Pythonic Indicator Variables

Tianchen Liu

A lot of folks in office hours are confused about indicator variables. How is a r.v. a sum of r.v.'s, where the smaller r.v.'s could take on either 1 or 0? What does this suppose to mean? In this short note I would like to share a cute little trick in understanding and designing indicator variables: by thinking about indicator variables *in Python*. Specifically, in the form,

```
X = sum([boolean_expression(i) for i in something])
```

And the boolean expression is the indicator variable we're looking for.

1 A Basic Example

Randomly throw k balls into n bins, what's the expected number of empty bins in total?

This is one of the most straightforward indicator-variable problems, I use it here to demonstrate a simple example.

How do I express the total number of empty bins in python? How can I write it as a sum of a list comprehension in the form above?

Here's how I'd do it:

```
number_of_empty_bins = sum([is_empty(i) for i in bins])
```

And now everything's clear: Create n indicator variables X_1 through X_n. $X_i = 1$ if bin i is empty, 0 otherwise. We want to compute $\mathbb{E}(X)$, where $X = X_1 + \cdots + X_n$. I'll leave the rest to you.

2 A Trickier Example

I have a normal deck of 52 cards face down, what's the expected number of cards that I need to turn over in order to see the first ace?

The way to come up with indicator variables is trickier than the last one, because the r.v. that we're trying to compute is not an explicit *sum* of anything... But we can make it look like one!

```
num_cards_flipped = 1 + sum([on_top_of_first_ace(i) for i in non_ace_cards])
```

The function `on_top_of_first_ace(i)` returns `True` if the card i is on top of the first ace in the card deck. Otherwise it returns `False`.

In other words, the number of cards we need to flip = the total number of non-ace cards on top of the first ace. $\mathbb{E}(X) = \mathbb{E}(X_1 + \cdots + X_{48})$, where $X_i = 1$ if card i is on top of the first ace in the card deck, 0 otherwise.

Remark: the hard part of the problem is to compute $P(X_i = 1)$.

3 A Homework Example

(Fa20 HW11 Q5) Create a graph by randomly select m edges from all possible edges within n vertices, what's the expected value of triangles in this graph?

As always, try to make this a sum of boolean expressions.

```
num_triangles = sum([forms_triangle(i, j, k) for (i, j, k) in groups_of_three])
```

The function `forms_triangle` returns `True` if vertices i, j, k forms a triangle in the graph. Otherwise it returns `False`. Coming up with $E(X_i) = P(X_i = 1)$ is not that trivial, and you may need to think about how many groups of three vertices are there (it's not n^3).