R_syntax_04

October 20, 2022

1 R syntax in a nutshell – Part IV Graphics

- 2.1 Obtaining and annotating binary plots
- 2.2 Additional high-level plotting functions
- 2.3 Creating custom layouts and axes
- 2.4 Exporting graphs from R; graphical devices
- 2.5 Interaction with plots

A key benefit of using R is the large range of functions for the production and export of (near) publication-quality diagrams. These functions can be divided into two types; **high-level** functions that open a new graphical window and set up a coordinate system of the brand new graph and **low-level** functions that annotate pre-existing plots. Note also that some of the functions (e.g., `curve`) can show both types of behaviour depending on its arguments.

An overview of the selected high-level graphical functions in R Function

Purpose

plot(x,y)

binary plot x vs. y (two numeric vectors)

curve(expr,from,to)

curve specified by expr (written as a function of x) in the interval from–to

contour(x,y,z)

contour plot (x and y specify a regular grid, z the values)

filled.contour(x,y,z)

filled contour plot (x and y specify a regular grid, z the values)

boxplot(x)

"box-and-whiskers" plot

coplot(formula)

conditioning plot; if formula = y~x|z, bivariate plots of x vs. y for each level of the factor z

pairs(x)

matrix of all possible bivariate plots between columns of x (matrix or data frame)

hist(x)

histogram of frequencies for x

pie(x)

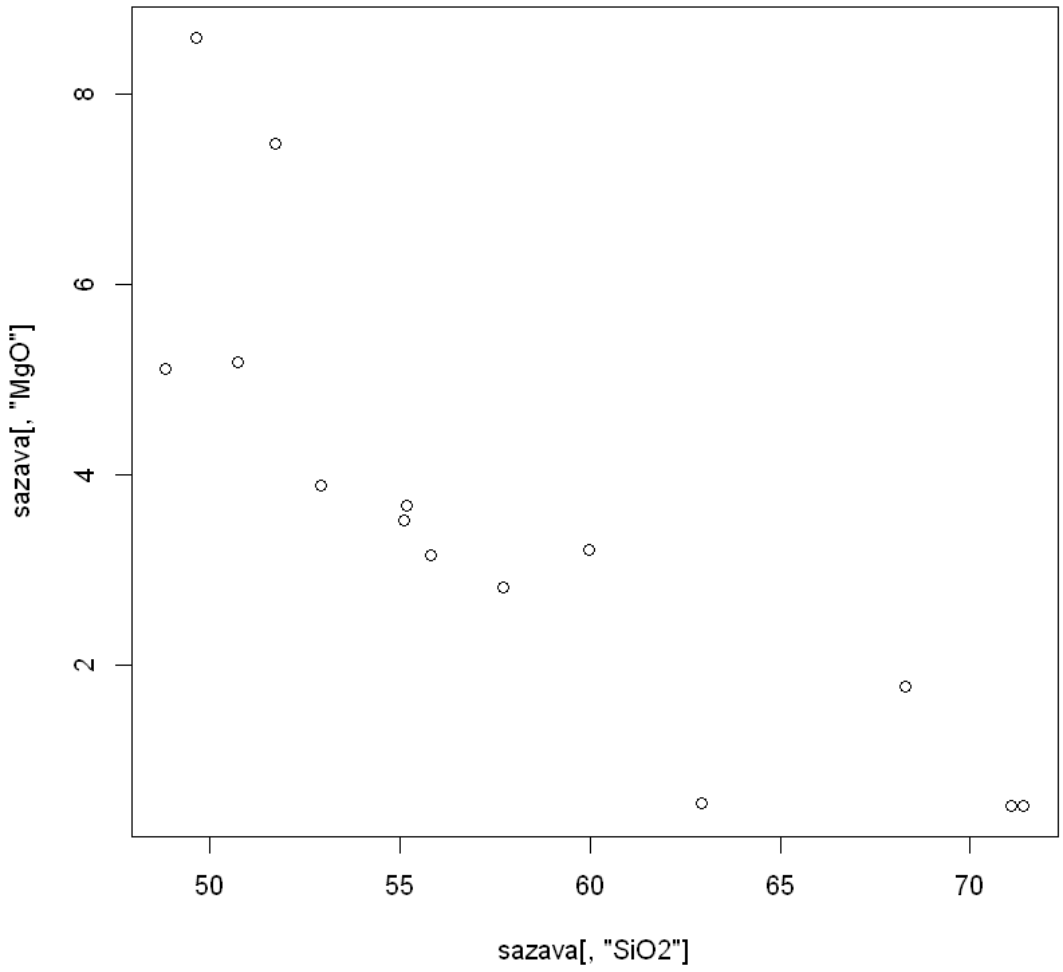
circular pie-chart

1.1 2.1 Obtaining and annotating binary plots

plot() This is a key plotting function. For two numeric vectors, it produces a binary plot [`plot(x,y)`]. If one vector is shorter, it is recycled as appropriate.

We can demonstrate plotting of binary diagrams using data from the calc-alkaline Sázava suite of the Variscan Central Bohemian Plutonic Complex (Janoušek et al. 2004):

[1]:
```
sazava  <-  read.table("data/sazava.data",sep="\t")
plot(sazava[,"SiO2"],sazava[,"MgO"])
```



1.1.1 Setting plotting options in Jupyter

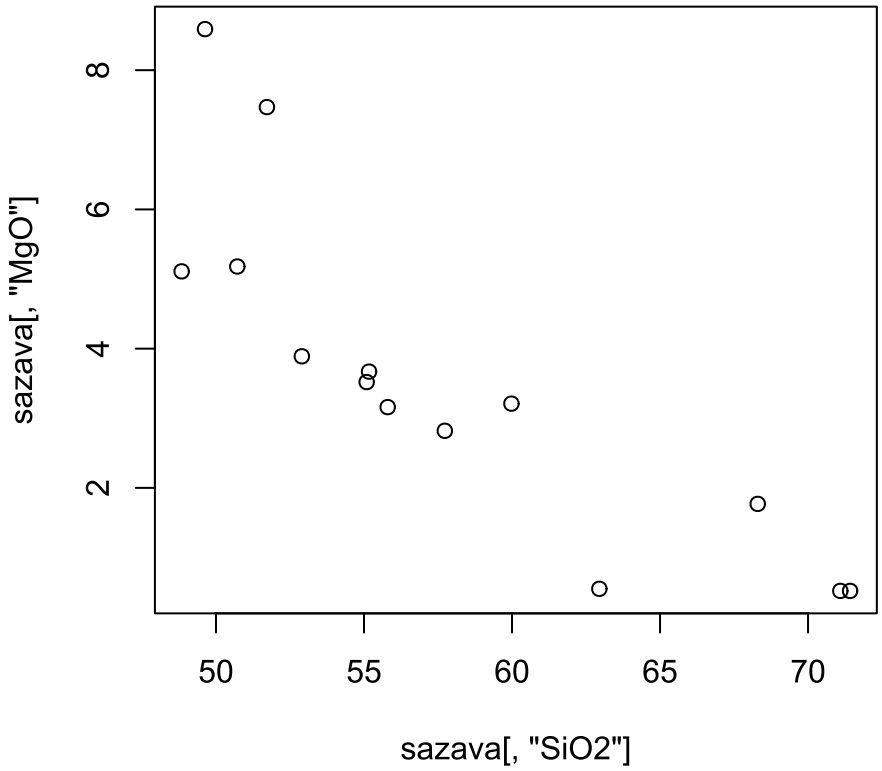
The following parameters influence the way how are plots rendered in the Jupyter nottebook (size, graphical format/mime type):

