In-Class Computing Day 4

Math 253: Statistical Computing & Machine Learning

Today's programming tasks provide an introduction to the basic building blocks of graphics: frames, points, lines, and polygons. In practice, you'll use higher-level graphics systems such as provided by `lattice`, `ggplot2`, `ggvis` and similar packages. But you'll understand the use of such packages better if you know the very basics.

Task 1: The frame

Graphics are drawn in a *frame*. You can think of this as a piece of paper, a computer screen, or whatever space you like. In addition to providing the physical space for drawing, the frame defines the *coordinate system* in which graphics are drawn. We'll work with the most common system: cartesian coordinates.[1]

The basic function to draw frames is `plot()`, part of the `graphics` package that is a part of the standard R distribution. An empty frame, with axes for the x- and y-axes is generated by this somewhat wordy command:

```
plot(1, xlim = c(0, 100), ylim = c(0, 100), type = "n")
```

`plot()` doesn't return any value. It's used for the *side effect* of setting up the graphics display with a frame.

[1] There are other coordinate systems, e.g., radial, map projections, etc.

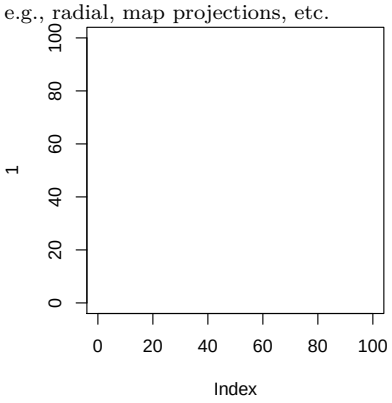

Figure 1: An empty frame.

Task 2

Create two objects named `x1` and `y1` that contain values marking the points at the corner of a square.

Use `lines()` to connect the points in `x1` and `y1`. Note that `lines()` doesn't close the square.

Create `x2` and `y2` so that `lines()` will draw all four sides of the square.

Task 3

Create objects named `x3` and `y3` that follow a circular path.[2] Center the circle at $(x = 50, y = 60)$ and give it a diameter of 40.

[2] Back to trigonometry! Use `seq()` to create a set of `angles` from 0 to $2 * pi$. Then `sin()` and `cos()` can convert the angles to the y-coordinate and x-coordinate respectively.

Task 4

Create objects named `x4` and `y4` to create a figure-8 shaped path. One way to do this is to use statements similar to those for drawing a circle, but vary the radius at each of the `angles` to be proportional to `abs(sin(angles))`.

Use `polygon()` to fill the path created by `x4` and `y4`.

Task 5

Bells and whistles …

Often you want to customize graphics, that is, set features of the graphic such as color, axes labels, etc. to values other than their defaults.

Use each of these optional arguments to customize a graphic:

- `col =` sets the color of points or lines and the fill of polygons. Values like "red", "blue", "wheat", etc. can be used. (Evaluate `colors()` at the console to see a complete list.) If you want a transparent color, use a construction like `col = col2rgb('tomato', alpha = .1)`
- `lwd =` takes a number indicating line width.
- `pch =` takes a small integer to set the shape of the point made by `points()`, or a single character, e.g. `x`.
- `xlab =` and `ylab =` set the x- and y-axis labels repectively
- `xaxt =` and `yaxt =` suppress the axis tick marks when set to the value `"n"`.