=FALSE,
          lwd.bar=3,
          xRight=0.9,
          plot.groups=TRUE,
          plot.cumul=TRUE,
          col.group=c("mediumblue", "darkgrey", "red2"),
          cumul.mult=0.3
          )

riojaPlot(RLGH.pH, RLGH.chron[, "Year", drop=FALSE],
           riojaPlot=rp,
           scale.minmax=FALSE,
           plot.bar=FALSE,
           plot.symb=TRUE,
           symb.cex=0.6)
```



2. How do I change the font sizes?

Use the following styles (note, font sizes are specified relative to the default size of 1): - `cex.xaxis` = font size of x-axis values (default 0.6) - `cex.yaxis` = font size of y-axis values (default 0.7) - `cex.xlabel` = font size of x-axis labels (default 0.9) - `cex.ylabel` = font size of y-axis labels (default 0.9) - `cex.cumul` = font size of group names in cumulative plot (default 0.7)

3. How do I change the limits and labels of the of the y-axes?

Use the style `y.rev` to reverse the scale of y-axis (default is TRUE, ie. values increase from top to bottom). Use the styles `ymin`, `ymax` and `yinterval` to change the start, end and interval values. The corresponding styles `sec.ymin`, `sec.ymax` and `sec.yinterval` control the secondary y-axis scale.

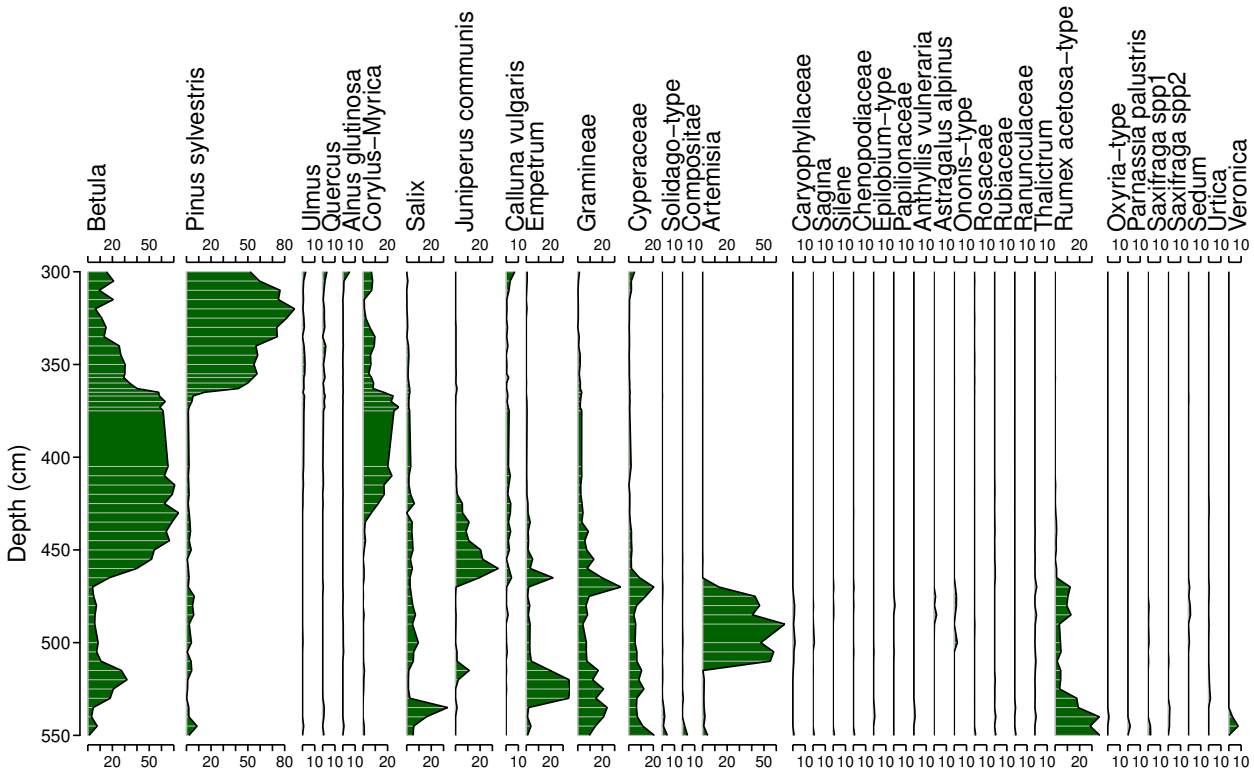
4. My variable names are too long and take up too much space at the top of the diagram.

Long names can be split by setting style `labels.break.long=TRUE` (the default). Names will be split so no part if greater than `labels.break.n` characters (default 20).

5. How do I add a second x-axis at the top of the plots?

Styles `plot.top.axis` and `plot.bottom.axis` allow plotting of the top and bottom x-axes. If there is not enough space between the top axis and the taxon names use style `ylabPos` to fine-tune the gap (default 0.1).

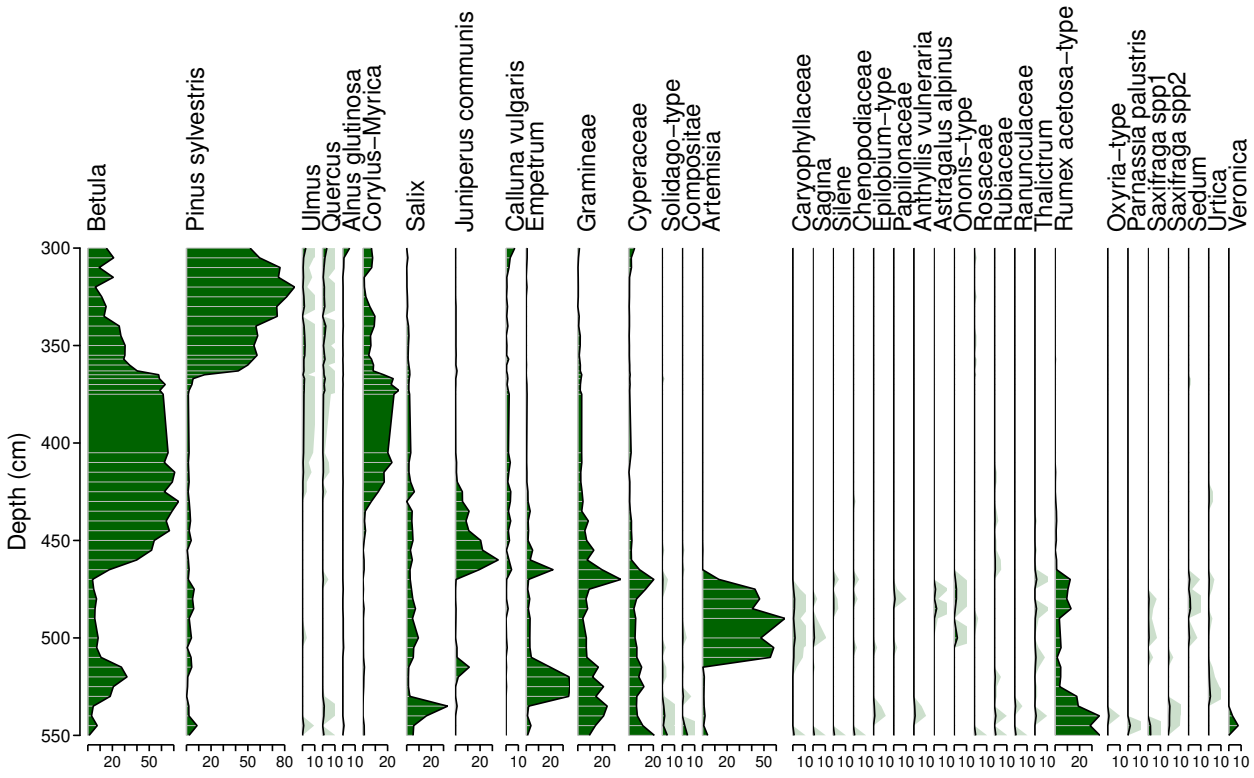
```
data(aber)
aber.poll <- aber$spec
colnames(aber.poll) <- aber$names$Name
aber.chron <- aber$ages

riojaPlot(aber.poll, aber.chron,
   scale.percent=TRUE,
   min.width.pc=10,
   plot.top.axis=TRUE)
```



6. How do I add exaggeration curves to selected taxa only?

Style `plot.exag` takes a single value (TRUE or FALSE) to add exaggerations to all curves but can also take a logical vector of length equal to the number of columns in the x-variables data frame to control exaggerations for each variable. The following example shows how to add exaggerations for pollen types with maximum relative abundance < 5 percent. We also change the exaggeration multiplier to 10 and set the min widths of curves to 10% (default is 5%).

```
aber.poll <- aber$spec
colnames(aber.poll) <- aber$names$Name
aber.chron <- aber$ages
mx <- apply(aber.poll, 2, max)
exag.sel <- mx < 5