[2]:
```
options(repr.plot.width=10, repr.plot.height=10)      # Setting the size of␣
 ↪output in Jupyter notebook

options(jupyter.plot_mimetypes = "image/png")         # Change mimetype to PNG

# Using image/svg+xml
# May not work in some browsers (but is optimal for Chrome, Firefox...)
options(repr.plot.width=5, repr.plot.height=5)
options(jupyter.plot_mimetypes = "image/svg+xml")
```

[3]:
```
sazava  <-  read.table("data/sazava.data",sep="\t")
plot(sazava[,"SiO2"],sazava[,"MgO"])
```



1.1.2 Plotting symbols

When calling the function `plot`, and indeed many other graphical functions, a number of additional parameters can be specified to modify the appearance of the plot. Some are fairly universal (e.g., `col` or `pch`), but others are restricted or may behave unexpectedly. An overview of the most commonly used graphical parameters is given in Table below. If in doubt, the manual page of the particular plotting function should be consulted; also `?par` gives a useful information. Colours may be arranged into collections called palettes. The codes for available plotting symbols, standard colours and preview of palettes are shown in figures left and below, respectively.

1.1.3 Plotting colours

Standard colour codes correspond to `"black", "red", "green3", "blue", "cyan", "magenta", "yellow"` and `"white"`. Function `colors()` displays all the symbolic names available (657 of them!).

An overview of the main graphical parameters Parameter

Meaning

Explanation

adj

text justification relative to the coordinates [x,y]

0 – left, 1 – right, 0.5 – centered

asp

aspect ratio of the plot, y/x

asp = 1

axes

logical, should the axes be drawn?

axes = FALSE

bg

background colour

bg = “khaki”

cex

relative character size expansion (of text or symbols)

cex = 2

cex.main, cex.sub

size of plot’s title, subtitle

col

plotting colour

col = 0, col = “red”

las

style of axis labels

0 – parallel to the axes,1 – horizontal,2 – perpendicular, 3 – vertical

log

which of the axes is/are logarithmic?

log = “xy”,log = “ “

lty

line type (a number or text string)

1 – “solid”, 2 – “dashed”, 3 – “dotted”, 4 – “dotdash”

lwd

relative line width

lwd = 2

main

main title of the diagram (top)

main = “My diagram”

mar

outer margins of a plot in lines of text c(bottom, left, top, right)

mar = c(4,4,0,0)

mfcol=c(nr,nc), mfrow=c(nr,nc)

splits the plotting window into nr rows and nc columns, the graphs are filled by columns (mfcol) or rows (mfrow)

mfcol = c(2,1), mfrow = c(2,1)

pch

plotting character; a numeric code or a single alphanumeric symbol

pch = 7, pch = “Q”

pos

position of the text relative to the coordinates [x,y]

1 – below, 2 – left, 3 – above, 4 – right

pty

type of plot region to be used

“s” – square, “m” – maximal

srt

rotation of the text in degrees

srt = 90

sub

subtitle of the diagram (bottom)

sub = “for thesis”

type

type of the diagram

“p” – points, “l” – lines, “b” – both, “o” – overplot, “n” – none (no plotting)

xaxs, yaxs

scaling style for axes (default is extended: plotted symbols cannot crash with axes)

“r” – extended range, “i” – exact scaling

xlab, ylab

labels for x and y axes

xlab = “SiO2[wt.%]”

xlim, ylim

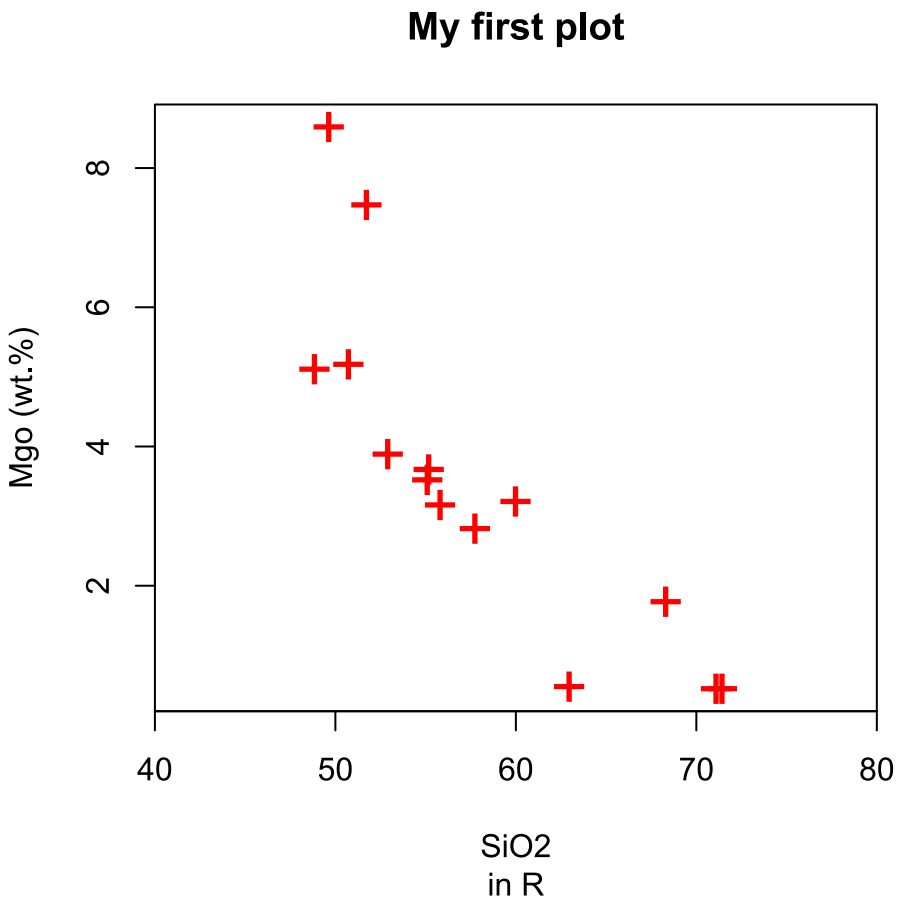
limits of the x and y axes

xlim=c(50,70)

If you want to get help on all available plotting parameters, type `?par`.

