Introductory course on the R software

P. Lafaye de Micheaux[1]

[1]Mathematics and Statistics Departement
Université de Montréal
CANADA

New South Wales University
Sydney, Australia
June 24, 2014

https://biostatisticien.eu/springeR/courseRw6.pdf

Goals of today lecture

Describing the instructions for

- manipulating graphics windows ;
- drawing simple plots ;
- dealing with colors ;
- adding text on graphs ;
- improving titles, axes and captions ;
- interacting with plots ;
- fine-tuning graphical parameters.

Basic stuff

To open a new graphical window, use the instructions `windows()` or `win.graph()` (for MacOS, use `quartz()`).

To close some graphical device, use : `dev.off(`*device-number*`)`.

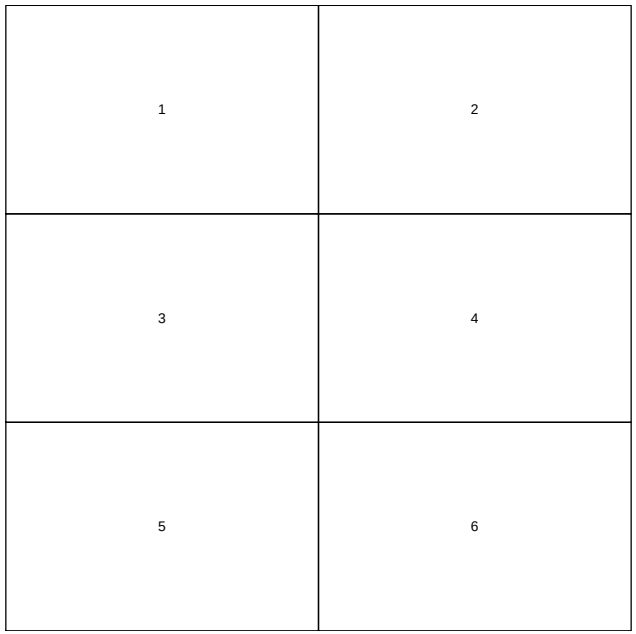
To save a plot that has already been drawn, use : `savePlot(filename="Rplot",type="pdf")`.

You can also proceed as follows :

```
> pdf(file="mygraph.pdf")
> hist(runif(100))
> dev.off()
```

Splitting the Graphics Window

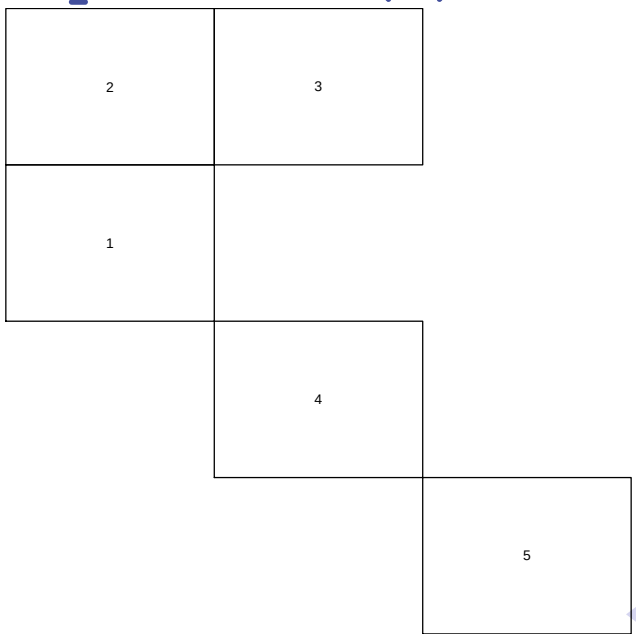
```
> par(mfrow=c(3,2))
```




```
> par(mfcol=c(3,2)) # by columns.
```