riojaPlot(aber.poll, aber.chron,
   scale.percent=TRUE,
   plot.exag=exag.sel,
   exag.mult=10,
   min.width.pc=10)
```

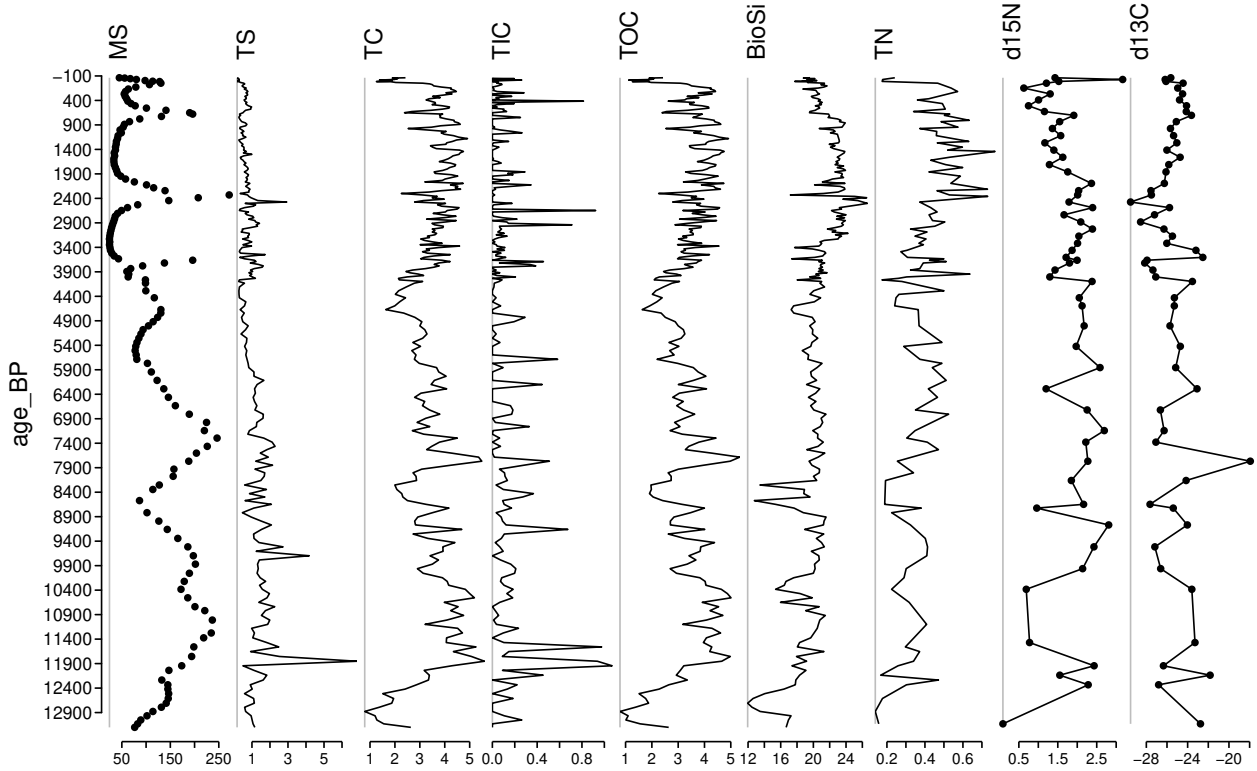


7. How I plot different styles for different variables?

In this example we use geochemical data from Laguna del Maule published by Frugone-Álvarez et al. (2020) and downloaded from the NOAA Paleoclimate data archive (https://www.ncei.noaa.gov/products/paleoclimatology). The data we plot here contain 9 geochemical variables. Some (TS, TC, TIC and TOC) are measured at every interval but others only at selected depths. We plot magnetic susceptibility with symbols, TS, TC, TIC, TOC and BioSi with lines, and d15N and d13C with lines and symbols.

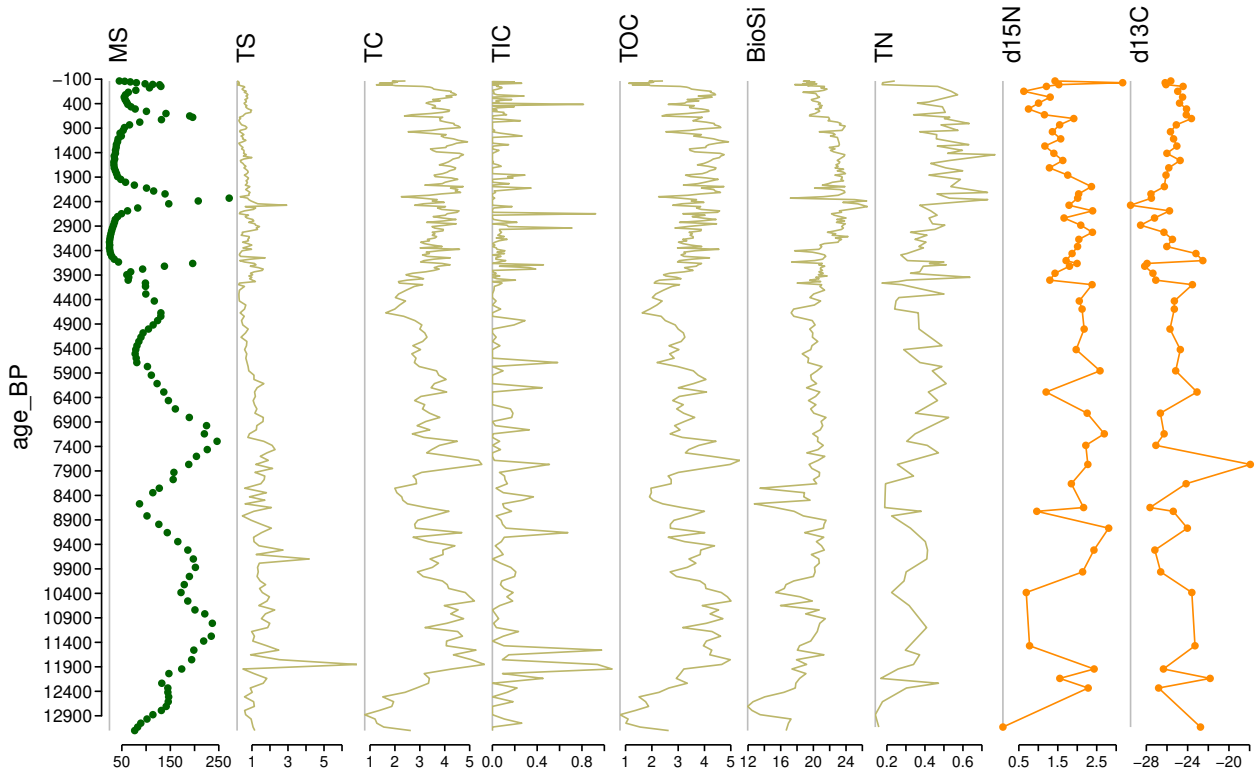
```
fpath <- system.file("extdata/maule2020geochem.txt", package="riojaPlot")
maule <- readr::read_delim(fpath, skip=158, show_col_types = FALSE)

maule.data <- maule %>% select(-(1:4))
maule.chron <- maule %>% select(1:4)

# How many measurements do we have for each variable:
unlist(lapply(maule.data, function(x) sum(!is.na(x))))
#>    MS    TS    TC   TIC   TOC BioSi    TN  d15N  d13C
#>   146   267   267   267   267   253   108    57    57

selsymb <- rep(FALSE, 9)
selsymb[c(1, 8:9)] <- TRUE
selbar <- rep(FALSE, 9)
selline <- rep(TRUE, 9)
selline[1] <- FALSE

riojaPlot(maule.data, maule.chron,
          yvar.name="age_BP",
          ymin=-100,
          ymax=13300,
          yinterval=500,
```

```
          plot.bar=FALSE,
          plot.poly=FALSE,
          plot.line=selline,
          plot.symb=selsymb,
          symb.cex=0.5,
          )
```



To also show the variables in different colours we need to group them.

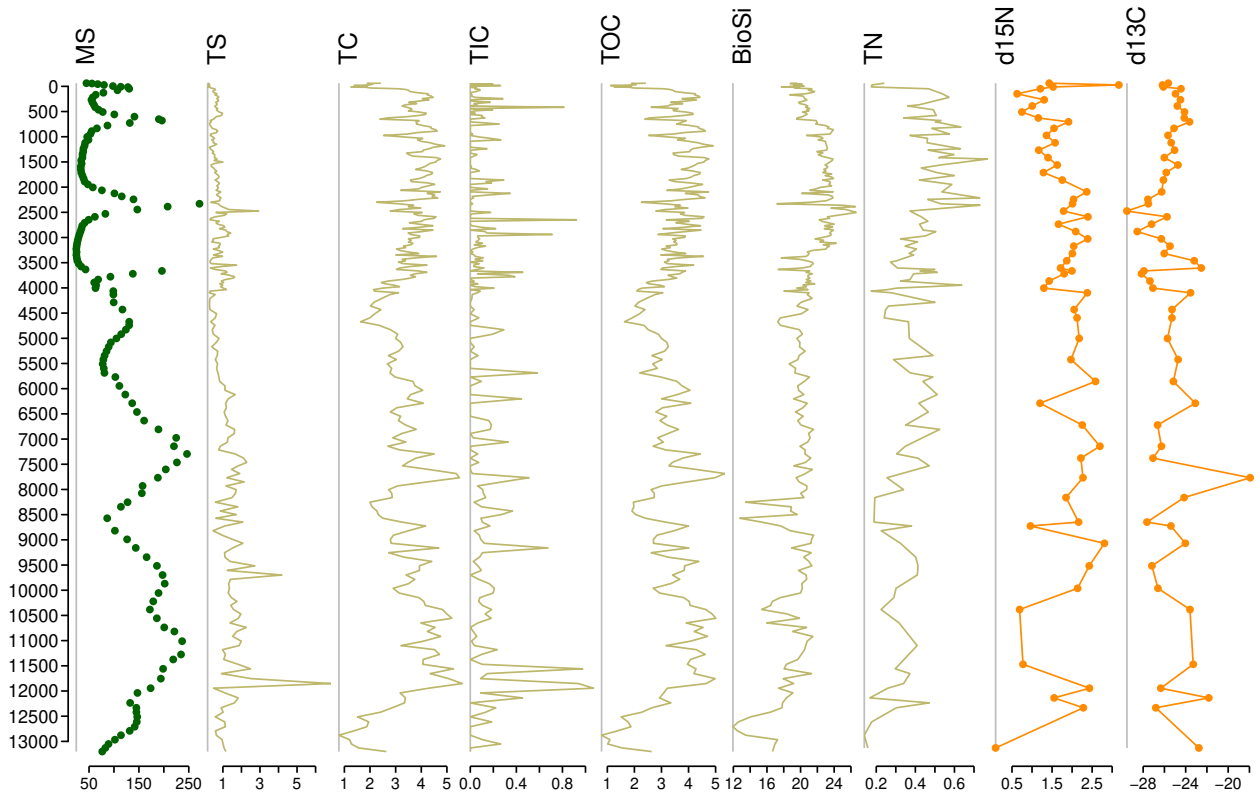
```
groups <- data.frame(names=colnames(maule.data), groups=c(1, 2, 2, 2, 2, 2, 2, 3, 3))
riojaPlot(maule.data, maule.chron, groups=groups,
          yvar.name="age_BP",
          ymin=-100,
          ymax=13300,
          yinterval=500,
          plot.bar=selbar,
          plot.poly=FALSE,
          plot.line=selline,
          plot.symb=selsymb,
          plot.groups=TRUE,
          symb.cex=0.5
          )
```



8. How do I control the tick values on the y-axis?

`riojaPlot` will try to add a y-axis with a sensible upper and lower limit and interval. It doesn't always suceed. If the default doesn't work, try specifying values for `ymin`, `ymax` and `yinterval` (and corresponding `sec.ymin` etc. value for the seconday axis). If this still doesn;t work you can pass a numeric vector of "tick values" to styles `ytks1` and `ytks2` for the primary and secondary axes respectively. In the plot above the y-axis values start at -100 and increment in 500 which gives unusal values. To fix this we create a vector of tick values from 0 to 13000 and use this.

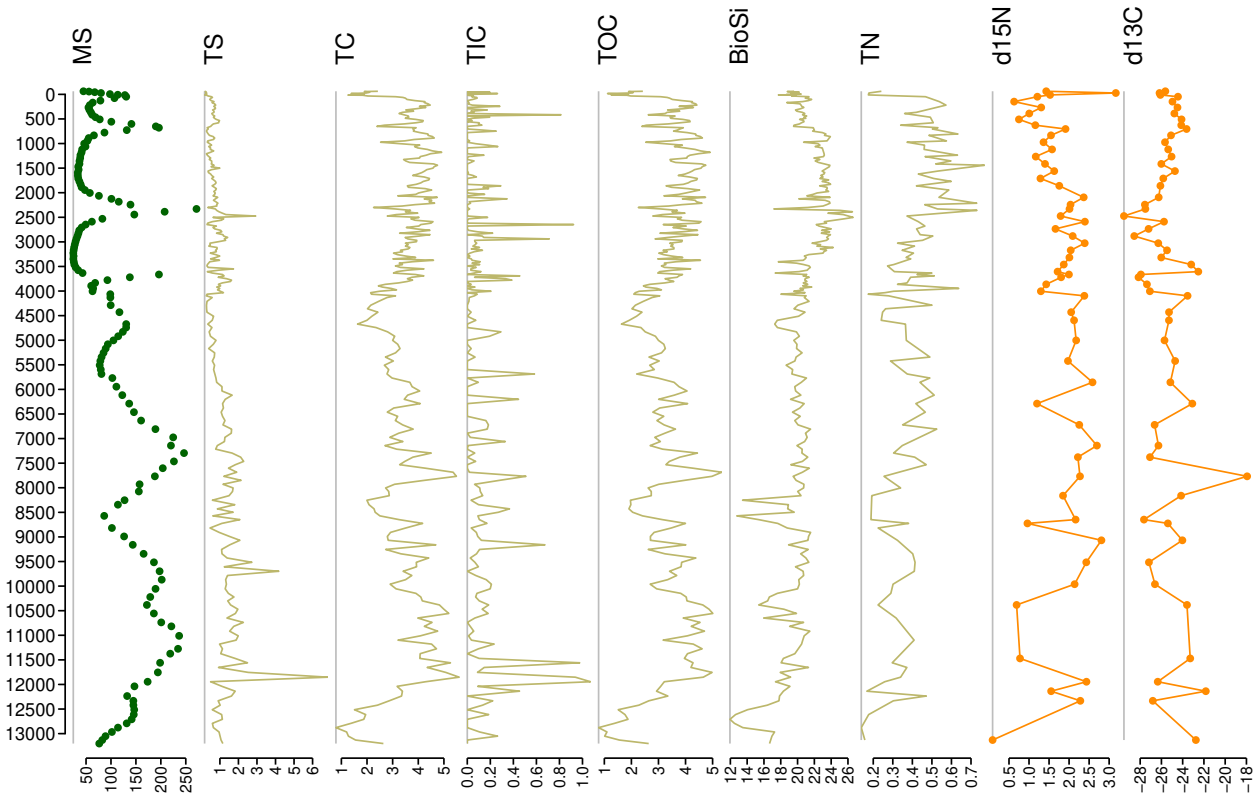
```
myticks <- seq(0, 13000, by=500)
riojaPlot(maule.data, maule.chron, groups=groups,
          yvar.name="age_BP",
          ymin=-100,
          ymax=13300,
          plot.bar=selbar,
          plot.poly=FALSE,
          plot.line=selline,
          plot.symb=selsymb,
          plot.groups=TRUE,
          ylabPos=2,
          symb.cex=0.5,
          ytks1=myticks
          )
```



9. How do I rotate the tick values on the x-axes?

USe the style `las=2`. You may also need to create a larger space at the bottom of the figure using yBottom.

