Stat 243: Problem Set 5, Due Friday Nov-05

October 24, 2016

Instructions

Please turn in (1) a copy on paper, as this makes it easier for us to handle AND (2) an electronic copy through bCourses so we can run your code if needed.

Random Walk in one dimension

Consider a random walk of one particle in one dimension. This type of walk involves taking a unit step with 0.5 probability to the right or to the left. Write a function that measures how many steps it takes for one particle to reach a given point $x = x_p$. Report how steps are needed to reach the points: $x = 5$, 50, 5000, and 50000.

Area of π

Use the Monte Carlo method "throwing random points" to compute π by computing the area of a circle. Choose the geometric region G as the circle with its center at the origin $(0, 0)$ and with unit radius. Choose the region B as the rectangle $[-1, 1] \times [-1, 1]$. A point (x, y) lies within G if $x^2 + y^2 < 1$. Compare your approximation of π with the R variable `pi`.

Random Walk in two space dimensions

Consider a discrete random walk in two dimensions. At each step there is a 0.25 probability of moving one step to the left, right, up, or down. An example of a random walk of three steps would be the first step is to move one unit to the right, the second step is to move one unit up, and the third step is to move one unit back down to the position attained after the first step.

- Write a function that generates such a random walk.
- The input argument should be the number of steps to be taken.
- There should be an optional second argument (with a default) specifying whether the user wants the full path of the walk returned or just the final position.
- If the user chooses the full path, the result should be given in a reasonable format.
- Your code should check that the input is a valid integer (it should handle zero and negative numbers and non-integers gracefully).
- Finally, you could use `Rprof()` to assess where the bulk of the computation is in your code.
- If you are using `for` loops in your code, try re-implementing another function that favors vectorized code.

Object Oriented Programming

Take your code from the previous problem and embed it in object-oriented code that nicely packages up the functionality. You can choose either S3 or S4 classes.

- You should create a *class* `"rw"` or `"randwalk"` with a constructor function.
- Write a *print* method for which the result should focus on where the final position is and some useful summary measures of the walk.
- Include a *plot* method that graphs the random walk.
- Likewise, write a `"["` operator that gives the position of the i-th step.
- Create a *replacement* method called `start()` that translates the origin of the random walk, e.g. `start(my_walk) <- c(3, 5)` should move the origin and the entire walk so that it starts at the position $x = 3, y = 5$.

Shiny App

With your OOP implementation of the 2-dimensional random walk, create a shiny app to visualize the plot:

- Use a `sliderInput()` to indicate the number of steps: min = 20, max = 10000.
- Use a `numericInput()` to indicate a number of the random seed—to be used in `set.seed()`.
- Use two `numericInput()`'s to provide the coordinates of the start position for the random walk. Limit the starting point to the region $[-10, 10] \times [-10, 10]$.
- Optional: try to display the coordinates of the final position on the sidebar panel.

The image below shows an example of what the shiny app could like. Notice that I added colors to the start (blue) and end (red) positions.

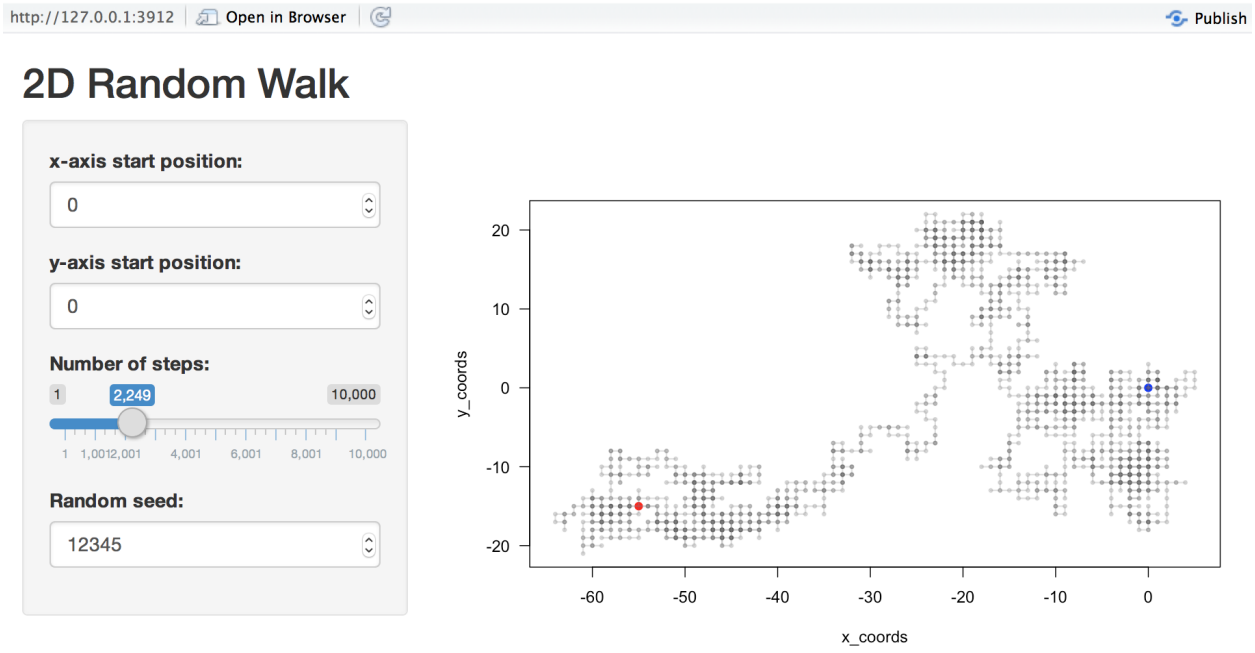

Figure 1: shiny app with random walk