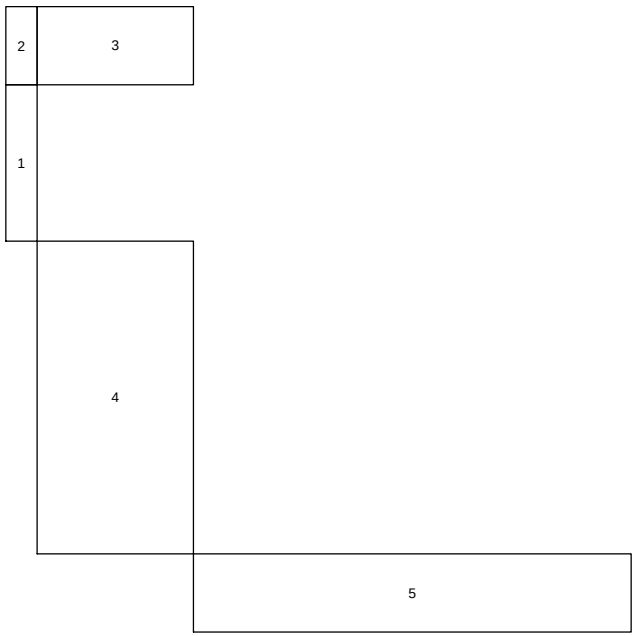
Splitting the Graphics Window

```
> (mat <- matrix(c(2,3,0,1,0,0,0,4,0,0,0,5),4,3,byrow=T))
     [,1] [,2] [,3]
[1,]    2    3    0
[2,]    1    0    0
[3,]    0    4    0
[4,]    0    0    5
> layout(mat) ; layout.show(5)
```



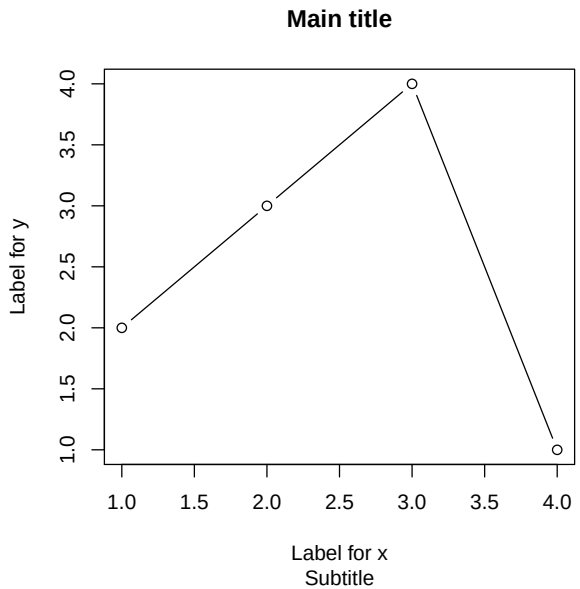
Splitting the Graphics Window

```
> layout(mat,widths=c(1,5,14),heights=c(1,2,4,1))
> layout.show(5)
```



The functions `plot()` and `points()`

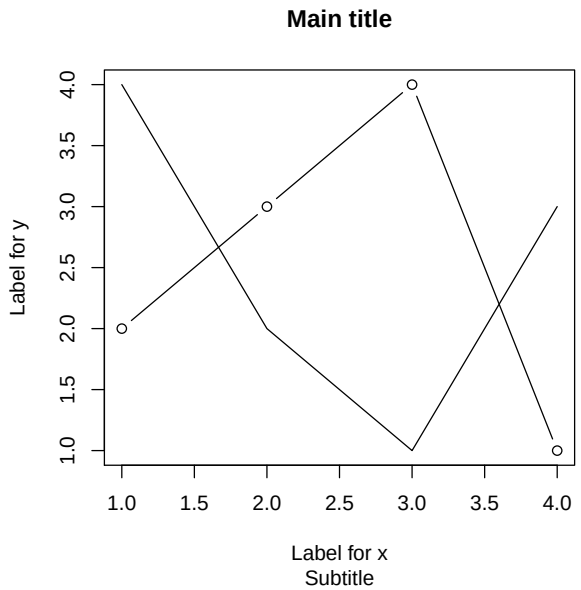
```
> plot(1:4,c(2,3,4,1),type="b",main="Main title",
+   sub="Subtitle",xlab="Label for x",ylab="Label for y")
```



Argument	Description
`x`	Vector of *x* coordinates of points to draw.
`y`	Vector of *y* coordinates of points to draw.
`type`	Specify the type of plotting : `"p"` for points, `"l"` for lines, `"b"` for both, `"c"` for empty points joined by lines, `"o"` for overplotted points and lines, `"h"` for vertical lines, `"s"` for stair steps and `"n"` to plot nothing (but to display the window, with axes).
`main`	Specify the main title.
`sub`	Specify the subtitle.
`xlab`	Specify the label of the *x* axis.
`ylab`	Specify the label of the *y* axis.
`xlim`	Vector of length 2. Specify the lower and upper bound for the *x* axis.
`ylim`	Vector of length 2. Specify the lower and upper bound for the *y* axis.
`log`	Character string which contains `"x"` (respectively `"y"`, `"xy"` or `"yx"`) if the *x* axis (respectively the *y* axis, both) is to be logarithmic.

