Tutorial: Graphics in R

Matteo Fasiolo

Graphics in R

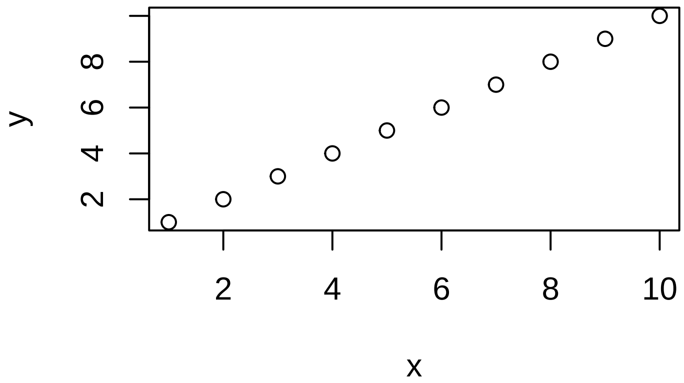
R has a powerful graphics features, which allows users to visualize data easily.

You can type `demo("graphics")` in the command line to see demo code/plots.

You may have already seen some of the graphics functions, so we can be brief.

Create a plot: `plot`

```r
plot(1:10, 1:10, xlab = "x", ylab = "y")
```

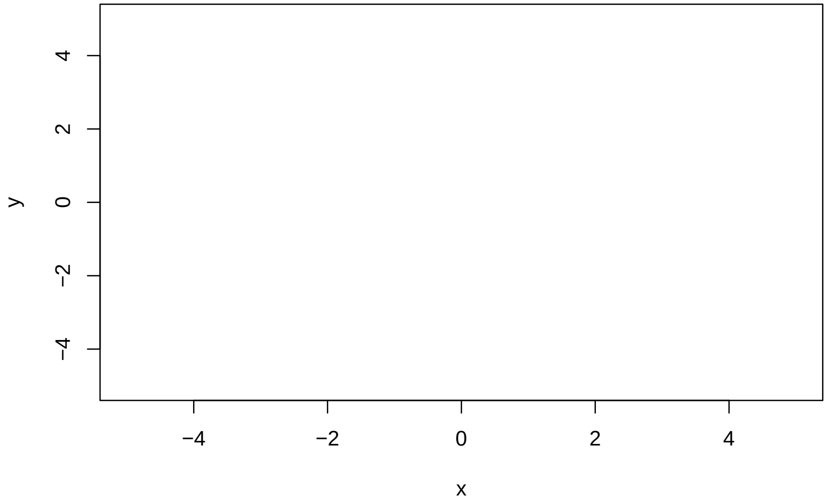


We are calling `plot()` for its **side effects**:

```r
a <- plot(1:10, 1:10, xlab = "x", ylab = "y")
a
NULL
```

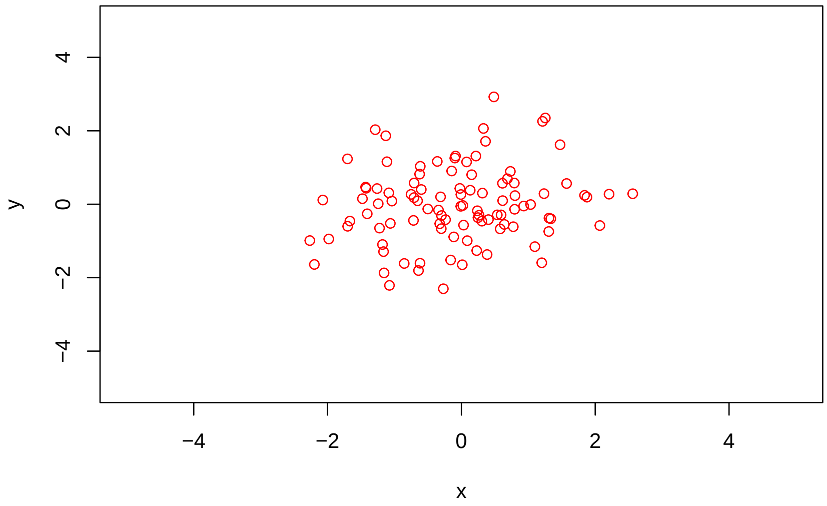
We can use it to produce an empty plot:

```r
plot(1, 1, type = "n", xlab = "x", ylab = "y",
     xlim = c(-5, 5), ylim = c(-5, 5))
```



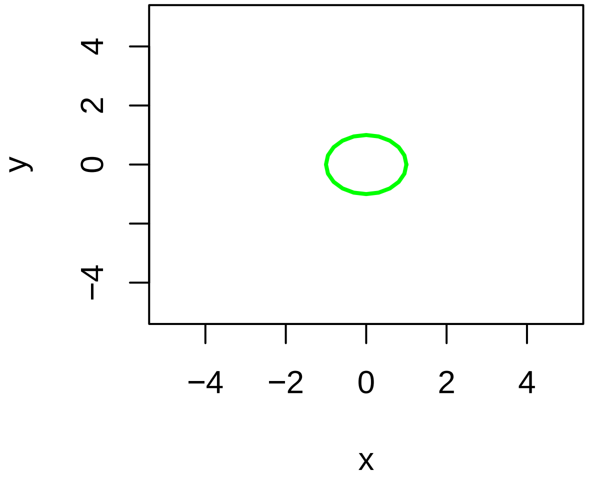
and then fills it with, e.g., points:

```r
x <- rnorm(100)
y <- rnorm(100)
points(x,y)
```