[4]:
```
options(repr.plot.width=5, repr.plot.height=5)
```

[5]:
```
sazava  <-  read.table("data/sazava.data",sep="\t")
plot(sazava[,"SiO2"],sazava[,"MgO"],axes=TRUE,main="My first plot",sub="in␣
 ↪R",xaxs="i",col="red",
    pch="+",cex=2,xlab="SiO2",ylab="Mgo (wt.%)",xlim=c(40,80))
```



An overview of the most useful low-level graphical functions in R Function (basic use)

Explanation

points(x,y,type="p")

adds extra data points at [x,y]

text(x,y,labels)

adds text specified by labels at [x,y]

mtext(text,side)

places text at margins, outside the plotting region on side (1 = bottom, 2 = left, 3 = top, 4 = right)

contour(x,y,z,add=TRUE)

contour plot (x and y specify the regular grid, z the values)

lines(x,y,type="l")

joins the points with straight line segments

curve(expr,add=TRUE)

adds a curve specified by expr (a function of x)

arrows(x0,y0,x1,y1)

arrows from [x0,y0] to [x1,y1]

abline(a,b)

straight line defined by intercept (a) and slope (b)

grid(nx,ny=nx)

grid with nx cells horizontally and ny vertically

rect(xleft,ybottom,xright,ytop)

rectangle given left, right, bottom, and top limits

polygon(x,y)

polygons whose vertices are given in x and y

axis(side,at,labels)

custom axis; side = 1 for x, 2 for y; at = values to be annotated by labels

box(which)

box around the plotting region (which = "plot") or "figure", "inner", "outer"

legend(x,y,legend,lty,lwd,pch)

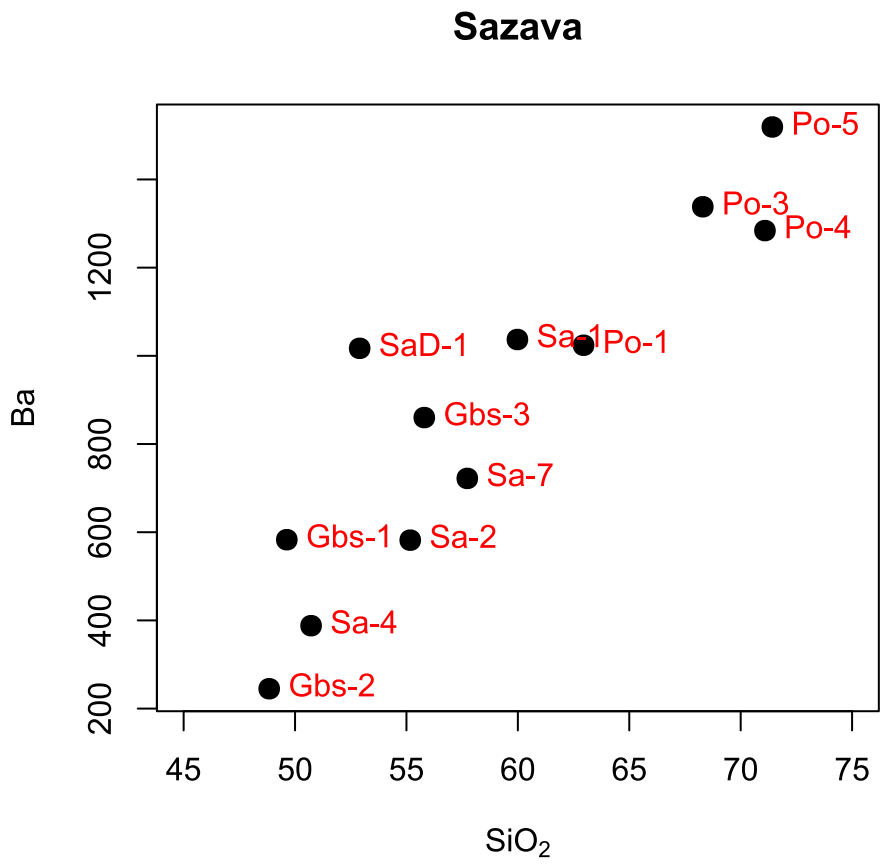
adds a legend at the point [x,y]

title(main,sub,xlab,ylab)

main title/subtitle and/or axes labels to the plot

text(x, y, labels) This low-level function displays the given text at coordinates `[x,y]`. It is especially useful to label individual data points of the binary diagrams (in combination with the optional `pos` parameter).

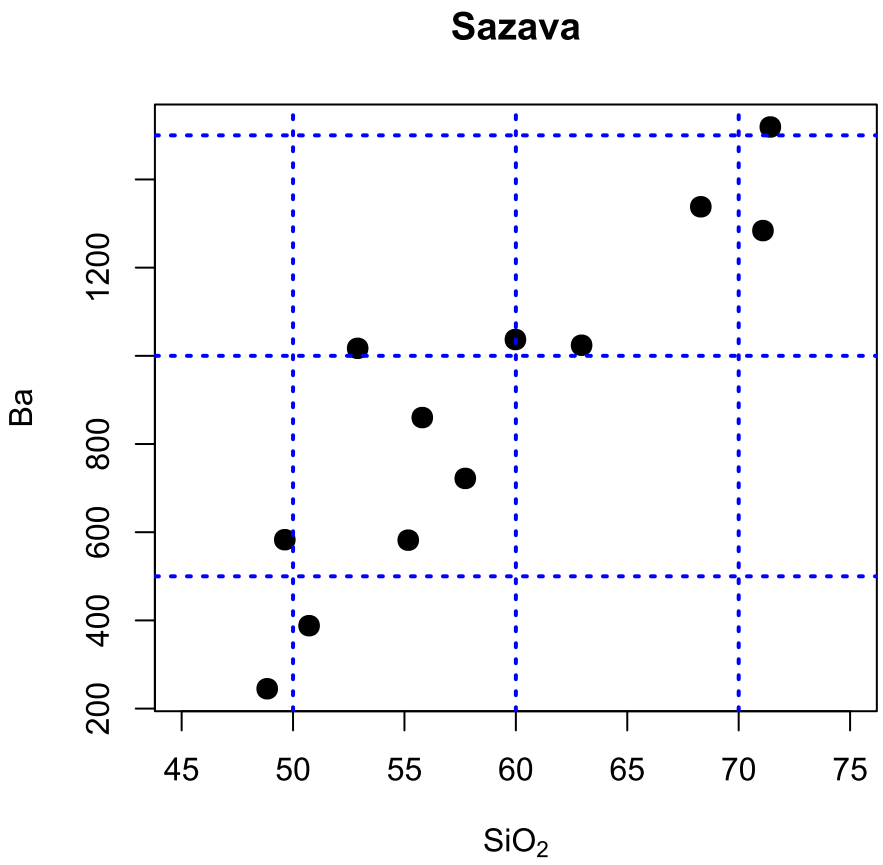
[6]:
```
plot(sazava[,"SiO2"],sazava[,"Ba"],xlab=expression(SiO[2]),cex=1.
↪5,ylab="Ba",pch=16,main="Sazava",xlim=c(45,75))
text(sazava[,"SiO2"],sazava[,"Ba"],rownames(sazava),pos=4,col="red")
```



abline(a = NULL, b = NULL, h = NULL) This command is used to draw a straight line. For instance:

```
> abline(a,b)                  # slope b, intercept a
> abline(h=y)                  # horizontal line(s)
> abline(v=1:5,lty="dotted")   # vertical dotted grid lines
```

[7]:
```
plot(sazava[,"SiO2"],sazava[,"Ba"],xlab=expression(SiO[2]),cex=1.
  ↪5,ylab="Ba",pch=16,main="Sazava",xlim=c(45,75))
abline(h=seq(0,1500,500),lty="dotted",col="blue",lwd=2)
abline(v=seq(40,80,10),lty="dotted",col="blue",lwd=2)
```



grid(nx, ny) A convenience function creating grids with `nx` and `ny` specifying the number of cells of the grid in x and y directions.