```
riojaPlot(maule.data, maule.chron, groups=groups,
          yvar.name="age_BP",
          ymin=-100,
          ymax=13300,
          yinterval=500,
          plot.bar=FALSE,
          plot.poly=FALSE,
          plot.line=selline,
          plot.symb=selsymb,
          plot.groups=TRUE,
          ylabPos=2.2,
          symb.cex=0.5,
          ytks1=myticks,
          las.xaxis=2,
          yBottom=0.06
          )
```

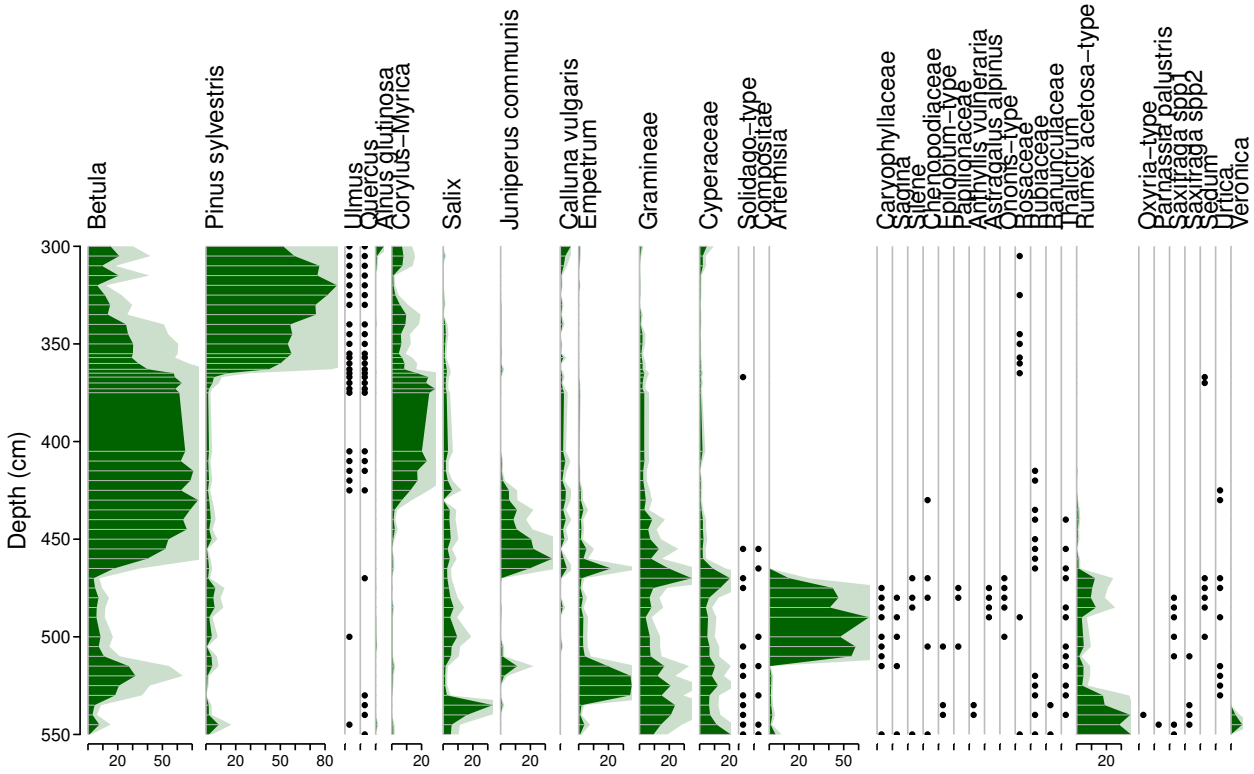


10. How do I plot symbols for rare types instead of bars or curves?

To do this we supply `plot.poly` etc. with a logical vector to control the plotting of each curve then set the values of these styles to FALSE for the rare types. Then we define a function to plot symbols for the rare types and create a list of functions to plot symbols only for the rare types.

```
aber.poll <- aber$spec
colnames(aber.poll) <- aber$names$Name
aber.chron <- aber$ages
# calculate of max of each column
mx <- apply(aber.poll, 2, max)
#create a logical vector which is TRUE for taxa with max < 5
sel <- mx < 5
# define a custom function to plot symbols
symb.fun <- function(x, y, i, nm, style) {
   sel <- x > 0
   if (sum(sel) > 0) {
      points(rep(3, sum(sel)), y[sel], cex=0.4, pch=19)
   }
}

# create a list of functions of length equal to the number of columns in the data
funlist <- lapply(1:ncol(aber.poll), function(x) symb.fun)
# now set the elements of the list where we don;t want to plot symbols to NULL
funlist[!sel] <- list(NULL)

# plot silhouettes and lines for taxa > 5% and apply our function to the others
riojaPlot(aber.poll, aber.chron,
   scale.percent=TRUE,
```