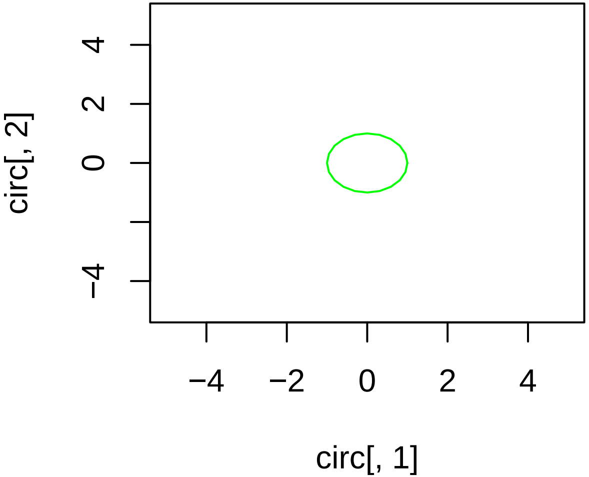
You can draw line in a 2D plot using `lines`:

```r
plot(1, 1, type = "n", xlab = "x", ylab = "y",
     xlim = c(-5, 5), ylim = c(-5, 5))
circ <- cbind(cos(0:20*pi/10), sin(0:20*pi/10))
lines(circ[ , 1], circ[ , 2], col = "green", lwd = 2)
```



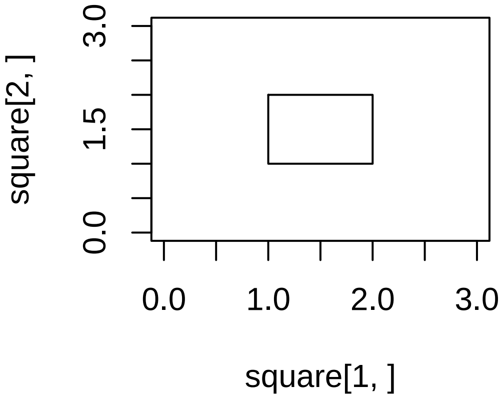
Alternatively:

```r
plot(circ[ , 1], circ[ , 2], type = "l",
     xlim = c(-5, 5), ylim = c(-5, 5), col = "green")
```

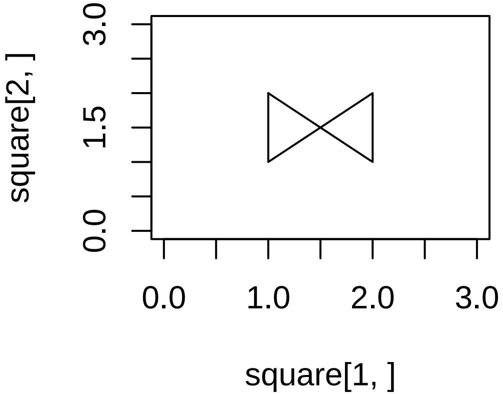


Note that the order matters:

```r
square <- rbind(c(1, 2, 2, 1, 1),
                c(2, 2, 1, 1, 2))
plot(square[1, ], square[2, ], type = "l",
     ylim = c(0, 3), xlim = c(0, 3))
```



```r
square <- square[ , c(1, 3, 2, 4, 5)]
plot(square[1, ], square[2, ], type = "l",
     ylim = c(0, 3), xlim = c(0, 3))
```



Tutorial Exercise

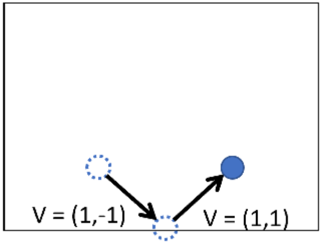
This exercise is inspired by a scene in the TV show “The Office”:

https://www.youtube.com/watch?v=QOtuX0jL85Y

We are going to write an animated visualization that mimics the DVD logo bouncing around the edge of the TV.

1. Create an empty plot, whose x-axis and y-axis range from -5 to 5.
2. Initialize two vectors:
    - $x = (0, 0)$ and $v = (0.23456, 0.12345)$.
    - Draw the vector x as a **red** dot in the plot.
    - Add plot title: “point simulation, press ESC to stop.”
        - Hint: `?title`

3. Now let us write a function that updates the point's `x` location based on the velocity vector `v`.
    - If the position of `x` is slightly out of the box, the point will be **bounced back**.



3. Write a function `update(x,v)` that takes two vectors `x` and `v` as inputs and returns the updated `x` and `v`.
   - First, the function checks if x is outside of the boundary.
   - If so, it updates the velocity vector by "reflecting it". See the picture in the previous slide.
   - Then, update `x <- x + v`
   - Finally, `return( list(x,v) )`
   - The `list` combines two vectors and allows you to return multiple values from a function.
   - Hint: `x[2] > 5 || x[2] < -5` is true if x is above the ceiling or is below the floor.

Note on lists, if you do `z <- list(x,v)` than `z[[1]]` returns `x` and `z[[2]]` returns `v`.

That's all you need to know, but if you prefer you can use a matrix to store `x` and `v`.

4. Your code should look like this:

```r
update <- function(x,v){
  # TODO: Check the boundary collision,
  #       and update v and x.
  return(list(x,v))
}

x <- c(0,0) #the initial position of x
v <- c(0.23456,0.12345) #the initial velocity of x

while(T){
  # TODO: create a new plot and draw x's current
  #       location
  x <- xv[[1]]
  v <- xv[[2]]
  Sys.sleep(0.1)  # wait a bit to give Rstudio
                  # the time to plot
}
```

Your plot should look like this:

https://github.com/anewgithubname/MATH10017-2022/blob/main/lecs/lab14_1.mp4

In the TV show, each time the logo bounces, it changes color randomly. Could you revise your code so that the point in your plot also changes its color every time it bounces?