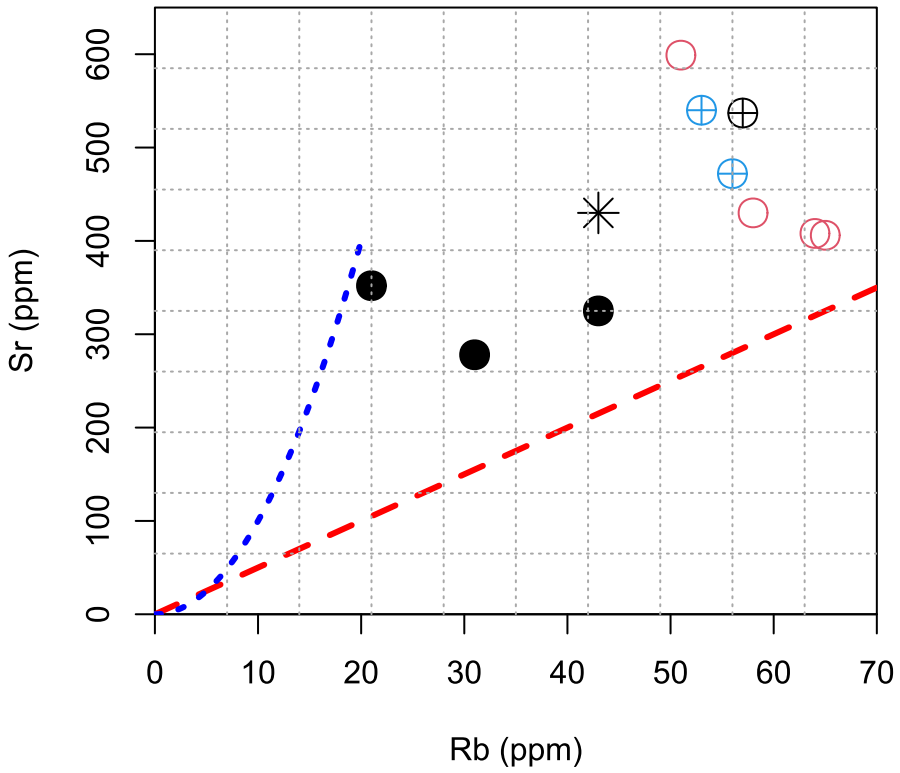
curve(expr, from = NULL, to = NULL, add = FALSE) A function for adding a curve (if `add = TRUE`) specified by an expression (`expr`; written as a function of x). Optionally, the range of the x axis can be also set, using the parameters `from` and `to`.

For example:

[8]:
```
options(repr.plot.width=5, repr.plot.height=5)
```

[9]:
```
plot(sazava[,"Rb"],sazava[,"Sr"],xlab="Rb (ppm)",ylab="Sr␣
↪(ppm)",pch=sazava[,"Symbol"],
     col=sazava[,"Colour"],cex=2,xlim=c(0,70),ylim=c(0,650),xaxs="i",yaxs="i")
abline(0,5, col="red",lwd=3,lty="dashed")
curve(x^2,add=TRUE,col="blue",lwd=3,lty="dotted",from=0,to=20)
```

```
grid(10,10,col="darkgray",lwd=1,lty="dotted")
```



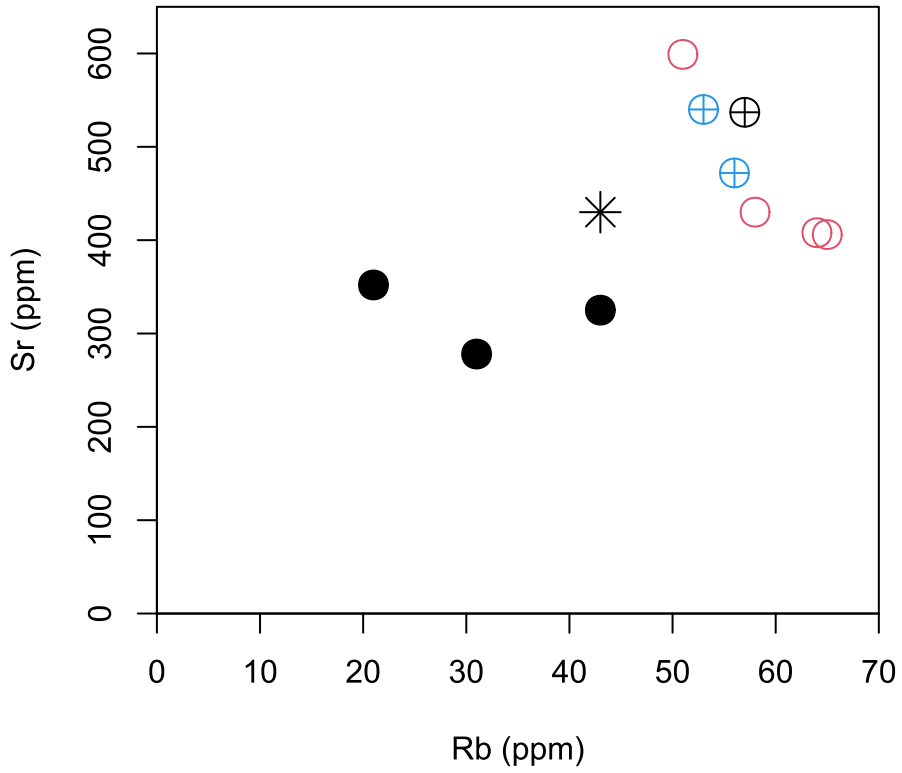
lm(formula) A function for fitting linear models. The simplest form of formula is `y~x`, i.e. y as a function of x, performing a linear regression. To see details of the object generated, use the function `summary`. Note that `abline` has a method for plotting thus generated linear fits.

Continuing our previous example, we can also produce a linear fit, and assign the result to an arbitrary variable `ee`:

[10]:
```
plot(sazava[,"Rb"],sazava[,"Sr"],xlab="Rb (ppm)",ylab="Sr␣
 ↪(ppm)",pch=sazava[,"Symbol"],
    col=sazava[,"Colour"],cex=2,xlim=c(0,70),ylim=c(0,650),xaxs="i",yaxs="i")
ee <- lm(sazava[,"Sr"]~sazava[,"Rb"])
ee
```

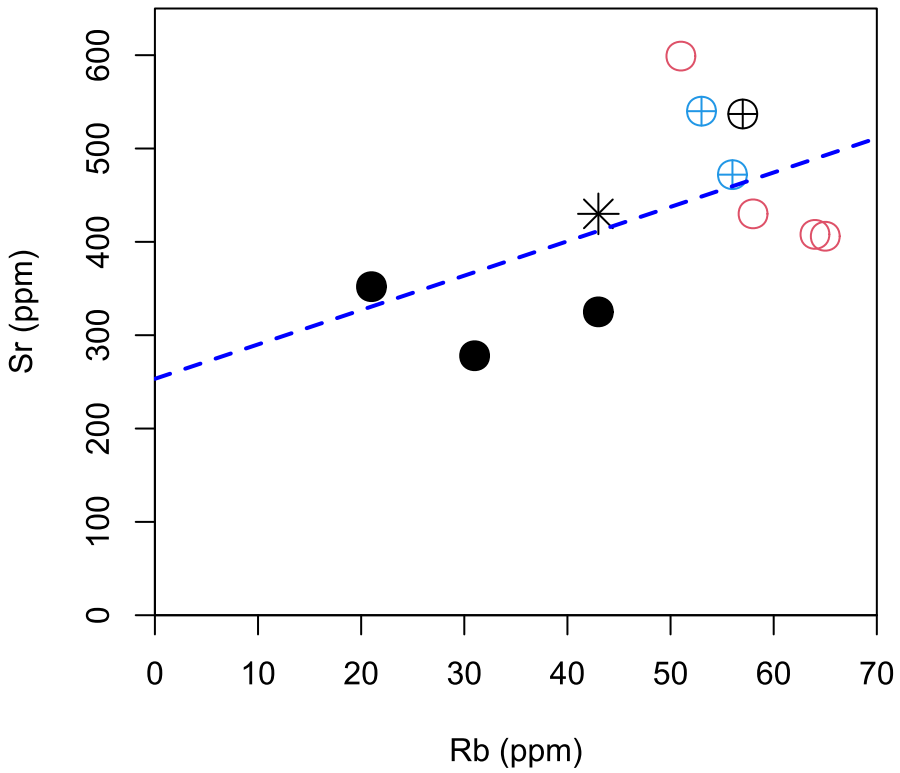
```
Call:
lm(formula = sazava[, "Sr"] ~ sazava[, "Rb"])
```

```
Coefficients:
   (Intercept)  sazava[, "Rb"]
       253.271           3.684
```



and the object `ee` is ready to be plotted by the function `abline`:

[11]:
```
plot(sazava[,"Rb"],sazava[,"Sr"],xlab="Rb (ppm)",ylab="Sr␣
 ↪(ppm)",pch=sazava[,"Symbol"],col=sazava[,"Colour"],cex=2,xlim=c(0,70),ylim=c(0,650),xaxs="i
ee <- lm(sazava[,"Sr"]~sazava[,"Rb"])
abline(ee,lwd=2,lty="dashed",col="blue")
```



If we want to know the details, we can display the whole list `ee` using:

[12]:
```
summary(ee)
```

```
Call:
lm(formula = sazava[, "Sr"] ~ sazava[, "Rb"])

Residuals:
    Min      1Q  Median      3Q     Max
-89.477 -82.459   9.081  34.457 157.842

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)    253.271     89.448   2.831   0.0178 *
sazava[, "Rb"]   3.684      1.672   2.204   0.0521 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
Residual standard error: 84.07 on 10 degrees of freedom
  (2 observations deleted due to missingness)
Multiple R-squared:  0.3269,        Adjusted R-squared:  0.2596
F-statistic: 4.857 on 1 and 10 DF,  p-value: 0.05211
```

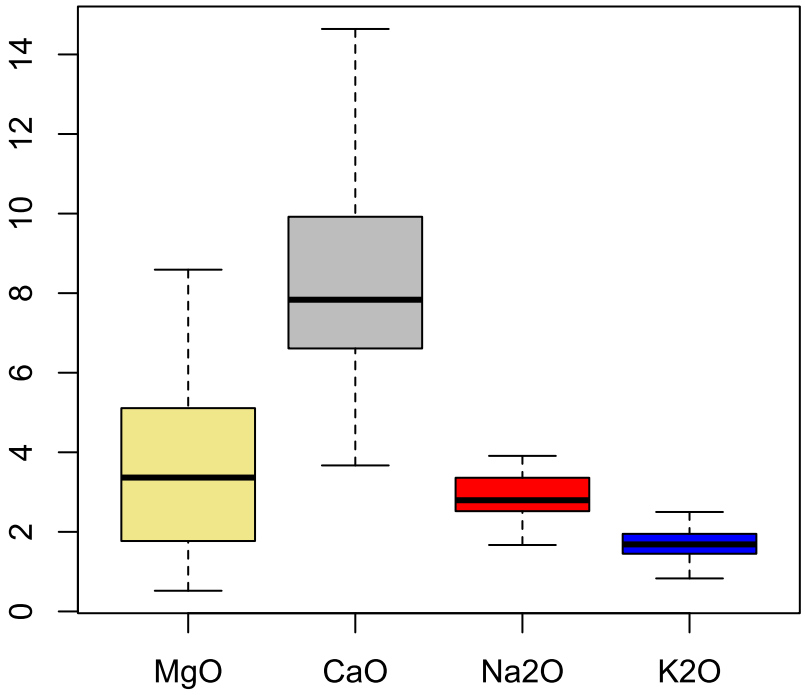
points(x, y, type = "p") Function `points` adds new data points with coordinates `[x,y]` to an existing plot. The argument `type` controls how they are displayed (as points, lines, etc.). #### lines(x, y) Draws straight line segments; `x` and `y` are vectors of corresponding coordinates. #### legend(x, y = NULL, legend, fill = NULL, col, pch, lty, lwd, inset = 0) Adds a legend at `[x,y]`. If `y = NULL`, the position (`x`) can be specified by a single keyword such as `"bottomright"` or `"center"`. The explanatory text is given by a character vector `legend`; the attributes for the symbols can be `fill` (colours to fill boxes), `pch` (plotting characters), `col` (colours of plotting characters or lines), `lty` (line types) or `lwd` (line widths). The numeric argument `inset` defines a distance from the margins as a fraction of the plot region size.

1.2 2.2 Additional high-level plotting functions

Here we show examples of the most useful of the other high-level functions for plotting boxplots, correlation matrices, histograms and such alike (see a Table above). #### boxplot(x) Creates a "box-and-whiskers" plot, i.e. a diagram, in which each variable (column of a data frame/matrix `x`) is represented by a rectangle (its horizontal sides correspond to the 1st and 3rd quartiles, a horizontal line denotes a median). Two vertical lines join thr extremes (minimum and maximum); outliers, if any, are plotted as tiny circles.

[13]:
```
sazava <- read.table("data/sazava.data",sep="\t")
oxides <- c("MgO","CaO","Na2O","K2O")
boxplot(sazava[,oxides],col=c("khaki","gray","red","blue"))
summary(sazava[,oxides])
```