```
plot.poly=!sel,
plot.bar = !sel,
plot.line=FALSE,
lwd.bar=0.6,
plot.exag=TRUE,
fun.xfront=funlist)
```



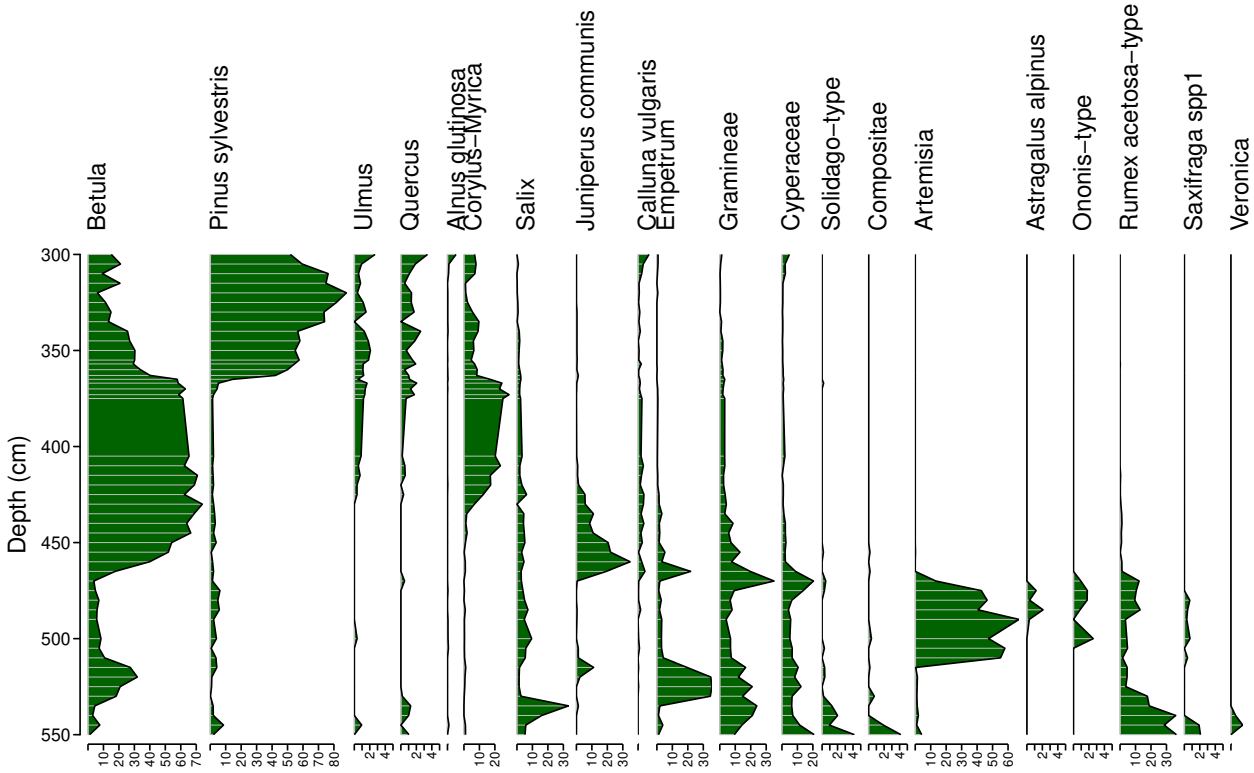
11. How do I change the widths of individual graphs?

In the above example we plotted rare types as symbols. In the example we can expand the x-axis for rare types.

For percentage diagrams the width of each graph is scaled uniformly so the percentage values for each graph are the same. We can change these using the style `graph.widths` which takes a vector of relative widths. We can also change the x-axis labels for each graph by supplying a numeric vector to style `x.inc.pc` to set the interval for x-values. Here we hide columns with maximum relative abundance $< 2\%$, and show those taxa with max $< 5\%$ as curves with 5 * normal width, and set the x-interval for these to 1% (rather than the default of 10%). We could, of course, chose the columns manually to have complete control over what is hidden and shown with expanded axes.

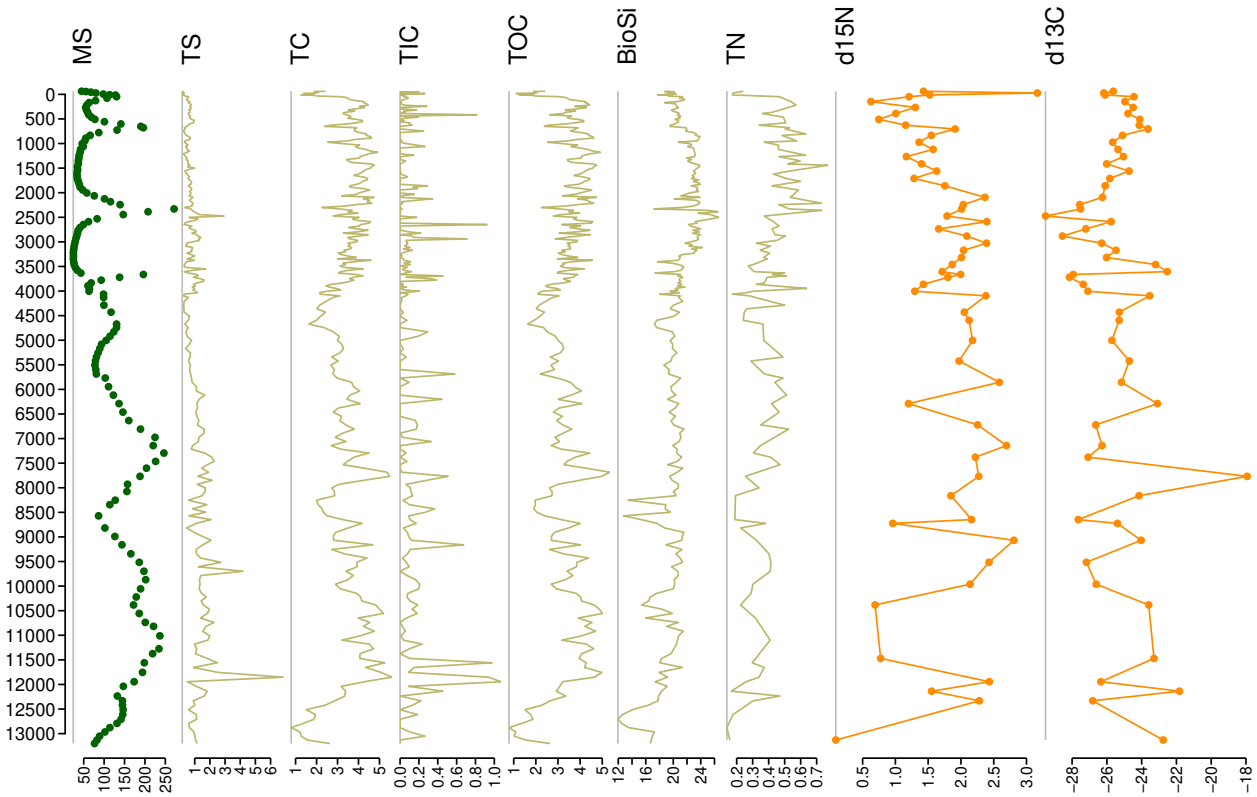
```
mx <- apply(aber.poll, 2, max) # calculate max % of each column
widths <- rep(1, ncol(aber.poll)) # generate numeric vector of widths, set to 1
inc <- rep(10, ncol(aber.poll)) # generate numeric vector of x-intervals, set to 10
selTaxa <- names(mx[mx > 2]) # generate character vector of column names to include in the figure
sel <- which(mx < 5) # generate a logical vector of columns to expand
widths[sel] <- 5 # set widths to 5 times normal width
inc[sel] <- 1 # set x-interval to 1%

riojaPlot(aber.poll, aber.chron, selVars=selTaxa,
          scale.percent=TRUE,
          graph.widths=widths,
```

```
          min.width.pc=5,
          x.pc.inc=inc,
          cex.xaxis=0.5,
          las.xaxis=2)
```



We can also change the widths for non-percent data. Here we return to the Laguna del Maule data in FAQ 7 and change the width of d15N and d13C columns to twice the standard width.

```
widths <- rep(1, ncol(maule.data))
#sel <- names(maule.data) %in% c("d15N", "d13C")
#widths[sel] <- 2
widths[8:9] <- 2
riojaPlot(maule.data, maule.chron, groups=groups,
          yvar.name="age_BP",
          ymin=-100,
          ymax=13300,
          yinterval=500,
          plot.bar=FALSE,
          plot.poly=FALSE,
          plot.line=selline,
          plot.symb=selsymb,
          plot.groups=TRUE,
          ylabPos=2.2,
          symb.cex=0.5,
          ytks1=myticks,
          las.xaxis=2,
          graph.widths=widths,
          yBottom=0.06
          )
```

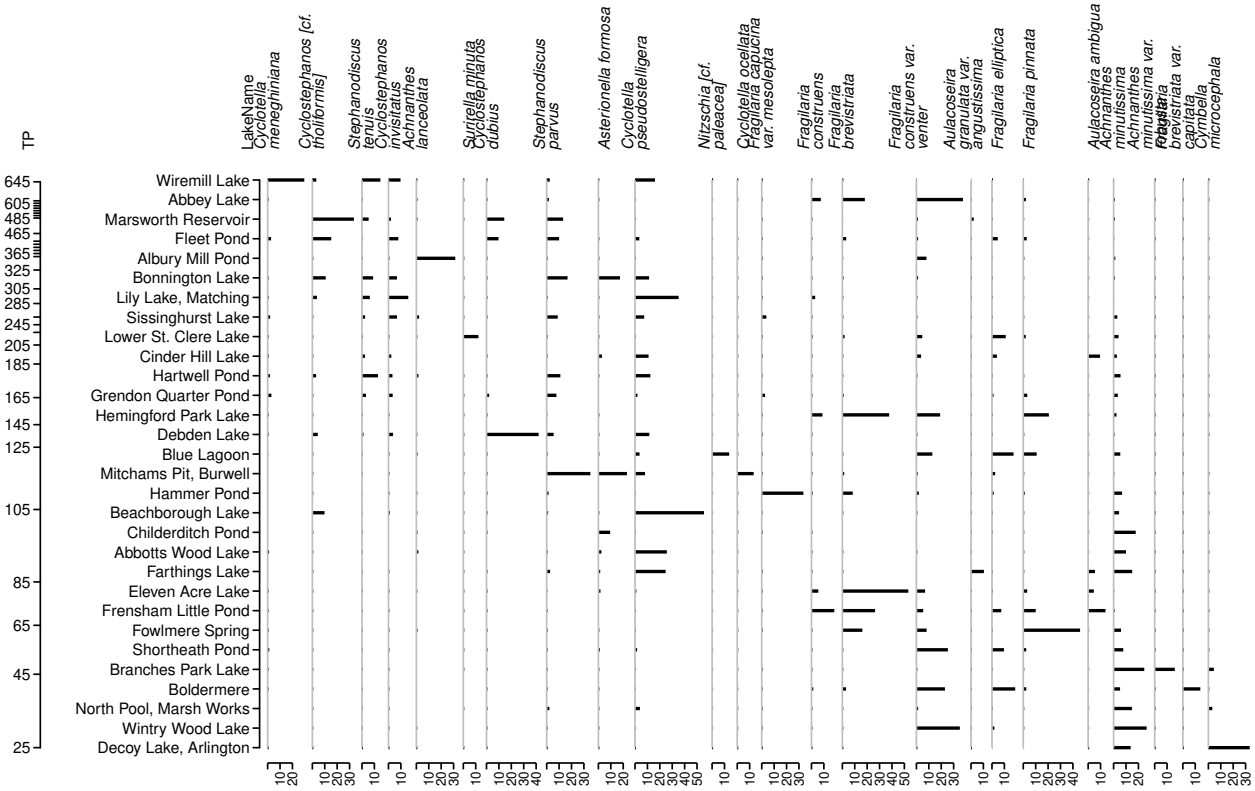


12. How do I plot surface pollen or diatom data?

`riojaPlot` can plot surface sediment data, you just need to think how to arrange samples on the y-axis. The example below plots surface sediment diatom data from small ponds and pools in SE England collected by Bennion (1994) (see ?rioja::Ponds) for details. Here we arrange the samples along the TP gradient and plot the data against Lake name with TP as a secondary y-axis.

```
data(Ponds)
# reorder the rows in decreasing TP
o <- order(Ponds$env$TP, decreasing=TRUE)
ponds.diat <- Ponds$spec[o, ]
ponds.TP <- data.frame(TP=round(Ponds$env$TP[o]), LakeName=Ponds$env$Name[o])
# replace taxon codes with names
colnames(ponds.diat) <- Ponds$names$Name
# remove rare species
mx <- apply(ponds.diat, 2, max)
ponds.sel <- colnames(ponds.diat)[mx > 10]

# With a large number of samples we would use points but with only 30
# in this data we use bars.  Although the data are arranged in order of
# increasing TP they are not a temporally or spatially contiguous so
# lines or silhouettes are not appropriate.

riojaPlot(ponds.diat, ponds.TP, selVars=ponds.sel,
          yvar.name="LakeName",
          sec.yvar.name="TP",
          sec.yinterval = 20,
          plot.sec.axis=TRUE,
          scale.percent=TRUE,
```