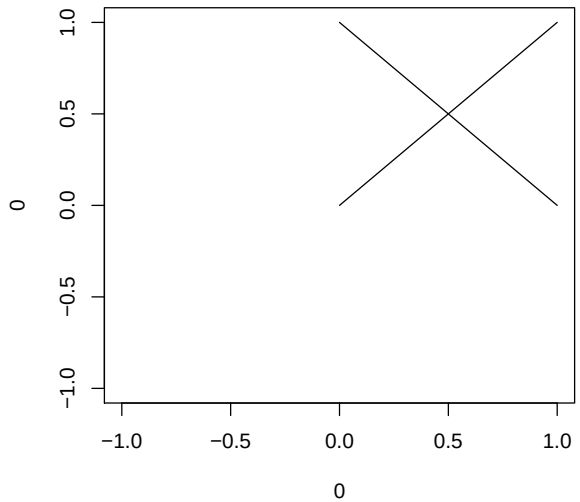
Note that successive calls of the function `plot()` create a new plot every time, which replaces the previous one (unless the graphics window has been split, as explained above). The function `points()` can remediate this issue by overlaying the new plot on top of the old one. It takes the same arguments as `plot()`.

```
> points(1:4,c(4,2,1,3),type="l")
```



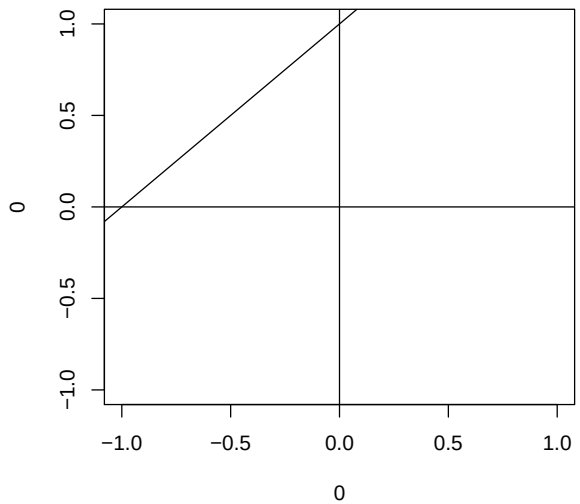
The functions segments(), lines() and abline()

```
> plot(0,0,"n")
> segments(x0=0,y0=0,x1=1,y1=1)
> lines(x=c(1,0),y=c(0,1))
```



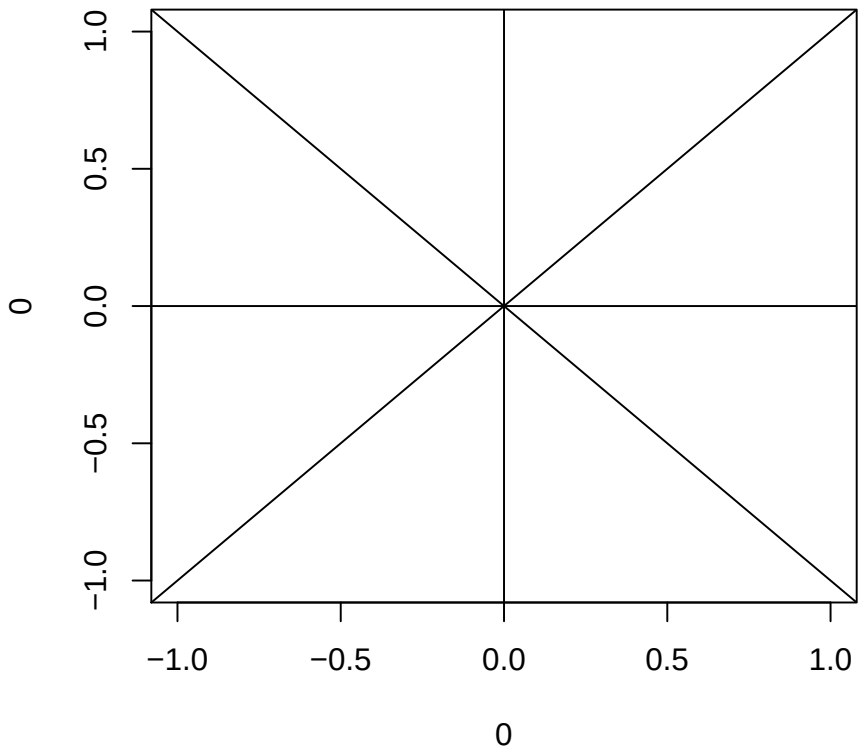
The function `abline()` is used to draw a straight line of equation $y = a + bx$ (specified by the arguments `a` and `b`), or a horizontal (argument `h`) or vertical (argument `v`) line.

```
> plot(0,0,"n");abline(h=0,v=0);abline(a=1,b=1)
```