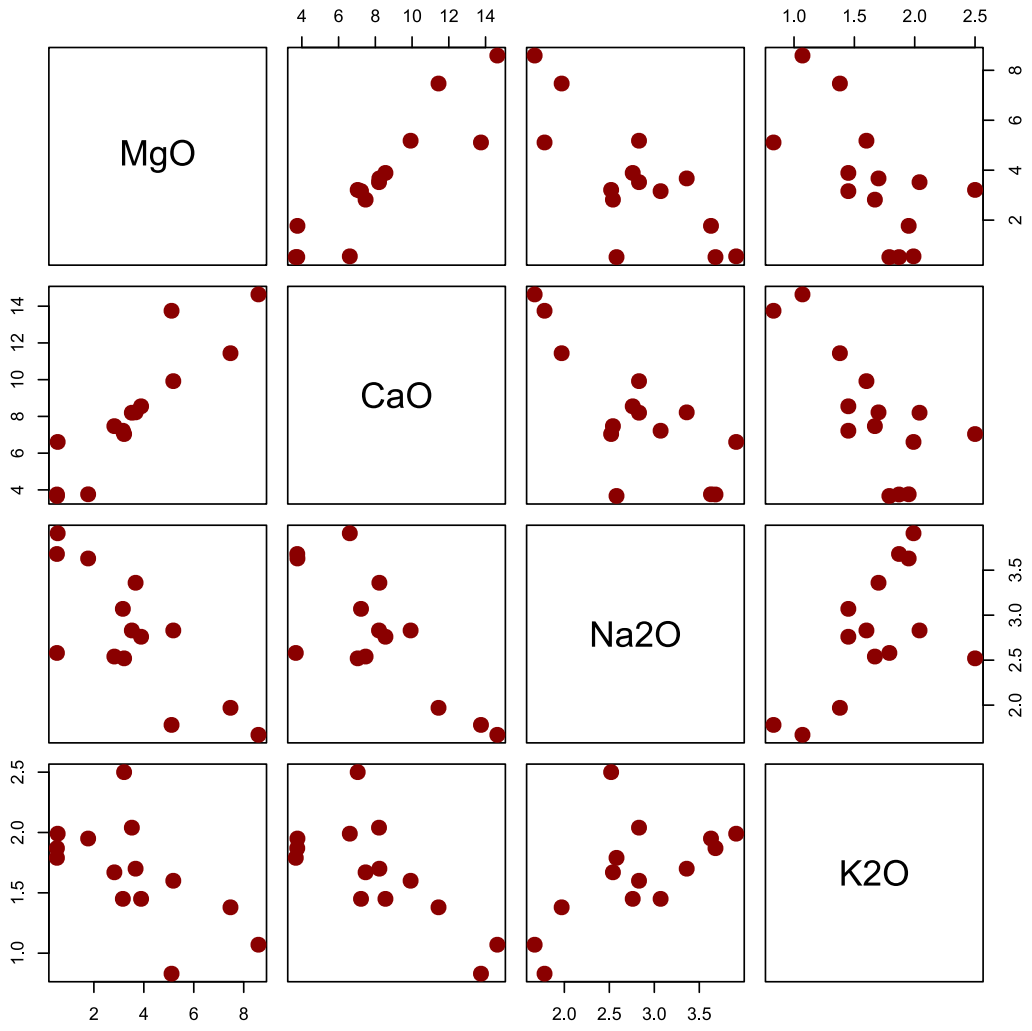
```
      MgO              CaO             Na2O            K2O        
 Min.   :0.520   Min.   : 3.670   Min.   :1.670   Min.   :0.830  
 1st Qu.:2.033   1st Qu.: 6.718   1st Qu.:2.525   1st Qu.:1.450  
 Median :3.365   Median : 7.835   Median :2.795   Median :1.685  
 Mean   :3.570   Mean   : 8.160   Mean   :2.795   Mean   :1.664  
 3rd Qu.:4.805   3rd Qu.: 9.578   3rd Qu.:3.288   3rd Qu.:1.930  
 Max.   :8.590   Max.   :14.640   Max.   :3.910   Max.   :2.500  
```



pairs(x) A scatterplot matrix for all possible combinations of columns in matrix `x`:

[14]:
```
options(repr.plot.width=7, repr.plot.height=7)
```

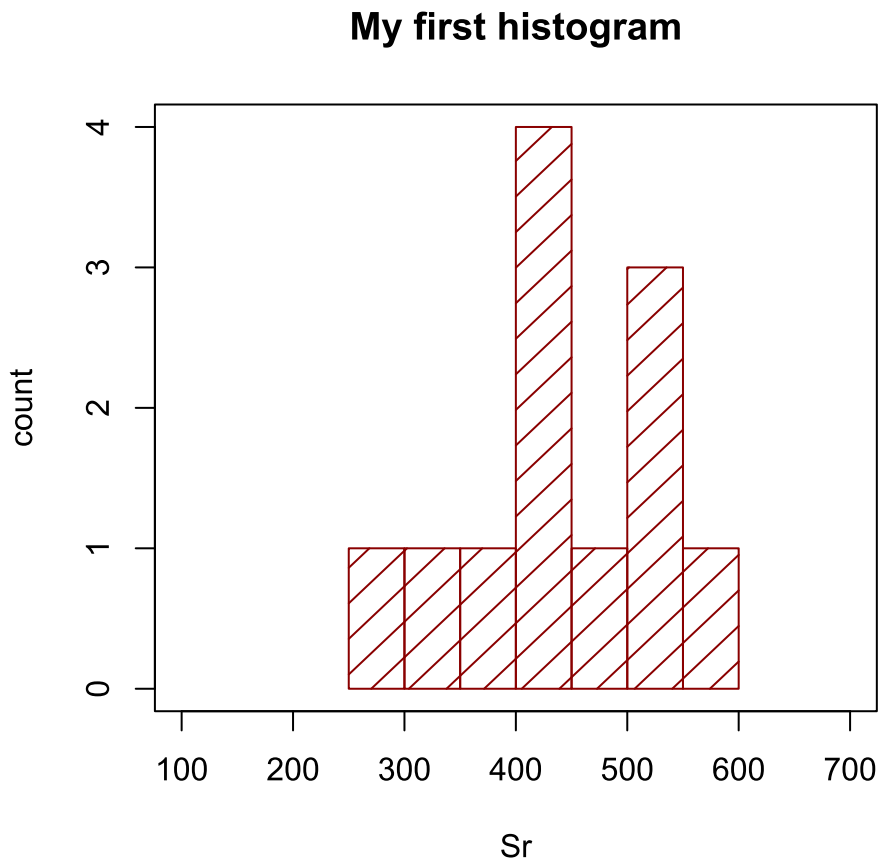
[15]:
```
sazava <- read.table("data/sazava.data",sep="\t")
oxides <- c("MgO","CaO","Na2O","K2O")
pairs(sazava[,oxides],pch=16,col="darkred",cex=2)
```



hist(x) Produces a histogram of frequencies of the vector `x`:

[16]:
```
options(repr.plot.width=5, repr.plot.height=5)
```

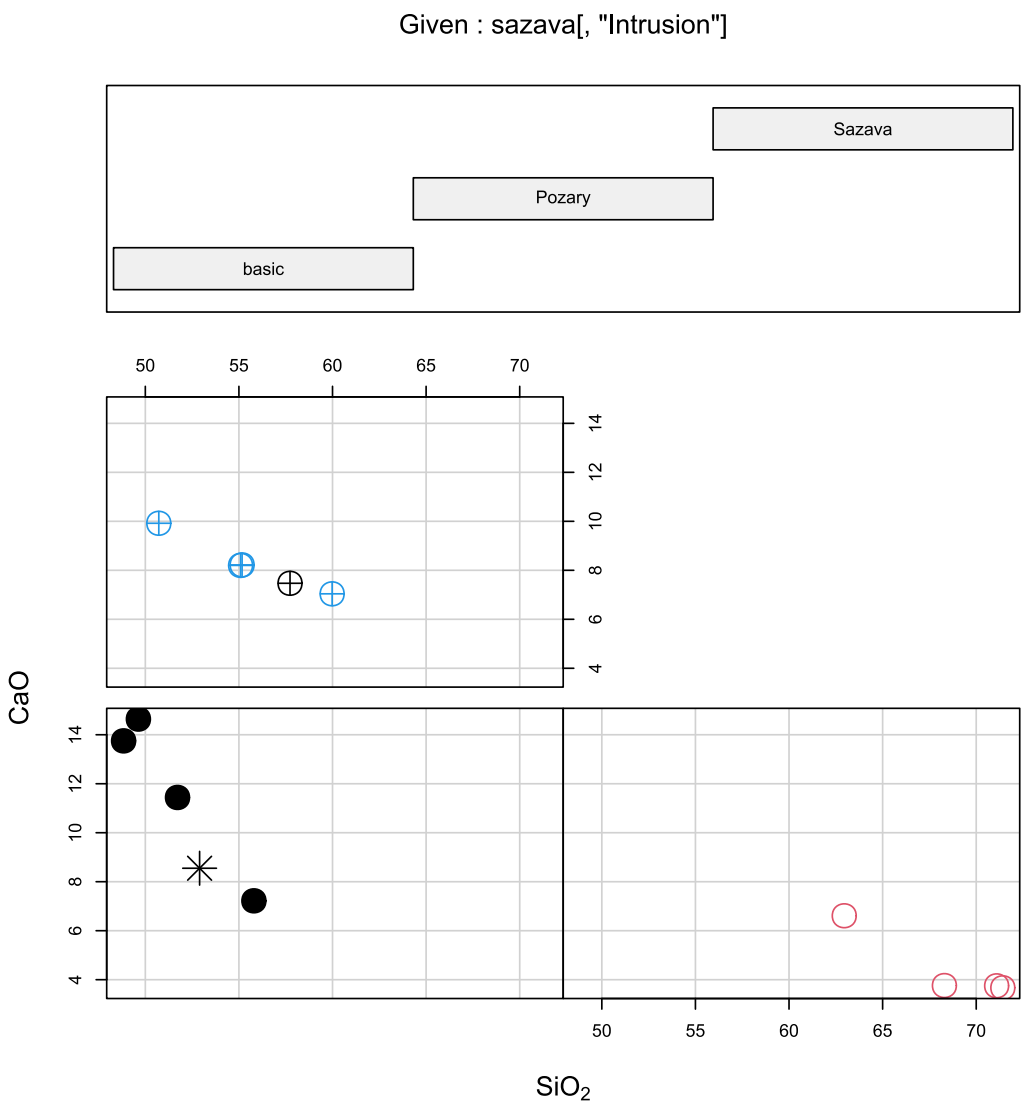
[17]:
```
sazava <- read.table("data/sazava.data",sep="\t")
hist(sazava[,"Sr"],xlab="Sr",ylab="count",xlim=c(100,700),col="darkred",density=5,angle=45,mai
  ↪first histogram")
box()
```