```
        plot.poly=FALSE,
        plot.line=FALSE,
        plot.bar=TRUE,
        col.bar="black",
        lwd.bar = 2,
        symb.cex=0.4,
        cex.xlabel=0.6,
        cex.yaxis=0.6,
        wa.order="topleft",
        labels.italicise=TRUE,
        las.xaxis=2,
        cex.xaxis=0.5)
```

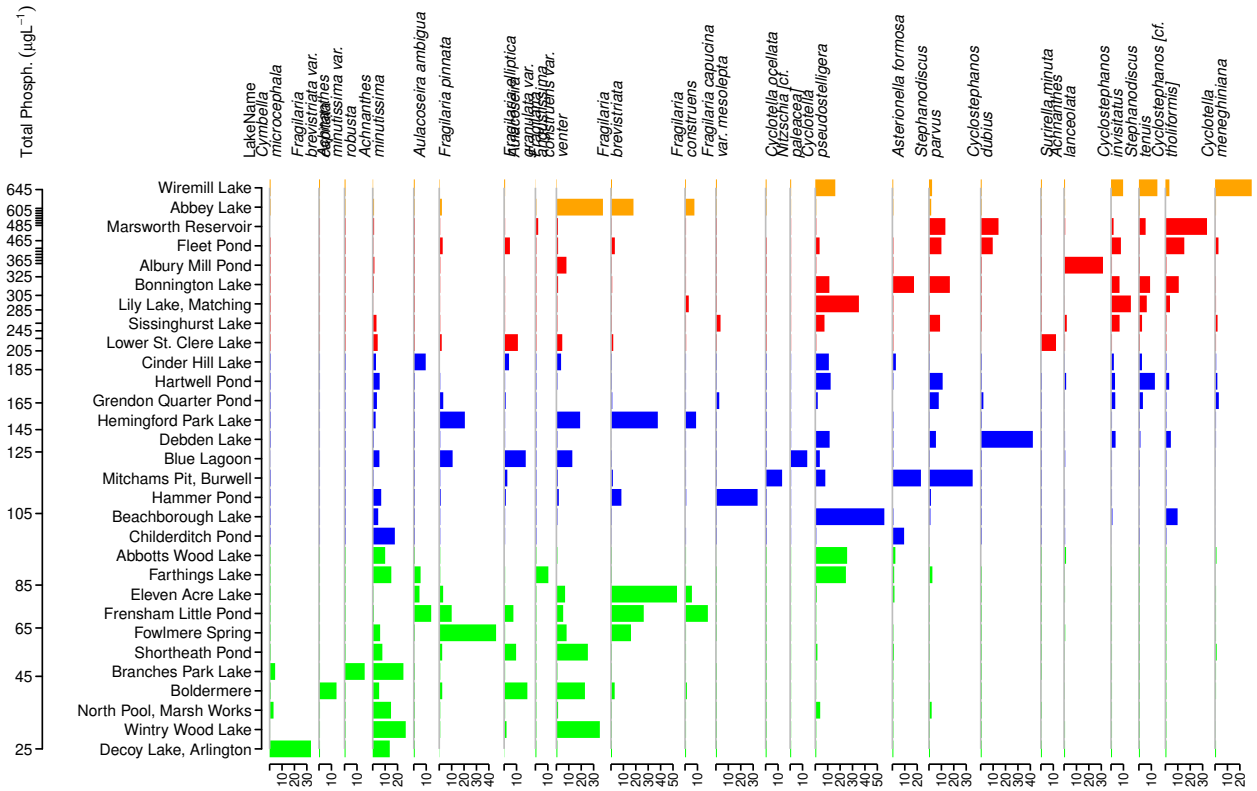


13. How do I plot different levels or samples with a different colour?

It is currently only possible to plot different samples with different coloured bars. Here we extend the example for FAQ 12 above and classify the samples into different groups based on their TP, then plot these different samples with different colours. We do this by creating a character vector of colour names, that corresponds to the samples, then pass this to style `col.sep.bar` and set `sep.bar` to true. We also create an expression to label the TP axis in the correct units.

```
ponds.TP <- ponds.TP %>%  mutate(bar.cols=case_when(TP < 100 ~ "green",
                            TP >= 100 & TP < 200 ~ "blue",
                            TP >= 200 & TP < 500 ~ "red",
                            TRUE ~ "orange"))

ylab <- expression(Total~Phosph.~(mu*gL^{-1}))

riojaPlot(ponds.diat, ponds.TP, selVars=ponds.sel,
          yvar.name="LakeName",
```

```
    sec.yvar.name="TP",
    sec.ylabel=ylab,
    sec.yinterval = 20,
    plot.sec.axis=TRUE,
    scale.percent=TRUE,
    plot.poly=FALSE,
    plot.line=FALSE,
    plot.bar=TRUE,
    col.bar="black",
    sep.bar=TRUE,
    col.sep.bar=ponds.TP$bar.cols,
    lwd.bar = 10,
    symb.cex=0.4,
    cex.xlabel=0.6,
    cex.yaxis=0.6,
    wa.order="bottomleft",
    labels.italicise=TRUE,
    las.xaxis=2,
    cex.xaxis=0.5)
```

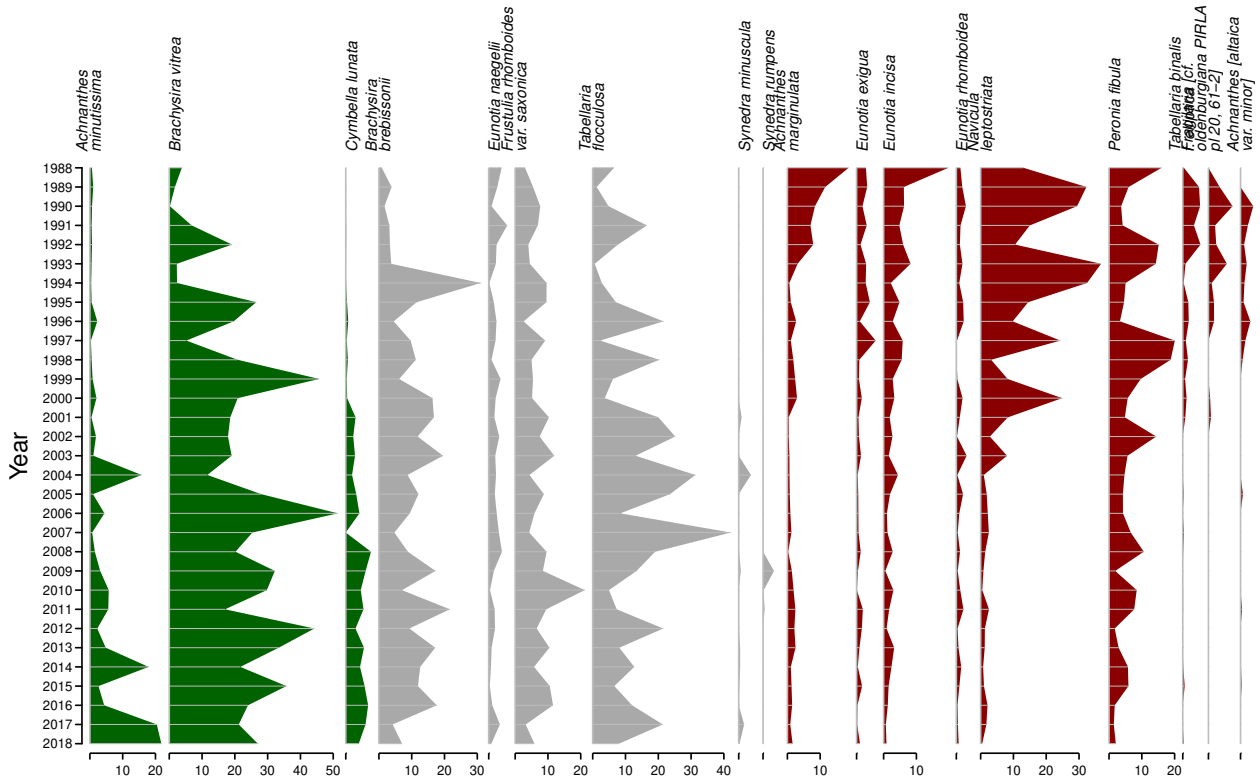


14. How do I plot ecological time series?

`riojaPlot` can also be used to plot multi-species ecological time series (or spatial transect) data. Here we plot diatom data from the UK Upland Waters Monitoring Network (https://uwmn.uk/). These are epilithic diatom samples collected annually from 1988-2018 from Loch Chon (Trossachs, Scotland). These data were collected to monitor the potential recovery of the loch from previous acidification. The file of diatom counts (expressed as relative abundance) also contains results of a trend test based on a multivariate generalised linear model using the `manyglm` function in package `mvabund` used to identify taxa that have a significant increasing or decreasing temporal trend.

We first re-order the taxa according to their trend over time and plot the data with silhouettes, coloured to show taxa with significant increasing, decreasing or no trend.

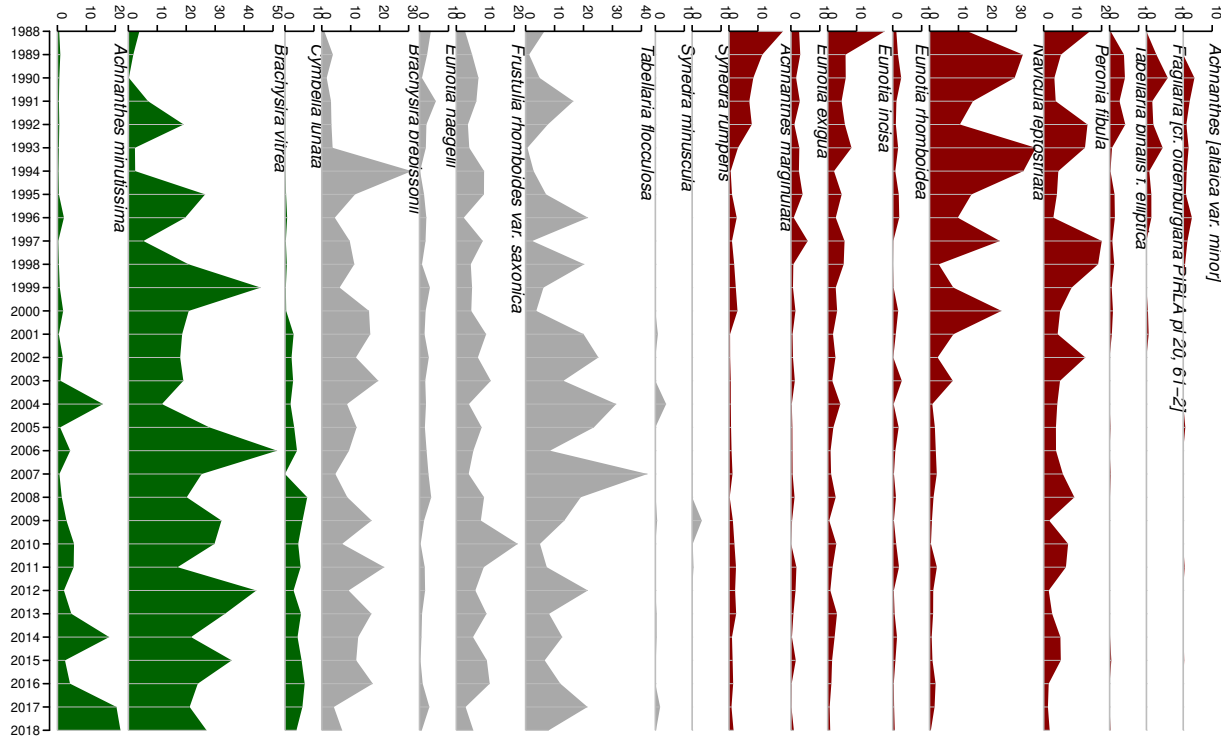
```
fpath <- system.file("extdata/LochChon.xlsx", package="riojaPlot")
chon <- readxl::read_excel(fpath, sheet="Diatoms")
sig_test <- readxl::read_excel(fpath, sheet="Sig_test")

chon.diat <- chon %>% select(-Year)
chon.year <- chon %>% select(Year)
# data has many rare species, remove these first
mx <- apply(chon.diat, 2, max)
chon.diat <- chon.diat[, mx > 3]

# Create a grouping variable based on significance of trend (increasing, decreasing or not sig.)
chon.groups <- data.frame(TaxonName=colnames(chon.diat)) %>%
  left_join(sig_test, by="TaxonName") %>%
  mutate(group = case_when(p_unadj < 0.1 & slope>0 ~ "Increasing",
                           p_unadj < 0.1 & slope<0 ~ "Decreasing",
                           TRUE ~ "Not sig"),
         group=factor(group, levels=c("Increasing", "Not sig", "Decreasing"))) %>%
  select(TaxonName, group)

# Sort variables according to significance
diat.order <- chon.groups %>% arrange(group) %>% pull(TaxonName)

chon.ordered <- chon.diat %>% select(!!diat.order)
riojaPlot(chon.ordered, chon.year, groups=chon.groups,
          scale.percent=TRUE,
          yinterval=1,
          plot.poly=TRUE,
          plot.line=FALSE,
          plot.groups=TRUE,
          cex.xlabel=0.6,
          cex.yaxis=0.5,
          labels.italicise=TRUE,
          cex.xaxis=0.5,
          col.group=c("darkgreen", "darkgrey", "darkred"))
```



This looks OK but we can "virtually" rotate this figure so it can be viewed as a stacked diagram. To do this we have to modify the axis labels and orientation of taxon names. We can't easily do this using riojaPlot styles but we can suppress the plotting of names and x-axis and create a custom function to plot these in any style or position we wish.