Do it yourself

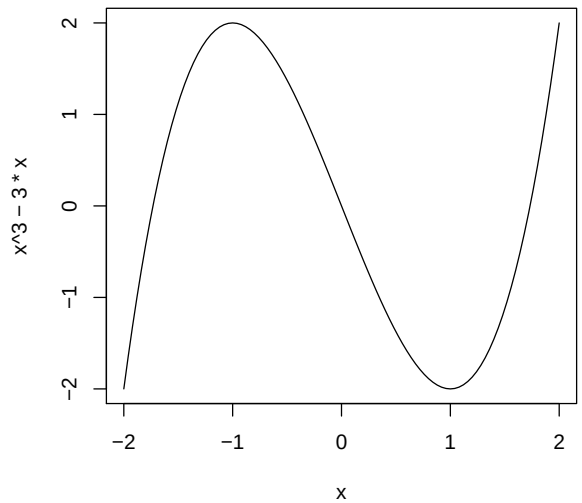
Reproduce this plot.



The function curve()

This function is used to draw a curve in a Cartesian coordinate system, on the interval specified by the bounds `from` and `to`.

```
> curve(x^3-3*x,from=-2,to=2)
```



The function colors()

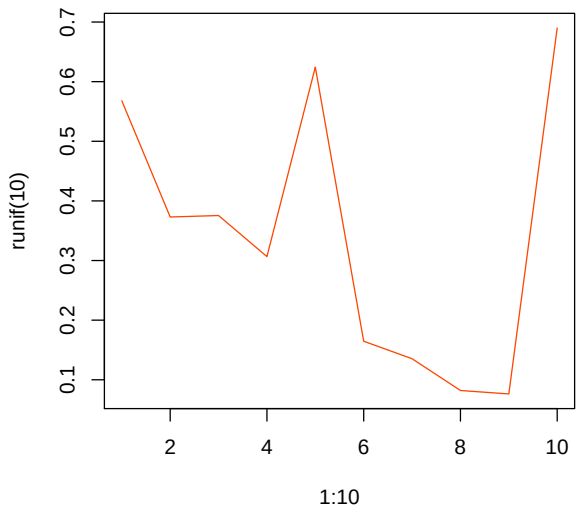
This function returns the names of the 657 colours known to R.

If you want to get the different shades of orange, you can use the instruction :

```
> colors()[grep("orange",colors())]
 [1] "darkorange"  "darkorange1" "darkorange2" "darkorange3"
 [5] "darkorange4" "orange"      "orange1"     "orange2"
 [9] "orange3"     "orange4"     "orangered"   "orangered1"
[13] "orangered2"  "orangered3"  "orangered4"
```

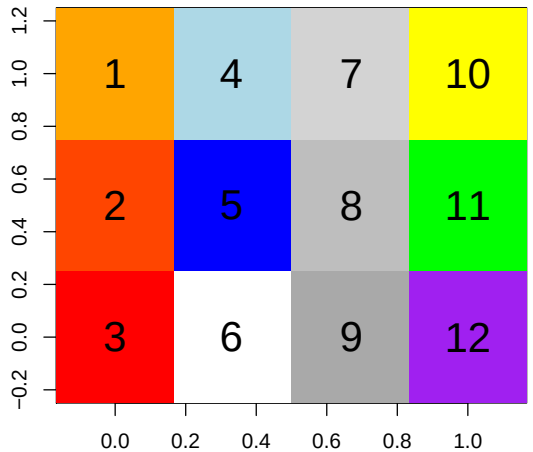
These colours can be used in your plots, for example with the argument `col` of the function `plot()`.

```
> plot(1:10,runif(10),type="l",col="orangered")
```