coplot(y~x|z) Conditioning plot; in this form it displays a set of bivariate plots of `x` vs. `y` for each level of the factor `z`.

[18]:
```
options(repr.plot.width=7, repr.plot.height=7)
```

[19]:
```
sazava <- read.table("data/sazava.data",sep="\t")
coplot(sazava[,"CaO"]~ sazava[,"SiO2"]|␣
 ↪sazava[,"Intrusion"],cex=3,xlab=expression(SiO[2]),
      ylab="CaO",pch=sazava[,"Symbol"],col=sazava[,"Colour"])
```



contour(z, add = FALSE) Creates a contour plot, or adds contour lines to a pre-existing plot (if `add = TRUE`) based on a data matrix `z`. The data must be prepared beforehand in a regular grid as the function does not perform any interpolation such as kriging. In this form, both coordinates are normalized from 0 to 1. The real ones can be provided by the optional arguments `x` and `y`.

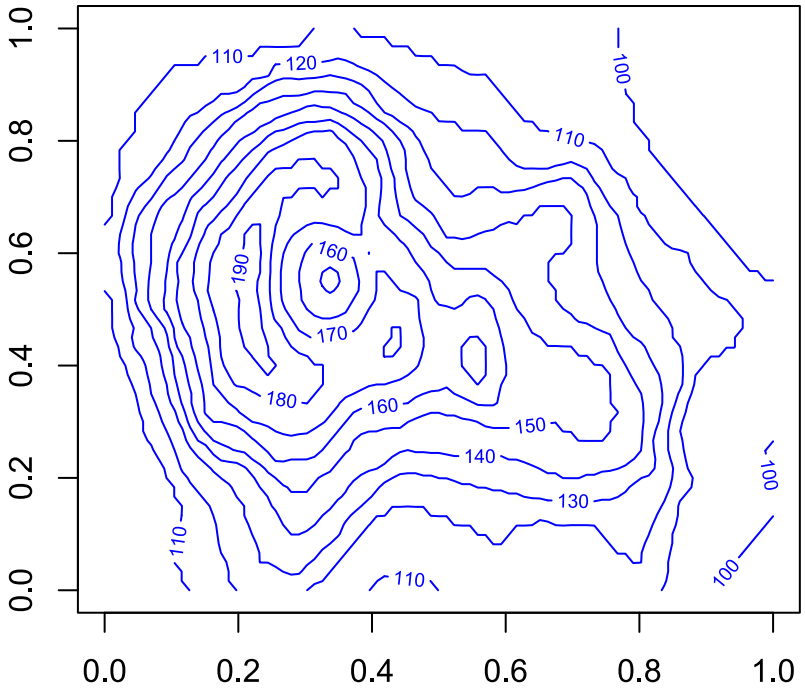
A simple use will be demonstrated on isohypses of the Maunga Whau volcano, New Zealand, as given in the `volcano` dataset (see `?volcano`):

[20]:
```
options(repr.plot.width=5, repr.plot.height=5)
```

[21]:
```
data(volcano)
contour(volcano,col="blue")
head(volcano)
```

A matrix: 6 × 61 of type dbl

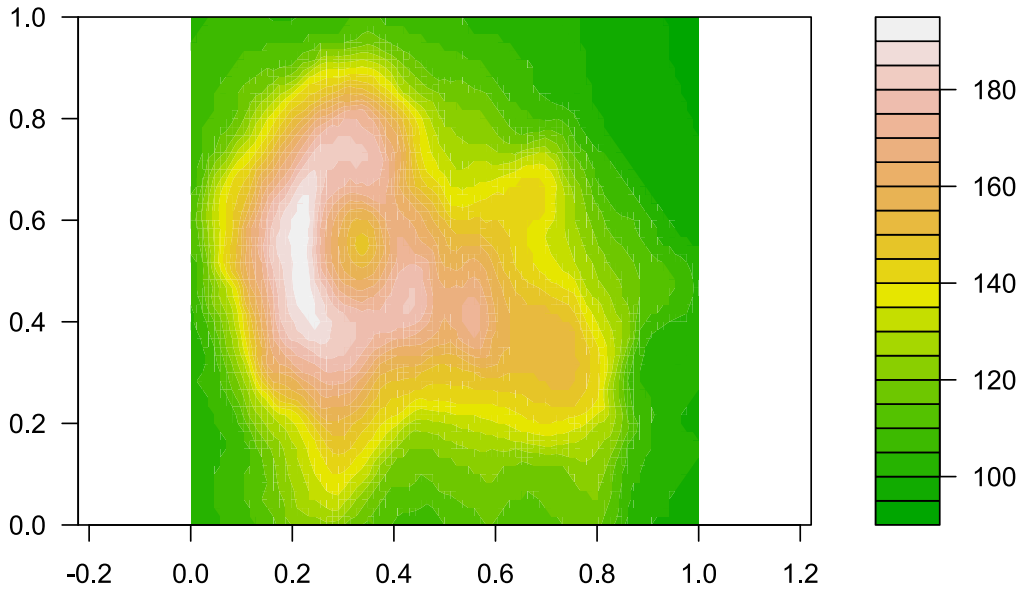
100	100	101	101	101	101	101	100	100	100	...	107	107	107	10
101	101	102	102	102	102	102	101	101	101	...	108	108	107	10
102	102	103	103	103	103	103	102	102	102	...	109	108	108	10
103	103	104	104	104	104	104	103	103	103	...	109	109	108	10
104	104	105	105	105	105	105	104	104	103	...	110	109	109	10
105	105	105	106	106	106	106	105	105	104	...	110	110	109	10



filled.contour(z, color.palette = cm.colors) Creates a filled contour plot, in which the values of `z` are represented by individual colours from the `color.palette`. Here is another example using the `volcano` dataset:

[22]:
```
options(repr.plot.width=7, repr.plot.height=5)
```

[23]:
```
data(volcano)
filled.contour(volcano,color.palette=terrain.colors,asp=1)
```



palette(value) This function serves to view or manipulate a colour palette. The parameter `value` is the name of palette predefined in standard R, followed by the desired number of colours given in brackets. For instance:

```
> ee <- palette(heat.colors(5))
> ee
[1] "red"     "#FF5500" "#FFAA00" "yellow"  "#FFFF80"
```

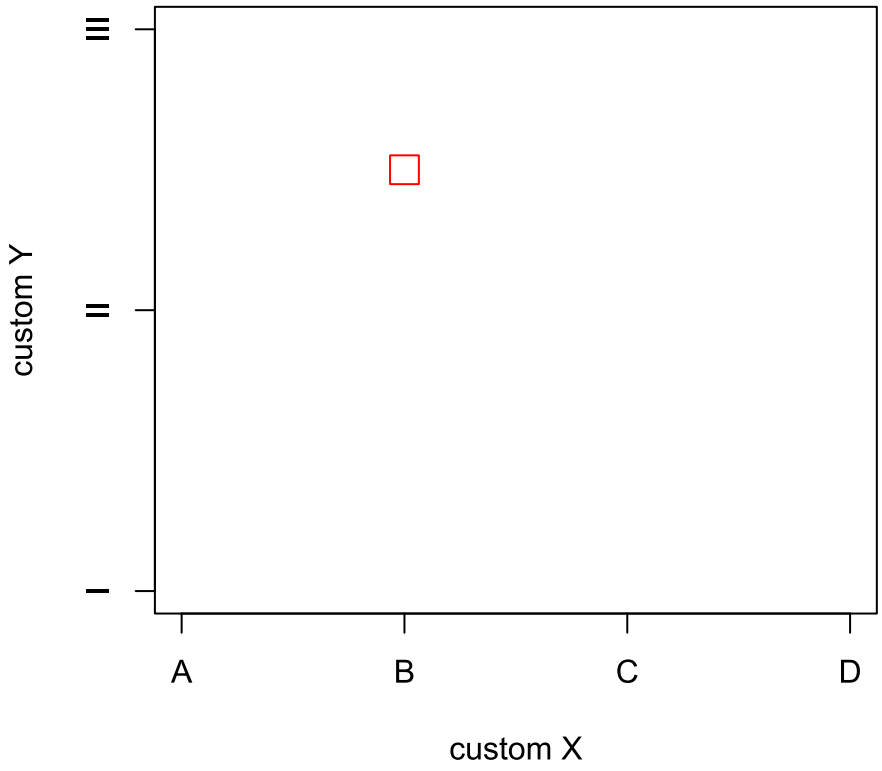
Clearly, the numeric representation above is hexadecimal.

1.3 2.3 Creating custom layouts and axes

In order to gain more control of the plotting window, one should use the graphical parameters function `par` to create multifigure plots and the `axis` function to fine-tune style and placement of axes. #### par(mfrow = c(nrow, ncol)) #### par(mfcol = c(nrow, ncol)) Creates multifigure layouts by splitting the graphical window into a matrix of `nrow` × `ncol` plotting regions to be sequentially filled by plotted graphs (row-wise — `mfrow` or column-wise — `mfcol`). Arbitrary sized plotting regions can be configured using the `fig` option to `par`. #### axis (side, at, labels) The `plot` function can be called with a parameter `axes = FALSE` such that no axes are drawn. This is a prelude to the command `axis`, defining a custom axes layout. The arguments are `side` = 1 for bottom (x), 2 left (y), 3 top, 4 right; `at` – a vector with values to be labelled; `labels` – character vector with the textual labels. For example:

[24]:
```
options(repr.plot.width=5, repr.plot.height=5)
```

[25]:
```
plot(1,1,xlim=c(0,3),ylim=c(-1,1),axes=FALSE,xlab="custom X",ylab="custom␣
↪Y",type="n")
axis(1,0:3,c("A","B","C","D"))
axis(2,-1:1,c("I","II","III"))
box()
points(1,0.5,col="red",pch=0,cex=2)
```



1.4 2.4 Exporting graphs from R; graphical devices

Graphs can be exported to a word processor, a desktop publishing (DTP) or a graphical package (e.g. Adobe Illustrator or CorelDraw!) for further editing. They can be copied to the Clipboard or saved into a file by right-clicking the graphical window and selecting `Save as...`.

Alternatively, corresponding items in the menu `File` of the graphical window can be invoked. There is a wealth of formats to choose from, including the most popular vector (PostScript, PDF, WMF) and bitmap (TIF, PNG, BMP, JPG) ones. Of course, for further editing or publishing, vector

formats are to be preferred.

NB that PostScript and the PDF files are generated in a quality superior to the Windows Metafiles (WMF) format.

Bitmaps (raster graphics)

- Formed by individual pixels
- Each is assigned a colour (or grey shade)
- Produced typically by cameras or scanners

Vector graphics

- Consists of objects that can be described mathematically
- Always newly recalculated
- Attributes: ink, fill colours, line thickness…
- Good for high-quality printing, output to plotters – technical illustrations, graphs, maps etc.
- Inappropriate for photographs

Advantages of the vector graphics

- Saves memory,
- Produces small files,
- Is always printed at the maximum resolution of the given output device,
- Easy to transform (magnification, translation, mirror, rotation…),
- Easy to change the attributes of individual objects (colour, line type, line thickness…),
- Text can be edited and searched, proofreading tools can be used etc.

Selected devices available in R As a useful alternative, the graphical output can be redirected to one of the many supported graphical devices:

Function

Description

Graphics type

windows()

a graphical window (Windows)

–

quartz()

a graphical window (Mac OS X)

–

x11()

a graphical window (Linux)

–

postscript(file)

PostScript (see also ?ps.options)

Vector

pdf(file)

Adobe PDF (Portable Document Format) (see also ?pdf.options)

Vector

win.metafile(filename)

Windows Metafile (WMF)

Vector

png(filename)

bitmap (lossless compression, less common))

Raster

tiff(filename)

bitmap (lossless uncompressed, widely accepted)

Raster

jpeg(filename)

bitmap (lossy compression, small files)

Raster

See `?grDevices` for details and complete list.

dev.off() Close the current graphical window. #### graphics.off() Close all the opened graphical windows.

1.5 2.5 Interaction with plots

The ability to interact with graphics makes it possible, for instance, to select outliers and label them with sample names. Or one can pick samples for further processing, such as setting end members for numerical modelling. Clearly these functions are only useful for interactive plotting devices. #### locator() The `locator` returns the coordinates of one or more points clicked on by the left mouse button. Identification is stopped by pressing the right mouse button and selecting `Stop`. #### identify(x, y, labels) This function annotates the plot with `labels` for each given coordinate pair `[x,y]`. Usually only useful when there are a small number of data points.

```
> sazava<-read.table("data/sazava.data",sep="\t")
> plot(sazava[,"Rb"],sazava[,"Sr"],xlab="Rb",ylab="Sr",pch=sazava[,"Symbol"],cex=2,xlim=c(0,70)
> identify(sazava[,"Rb"],sazava[,"Sr"],rownames(sazava))
> locator()
```