```
# create list of blank names to suppress taxon names
names <- rep("", ncol(chon.ordered))
myfun <- function(x, y, i, nm, style) {
  usr <- par("usr") # extract the x and y data limits of the plot
  name <- bquote(italic(.(names(nm)))) #  extract name from nm and italicise it
  text(usr[2]-2, usr[4]+0.5, name, adj=c(0, 0), xpd=NA, srt=-90, cex=0.7)
  xval <- seq(0, usr[2], by=10) # create a vector of labels for the axis
  xlab <- rep("", length(xval))
  axis(side=3, at=xval, labels=xlab, tcl=-0.2, cex.axis=0.5, mgp=c(3, 0.1, 0))
  text(xval, usr[4]-0.5, xval, cex=0.5, srt=-90, adj=c(1, 0), xpd=NA) # ad the x-axis vales
}

riojaPlot(chon.ordered, chon.year, groups=chon.groups,
          scale.percent=TRUE,
          yinterval=1,
          xlabels=names, # replace taxon names with a vector of blank names
          ylabel = " ",  # suppress y-axis label
          plot.poly=TRUE,
          plot.line=FALSE,
          plot.groups=TRUE,
          cex.xlabel=0.6,
          cex.yaxis=0.5,
          labels.italicise=TRUE,
          cex.xaxis=0.5,
```

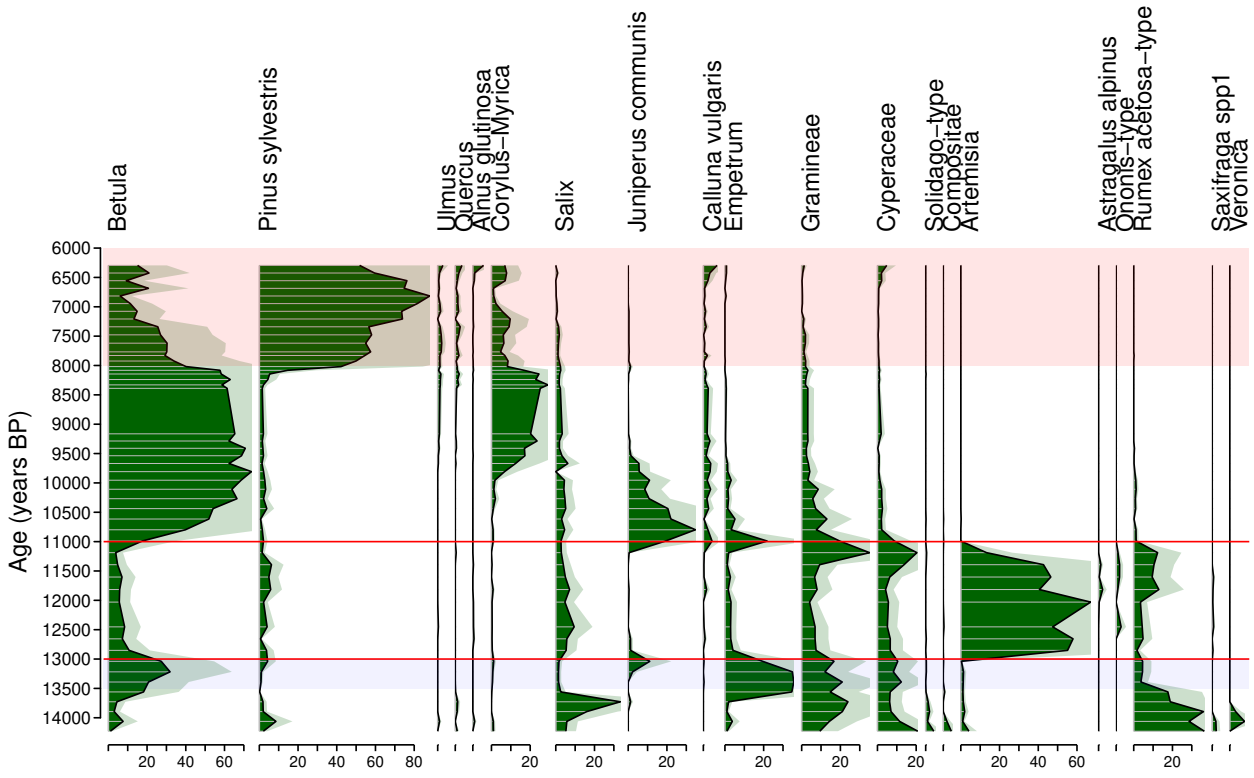
```
col.group=c("darkgreen", "darkgrey", "darkred"),
plot.bottom.axis=FALSE,
min.width.pc=10, # set minimum size of x-axes
fun.xfront=myfun)
```



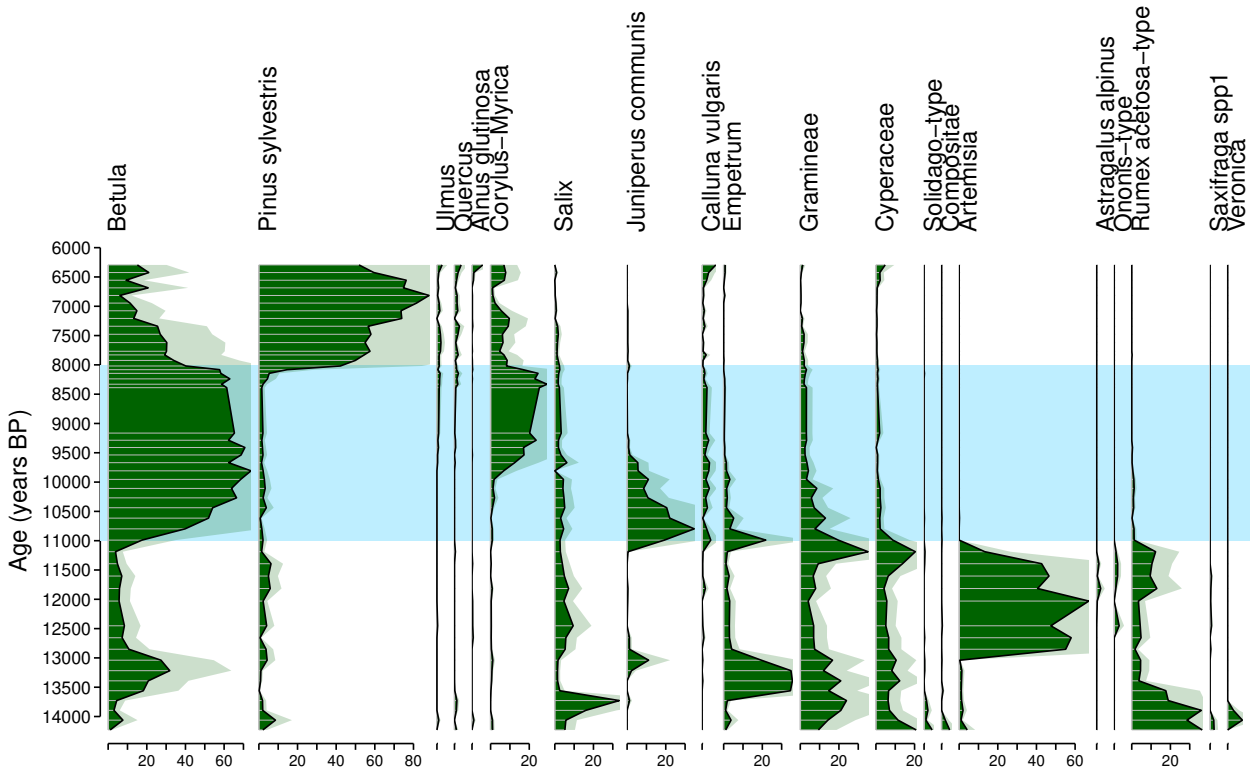
15. How do I add my own zones to a diagram?

Arbitrary zones can be added using function `addRPZone`, as either single lines, or as shaded rectangles. Simply save the riojaPlot object and pass it to `addRPZone`.

```
mx <- apply(aber.poll, 2, max)
aber.sel <- names(mx[mx > 2])

rp <- riojaPlot(aber.poll, aber.chron, aber.sel,
   yvar.name="Age (years BP)",
   ymin=6000,
   ymax=14300,
   yinterval=500,
   scale.percent=TRUE,
   plot.exag=TRUE)