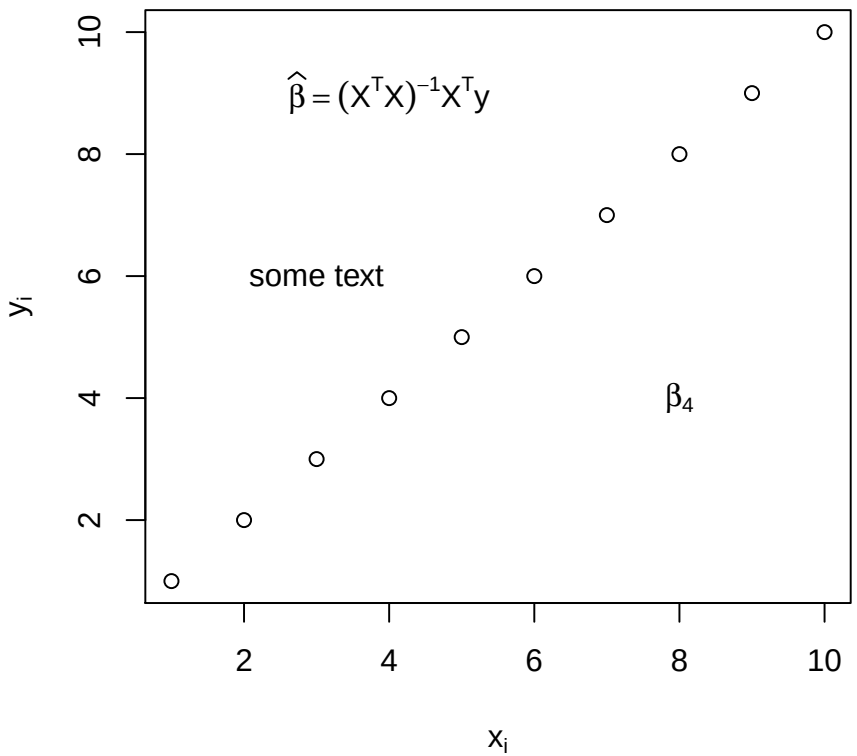
The function image()

```
> (X <- matrix(1:12,nrow=3))
     [,1] [,2] [,3] [,4]
[1,]    1    4    7   10
[2,]    2    5    8   11
[3,]    3    6    9   12
> colours <- c("orange","orangered","red","lightblue",
+                    "blue", "white","lightgrey","grey",
+                    "darkgrey","yellow","green","purple")
> image(as.matrix(rev(as.data.frame(t(X)))),col=colours)
> text(rep(c(0,0.33,0.67,1),each=3),rep(c(1,0.5,0),4),1:12,cex=2)
```



The function `text()`

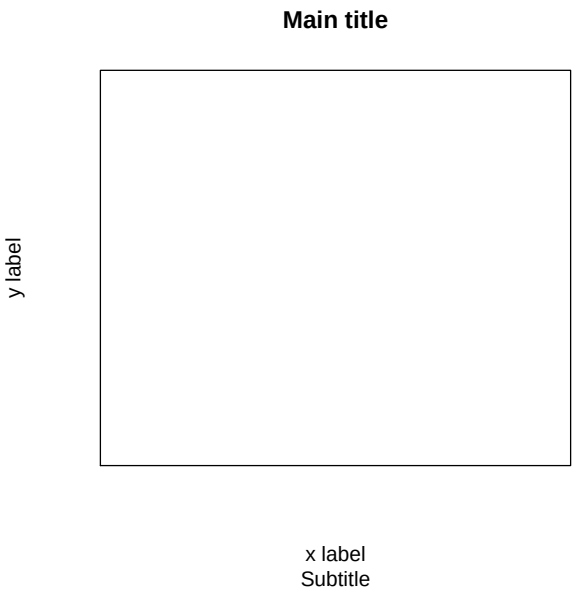
```
> plot(1:10,1:10,xlab=bquote(x[i]),ylab=bquote(y[i]))
> text(3,6,"some text")
> text(4,9,expression(widehat(beta) == (X^T * X)^{-1} * X^T * y))
> p <- 4; text(8,4,bquote(beta[.(p)])) # Combining "math" and
                                       # numerical variables.
```



Use `mtext()` to add text in the margins of a plot.

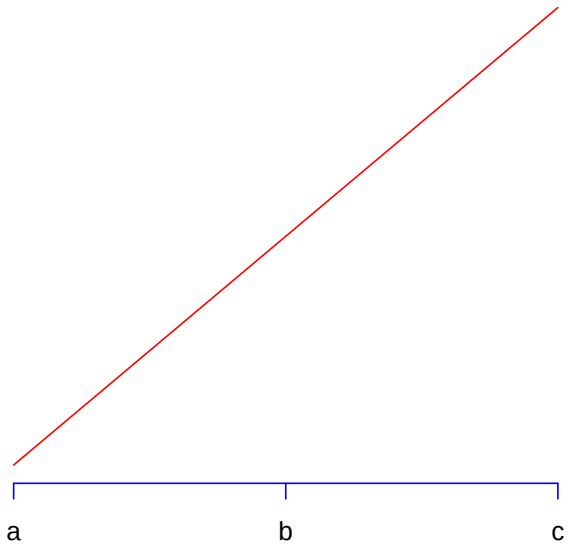
The function `title()`

```
> plot.new()
> box()
> title(main = "Main title", sub = "Subtitle",
+           xlab = "x label", ylab = "y label")
```



