question 2 views

Daily Challenge 25.7

~~(**Due: Sunday 3/31 at 12:00 noon Eastern**)~~
(**Due: Monday 4/1 at 12:00 noon Eastern**)

For this chapter's tutorial, you will prepare a pedagogical Jupyter notebook mimicking the style of a textbook section. The goal is to explain to a fellow student, who has not yet studied probability and applications of integration, how various probability distributions emerge from the random walk (and how their properties can be studied with integrals).

For examples of good textbooks in Jupyter, you might glance at:

- This random forests chapter of the Python Data Science Handbook.
- This chapter on Bayesian filtering.
- This introduction to probability from a data science course.

All three examples have the same core features: they blend $\LaTeX$ calculations, snippets of Python code, and explanatory text to create an interesting computational narrative for the reader.

(**Part a**) Revisit your solution to DC 24.2 on random walks and diffusion. Open a fresh Jupyter notebook and create an outline of the sections you'd like to include to explain this calculation to a student. Start entering whatever $\LaTeX$, code, and figures you might want into your outline. Think about what additional plots, examples, and explanations you need to include (besides the ones I asked you to do in the original DC) in order to explain this clearly to the student.

Don't break up the notebook into the sub-parts (a), (b), (c), etc., as though you are writing a solution to an assigned problem. This is not simply a re-write of your previous solution; you need to be creative, think of additional content, and design the presentation in a way that helps the reader follow the flow of the argument.

(**Part b**) I'll have you add some more content to the notebook to make it long enough for a 40-60 minute presentation.

For now, write a function which performs some number of random walks, with a given number of steps per walk, and counts how often the random-walker is at a *positive* value of x.

For instance, if my walk begins $[0, +1, 0, -1, 0, +1, +2]$ your code should return $\frac{3}{7}$, since the walker was at $x > 0$ at three out of the seven times.

The code might look like

```
def count_positive(n_walks = 100, n_steps = 1000):
   """
   Runs n_walks random walks with n_steps each, then
   returns a list of the fractions of time spent at
   positive x in each walk
   """
   positive_fractions = []

   for walk in range(n_walks):
      positive_counter = 0

      ## Do a walk with n_steps steps, incrementing the counter when he's positive
      ## Divide the positive count by the number of steps and append to the list

   return positive_fractions
```

We'll explain why this distribution is interesting tomorrow, and then you can add it to your Jupyter explanation.

daily_challenge

Updated 13 days ago by Christian Ferko

the students' answer, *where students collectively construct a single answer*

bloop

Updated 5 days ago by Logan Pachulski

the instructors' answer, *where instructors collectively construct a single answer*

Click to start off the wiki answer

followup discussions *for lingering questions and comments*