# define zone lines at 11000 and 13000 years BP
myzones <- c(11000, 13000)
addRPZone(rp, myzones, col="red")

# add shaded zone from 6000 to 8000 years BP
addRPZone(rp, 6000, 8000)

# change colour and shading
addRPZone(rp, 13000, 13500, col="blue", alpha = 0.05)
```




```
# The zones above are plotted over the existing diagram.
# It is possible to plot zones or other annotations behind
# the diagram using a custom function supplied to style fun.plotback
# this takes 2 arguments, usr = vector of 4 numbers giving the
# coordinates of the plotting area in data units (x is 0-1)
# and fig,coordinates of the plotting area as fractions of the page size

fun.back <- function(usr, fig, style) {
  print(usr)
  rect(0, 8000, 1, 11000, col="lightblue1", border=NA)
}

rp <- riojaPlot(aber.poll, aber.chron, aber.sel,
   yvar.name="Age (years BP)",
   ymin=6000,
   ymax=14300,
   yinterval=500,
   scale.percent=TRUE,
   plot.exag=TRUE,
   fun.plotback=fun.back
   )
#> [1]     0     1 14300  6000
```



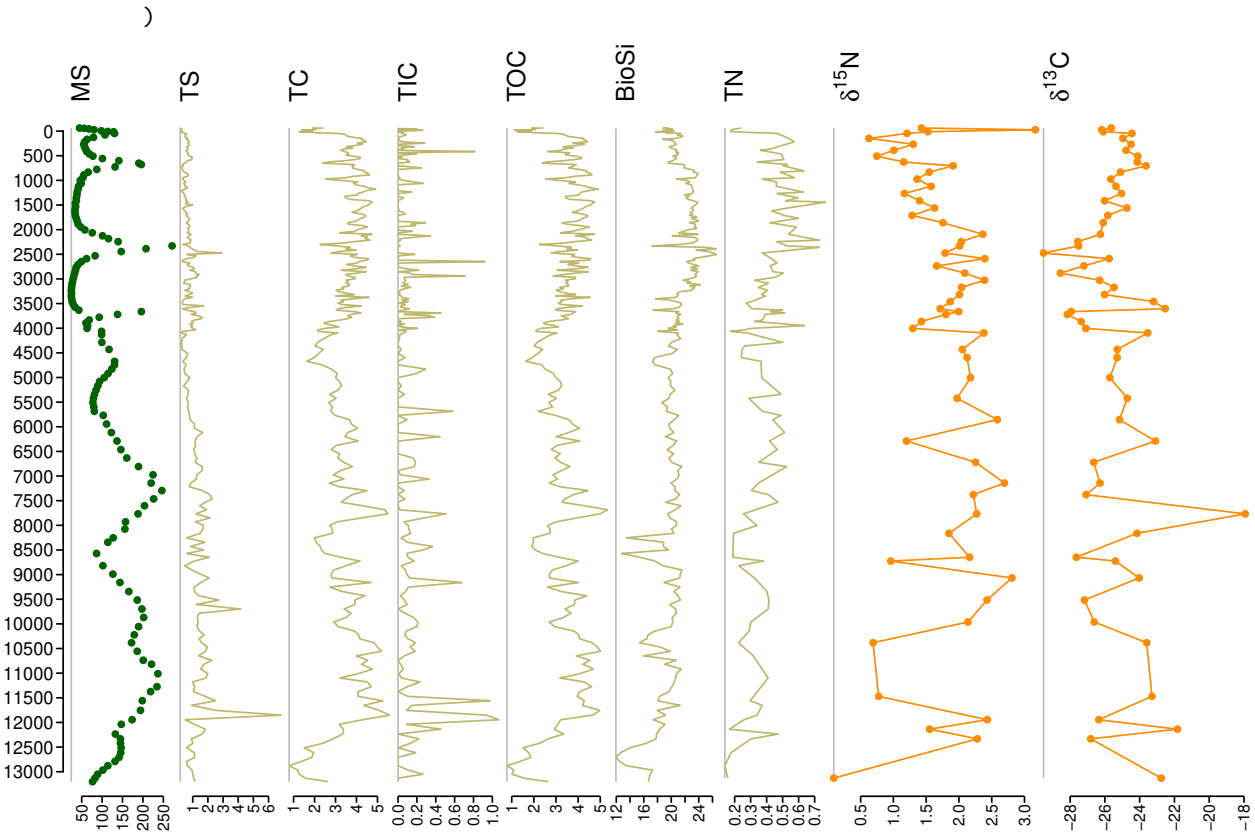
16. How do I include special symbols in the variable names

We can pass a character vector of custom names for each column using style `xlabels` and use `expression` to create names with superscripts, subscripts or greek letters.

```
nms <- colnames(maule.data)
nms[8] <- expression(delta^15*N)
```

We can also use function `label_geochem` in package `tidypaleo` to automatically add annotations for a slected range of common geochemical variables.

```
library(tidypaleo)
nms <- colnames(maule.data)
nms2 <- as.expression(sapply(nms, function(x) unlist(label_geochem(x)[[1]])))
riojaPlot(maule.data, maule.chron, groups=groups,
          yvar.name="age_BP",
          xlabels=nms2,
          ymin=-100,
          ymax=13300,
          yinterval=500,
          plot.bar=FALSE,
          plot.poly=FALSE,
          plot.line=selline,
          plot.symb=selsymb,
          plot.groups=TRUE,
          ylabPos=2.2,
          symb.cex=0.5,
          ytks1=myticks,
          las.xaxis=2,
          graph.widths=widths,
          yBottom=0.06
```



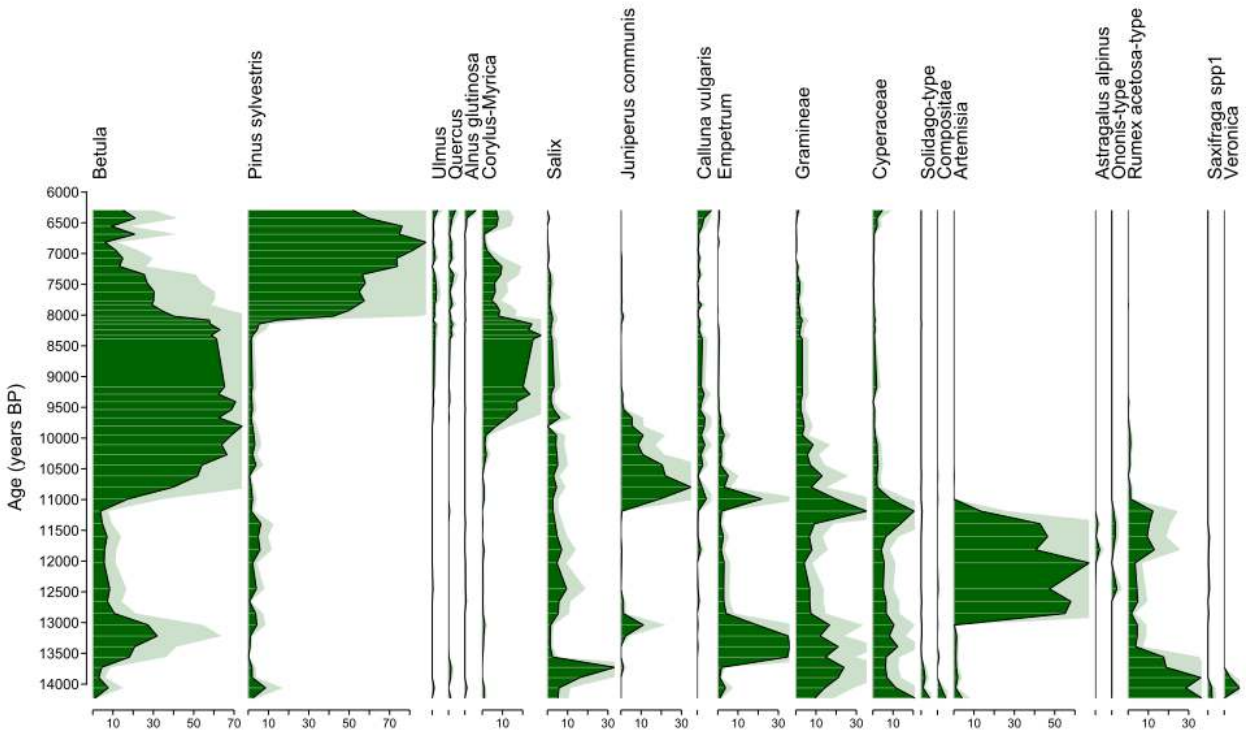
17. What is the best way to save riojaPlot diagrams?

'`riojaPlot` figures can be saved to file or copied to the clipboard from R or RStudio. This will work but will not always give the best diagram because the plot is scaled to the window it is drawn in, and saving a plot that is already drawn to a different size or aspect ratio may result in unexpected gaps between axes or truncated text. To avoid this, place device calls around the plot to explicity draw the figure to a graphics file. For a stand-alone figure `pdf` is appropriate. For a figure to include in a document or presentation `svg` will usually work best for embedding in recent MS Office documents, or `png` for earlier documents. Just be careful that if you get an error during the plot to remember to call `dev.off()` to return plotting to the screen.

```
svg("Myplot1.svg", width=10, height=6) # figure dimensions in inches
# or pdf("Myplot.pdf", width=10, height=6) for output to a pdf file
riojaPlot(aber.poll, aber.chron, aber.sel,
   yvar.name="Age (years BP)",
   ymin=6000,
   ymax=14300,
   yinterval=500,
   scale.percent=TRUE,
   plot.exag=TRUE,
   )
dev.off()
#> pdf
#>   2
```

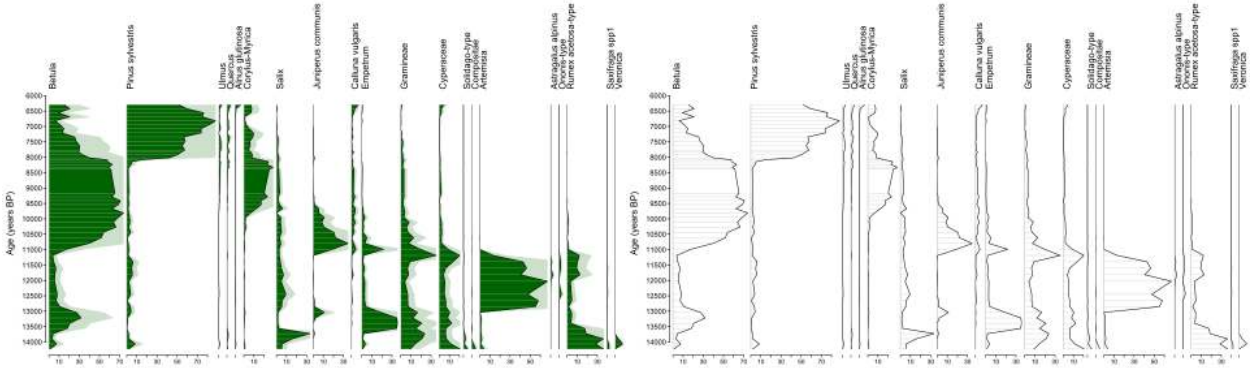
To view a saved svg or pdf file use `cowplot::draw_image`.