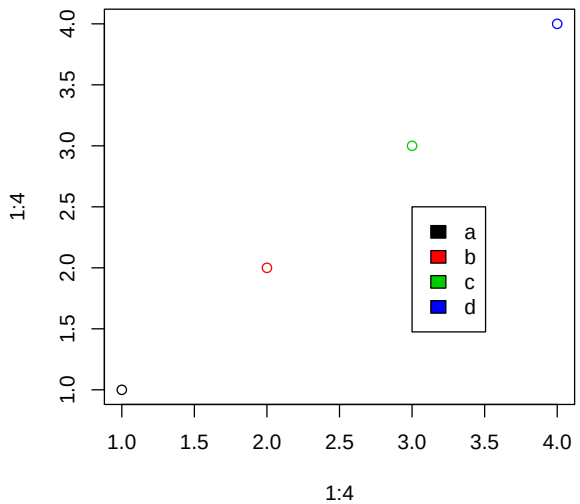
The function axis()

```
> plot.new()
> lines(x=c(0,1),y=c(0,1),col="red")
> axis(side=1,at=c(0,0.5,1),labels=c("a","b","c"),
+          col="blue")
```

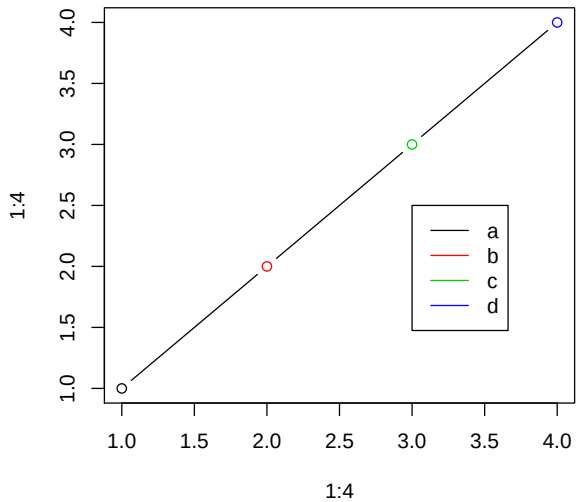


The function `legend()`

```
> plot(1:4,1:4,col=1:4)
> legend(x=3,y=2.5,legend=c("a","b","c","d"),fill=1:4)
```



```
> plot(1:4,1:4,col=1:4,type="b")
> legend(x=3,y=2.5,legend=c("a","b","c","d"),col=1:4,
+        lty=1)
```



The function `locator()`

It is used to place a point on a plot, or to get its coordinates with a click of the mouse. It can also be useful to add text (or a caption) at a specific location, thanks to the mouse.

Enter the following instructions, then click anywhere on the plot you get.

```
plot(1,1)
text(locator(1),labels="Here") # Click on the window.
```

The function `identify()`

It is used to identify and mark points already present on a plot.

Enter the following instructions, then click next to points on the plot. Use a right-click to exit the interactive mode.

```
> plot(swiss[,1:2])
> x <- identify(swiss[,1:2],labels=rownames(swiss))
> x
```

The par() function

The function `par()` takes many arguments to fine-tune your plots. Use this function to set (or query) general graphical parameters. Here is how to use this instruction :

- `par(`*arg-name*`)` outputs the default value of the parameter *arg-name* of the function `par()` ;
- `par(`*arg-name*`=val)` changes the value of the parameter *arg-name* to the value `val` ;
- `par()` returns the list of all graphical parameters currently in use, as well as the current values.

Before changing the values of parameters of the function `par()`, you should save the old values. That way, you can restore them later if needed.

```
# Save the default values of par().
save.par <- par(no.readonly = TRUE)
# Now we can change some parameters.
par(bg="red")
# Then restore the old values.
par(save.par)
```

There are a lot (really !) different argument for the function `par()`.
See `help("par")` or the book.

Your turn to work !

Do the Exercises and the Worksheet on pages 188–192.

http://biostatisticien.eu/springeR/Rbook-chap7.pdf