```
cowplot::ggdraw() + cowplot::draw_image("Myplot1.svg")
```



Or save the plots to an object and combine using plot_grid:

```
svg("Myplot2.svg", width=10, height=6) # figure dimensions in inches
riojaPlot(aber.poll, aber.chron, aber.sel,
   yvar.name="Age (years BP)",
   ymin=6000,
   ymax=14300,
   yinterval=500,
   scale.percent=TRUE,
   plot.poly=FALSE
   )
dev.off()
#> pdf
#>   2
p1 <- cowplot::ggdraw() + cowplot::draw_image("Myplot1.svg")
p2 <- cowplot::ggdraw() + cowplot::draw_image("Myplot2.svg")
cowplot::plot_grid(p1, p2, align="v", axis="tblr")
```

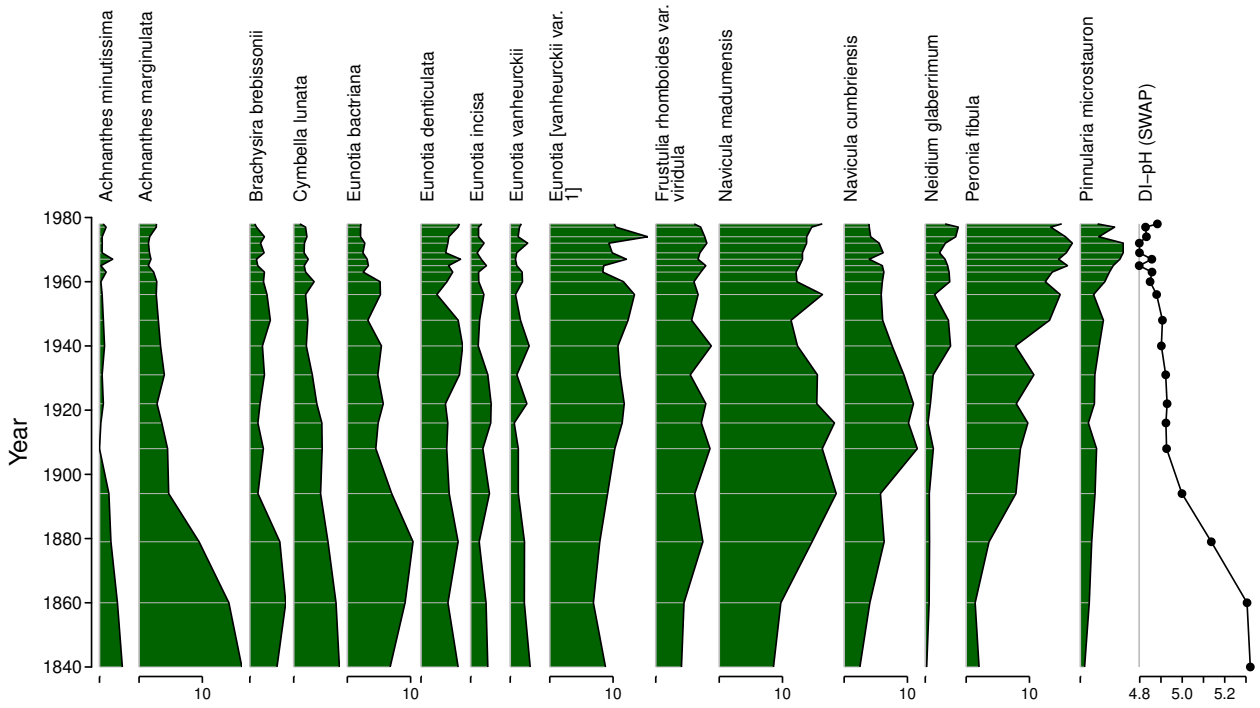


```
file.remove("Myplot1.svg")
#> [1] TRUE
file.remove("Myplot2.svg")
#> [1] TRUE
```

18. Can I use a pipe with riojaPlot

Yes, for diagrams with multiple parts you can pipe the output from one section into the next. Use function `riojaPlot2` to add additional dataset to the plot. This is just a wrapper around `riojaPlot()` that takes a `riojaPlot` object as the first argument. Here is the example from section 8 above rewritten to use the pipe.

```
riojaPlot(RLGH.diat, RLGH.chron,
          yvar.name="Year",
          scale.percent=TRUE,
          y.rev=FALSE,
          selVars=RLGH.sel,
          ymax=1980,
          xlabels=RLGH$names$TaxonName,
          cex.xlabel=0.7,
          labels.break.n=25,
          xRight=0.9) |>
  riojaPlot2(RLGH.pH, RLGH.chron[, "Year", drop=FALSE],
          scale.minmax=FALSE,
          plot.bar=FALSE,
          plot.symb=TRUE,
          symb.cex=0.6)
```



19. How do I cite riojaPlot in a publication

```
#>
#> To cite package 'riojaPlot' in publications use:
#>
```

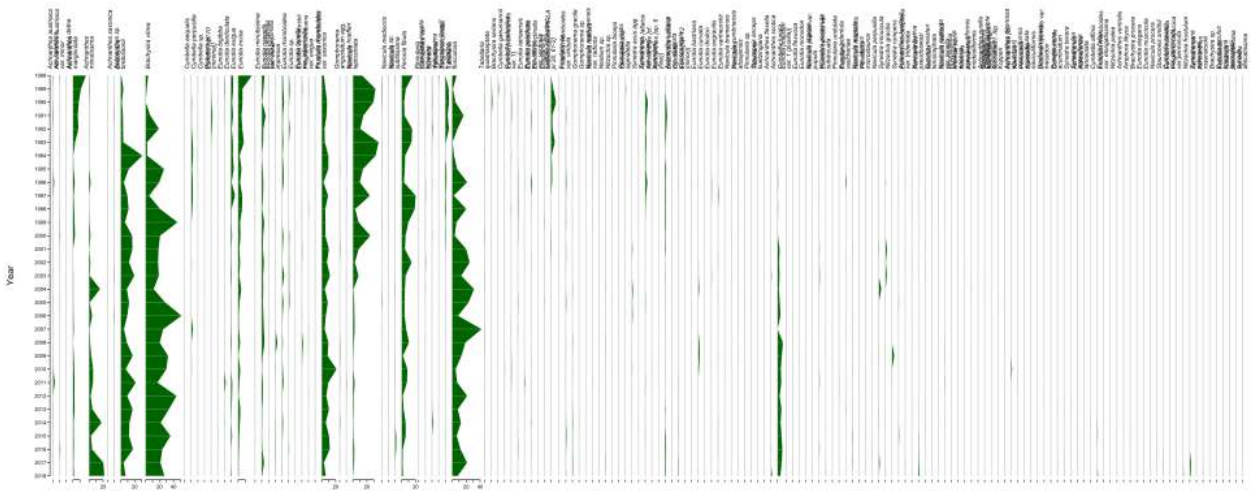
```
#>   Juggins, S. (2022) riojaPlot: Stratigraphic diagrams in R, package
#>   version (0.1-19). (https://cran.r-project.org/package=riojaPlot).
#>
#> A BibTeX entry for LaTeX users is
#>
#>   @Manual{,
#>     title = {riojaPlot: Stratigraphic diagrams in R},
#>     author = {Steve Juggins},
#>     year = {2022},
#>     note = {R package version 0.1-19},
#>     url = {https://cran.r-project.org/package=riojaPlot},
#>   }
```

20. I get an error "Too many variables, curves will be too small"

If you try to plot too many variable in a single diagram the width of each will be too small to show in a meaningful way and riojaPlot will issue this error message. To fix this either reduce the number of variables or increase the size of the plotting window. If you want to view onscreen you can open a new plotting window with the `windows(width=10, height=5)` (on Windows) or `quartz(width=10, height=5)` (on MacOS), where width and height are in inches. Don't forget to close the windows when finished to return plotting to the R/Rstudio plot window. If you want to save to file just use function `pdf()` or `svg()` with appropriate width / height (see FAQ 17 above).

```
fpath <- system.file("extdata/LochChon.xlsx", package="riojaPlot")
chon <- readxl::read_excel(fpath, sheet="Diatoms")

chon.diat <- chon %>% select(-Year)
chon.year <- chon %>% select(Year)

# this will give an error
riojaPlot(chon.diat, chon.year,
          scale.percent=TRUE,
          yinterval=1,
          plot.poly=TRUE,
          plot.line=FALSE,
          plot.groups=TRUE,
          cex.xlabel=0.6,
          cex.yaxis=0.5,
          labels.italicise=TRUE,
          cex.xaxis=0.5)
#> Error in .riojaPlot2(d, yvar = yvar, y.rev = style$y.rev, scale.percent = style$scale.percent, : Too

# but this will work

svg("MyPlot.svg", width=20, height=8)
riojaPlot(chon.diat, chon.year,
          scale.percent=TRUE,
          yinterval=1,
          plot.poly=TRUE,
          plot.line=FALSE,
          plot.groups=TRUE,
          cex.xlabel=0.6,
          cex.yaxis=0.5,
          labels.italicise=TRUE,
          cex.xaxis=0.5)
```

```
dev.off()
#> pdf
#>   2

cowplot::ggdraw() + cowplot::draw_image("Myplot.svg")
```




```
file.remove("Myplot.svg")
#> [1] TRUE
```

‘‘‘ ## 11. List of modifiable styles and default values

Style	Value
yvar.name	
sec.yvar.name	
ylabel	
sec.ylabel	
plot.sec.axis	FALSE
scale.percent	FALSE
scale.minmax	FALSE
minmax	NA
y.rev	TRUE
ymin	NA
ymax	NA
yinterval	NA
sec.ymin	NA
sec.ymax	NA
sec.yinterval	NA
xlabels	NA
wa.order	none

Style	Value
plot.bar	TRUE
plot.line	TRUE
plot.poly	FALSE
plot.symb	FALSE
do.clust	FALSE
plot.clust	FALSE
plot.zones	0
lwd.bar	0.5
lwd.line	1
lwd.poly.line	0.5
lwd.cumul.line	0.5
col.bar	grey
bar.back	FALSE
col.symb	black
col.line	black
col.poly	darkgreen
col.poly.line	NA
col.cumul.line	NA
col.zones	red
col.zone.column	grey
lwd.zones	1
symb.pch	19
symb.cex	1
cex.xaxis	0.6
cex.yaxis	0.7
cex.ylabel	0.9
cex.xlabel	0.9
srt.xlabel	90
srt.ylabel	NA
centre.xlabel	FALSE
tcl	-0.2
cex.cumul	0.7
clust.data.trans	none
clust.use.selected	FALSE
clust.width	0.05

Style	Value
graph.widths	1
plot.exag	FALSE
col.exag	auto
col.exag.line	NA
lwd.exag.line	0.6
exag.mult	2
exag.alpha	0.2
labels.break.long	TRUE
labels.break.n	20
labels.italicise	FALSE
plot.groups	FALSE
plot.cumul	FALSE
cumul.mult	1
col.group	darkgreen, darkkhaki, darkorange, darkred, deepskyblue, sienna3, darkgoldenrod3, darkseagreen, yellow3, darkgrey
xRight	0.99
xLeft	NA
yBottom	0.05
yTop	NA
fun.xback	NA
fun.xfront	NA
fun.plotback	NA
fun.yaxis	NA
ylabPos	NA
xlabPos	0.1
xSpace	0.005
x.pc.omit0	TRUE
lwd.axis	1
col.axis	grey
min.width.pc	5
las.xaxis	1
las.yaxis	1
ytks1	NA
ytks2	NA
omitMissing	TRUE

Style	Value
col.sep.bar	black
sep.bar	FALSE
plot.bottom.axis	TRUE
plot.top.axis	FALSE
cumulSpace	NA
plot.yaxis	TRUE
start.new.plot	TRUE
xGap	0.01
x.pc.inc	10
fun.lithology	NA
lithology.width	0.03
user1	NA
user2	NA
user3	NA
user4	NA

12. References

Allen, J.R.M. *et al.* (1999) Rapid environmental changes in southern Europe during the last glacial period. Nature, 400, 740-743.

Bennion, H. (1994) A diatom-phosphorus transfer function for shallow, eutrophic ponds in southeast England. *Hydrobiologia*, **275/276**, 391-410.

Frugone-Álvarez, M. et al. (2020) Volcanism and climate change as drivers in Holocene depositional dynamic of Laguna del Maule (Andes of central Chile - 36S). Climate of the Past, DOI 10.5194/cp-2019-147.