Permutations and Factorials

(https://github.com/learn-co-curriculum/dsc-permutations-and-factorials-v2-1) (https://github.com/learn-co-curriculum/dsc-permutations-and-factorials-v2-1/issues/new/choose)

Introduction

In the previous lab, you defined a few sample spaces by counting the total number of possible outcomes. This is not very practical when sample spaces grow. In this lab, you'll be introduced to *permutations*, which will provide a structured way to help you define sample space sizes!

Objectives

You will be able to:

- Describe how factorials are related to permutations
- Mathematically derive how many permutations there are for large sets
- Calculate permutations of a subset
- Calculate permutations with repetition and replacement

Defining the Sample Space by Counting

Let's consider the following example.

The Beyoncé tribute band "The Single Ladies" is playing a free mini gig in your local park next week. They have selected three all-time classics: "Drunk in Love", "Crazy in Love" and "Formation", but still have to decide the order they will play the songs in. Knowing this, how many playlists are possible?

It is easy and fairly quick to write down possible orders here:

"Drunk in Love", "Crazy in Love", "Formation"

"Drunk in Love", "Formation", "Crazy in Love"

"Crazy in Love", "Drunk in Love", "Formation"

"Crazy in Love", "Formation", "Drunk in Love"

"Formation", "Drunk in Love", "Crazy in Love"

"Formation", "Crazy in Love", "Drunk in Love"

That's it! When we count the possible outcomes, we get to 6 elements in the sample set. Now what if "T Chat 'ies" plays a setlist of 4 songs? or 5? That's where the notion of *permutations* comes in

Permutations

The problem setting, in general, is that there are n objects and we want to know how many *permutations* are possible.

This is a way how you can tackle this. You're the lead singer and have to decide which song to play first. You have 3 songs to choose from, so 3 ways of choosing a first song. Then, you move on to the second song. You've chosen the first one, so you have 2 songs to choose from now, etc. Mathematically, this boils down to:

`# Beyoncé permutations` $= 3 \times 2 \times 1 = 3! = 6$

Generalizing this to n, this means that the number of permutations with n distinct objects is $n!$, or the factorial of n.

Permutations of a Subset

Now, let's consider another example. "The Single Ladies" are still playing a concert at central park, but they disagree on the final three songs that they will play. They only get a 12 min gig slot, so they really can't play more than 3, yet they have a shortlist of 8 they need to pick from. How many final song selections are possible given this info? As for the first example, the order of the songs played is still important.

When the band members decide on the first song, they have 8 possible songs to choose from. When choosing the second song, they have 7 to choose from. Then for the third song, they have 6 left.

`# Beyoncé k-permutations` $= 8 \times 7 \times 6 = 336$

formalizing this, the question is how many ways we can select k elements out of a pool of n objects. The answer is

$n \times (n-1) \times \ldots \times (n-k+1)$ or in other words, $P^n_k = \dfrac{n!}{(n-k)!}$

This is known as a k-permutation of n.

The idea is here that we only "care" about the order of the first k objects. The order of the other $(n-k)$ objects doesn't matter, hence they're left out of the equation.

Permutations with Replacement

When talking about setlists, it makes total sense to assume that songs will not be played twice. This is not always how it works though. Imagine a bag with three marbles in it: a green one, a red one, a Chat Now we'll draw marbles three times in a row, but each time, we'll write down the m ıd *put it back in the bag* before drawing again.

Now the number of possible outcomes is $3 \times 3 \times 3$.

Generalizing this to n, this means that the number of permutations with replacement when having n distinct objects is equal to n^j where j is the number of "draws".

Permutations with Repetition

When using permutations, some elements may be *repeated*.

A classic example is using permutations on words. Let's say you have the letters of the word "TENNESSEE". How many different words can you create using these letters?

Simply saying that there are 9 letters so the answer is $9!$ does not give you the correct answer. Looking at the word TENNESSEE by itself, you can swap the 3rd and the 4th letter and have the same word. So the total number is less than $9!$.

The solution is to divide $9!$ by the factorials for each letter that is repeated!

The answer here is then (9 letters, 4 x E, 2 x N, 2 x S)

$$\frac{9!}{4!2!2!} = 3780$$

The general formula can be written as:

$$\frac{n!}{n_1!n_2!\dots n_k!}$$

where n_j stands for identical objects of type j (the distinct letters in our TENNESSEE example).

Level-Up: Factorials and Recursion

At the start of this lesson, when discussing the number of possible permutations we can obtain for n distinct objects, we mentioned the concept of the factorial of n, denoted by $n!$.

In the example presented to you, we wanted to count all possible ways in which three different Beyoncé songs could be played by the Beyoncé tribute band "The Single Ladies". There were 3 possible ways of choosing a first song, 2 possible ways of choosing a second song, and only 1 way of choosing a third and final song, for $3 \times 2 \times 1 = 6$ different ways in which the three different songs could be played. This number, 6, is equal to the factorial of 3, $3!$, the number of permutations of 3 distinct objects.

Here, $3! = (3 \times 2 \times 1) = 6$. Notice that this is the same as writing $3 \times 2! = 3 \times (2 \times 1)$ and $3 \times 2 \times 1! = 3 \times 2 \times (1)$. (By definition, the factorial of 1, $1!$, is equal to 1. The factorial of 0, $0!$ is also defined to be equal to 1.)

Chat

We can generalize this to the case of computing the factorial of an integer n, $n!$. The factorial of n, $n!$, can be written as $n * (n-1)!$, which itself can be written as $n \times (n-1) \times (n-2)!$. That is, we can define the factorial of n in terms of the product of n and the factorial of $(n-1)$, and the factorial $(n-1)$ can be defined in terms of the product of $(n-1)$ and the factorial of $(n-2)$, and so on and so forth, as seen in the equation below, until we get to $1!$, which is defined to be equal to 1:

$$n! = n \times (n-1)! = n \times (n-1) \times (n-2)! = \ldots = n \times (n-1) \times (n-2) \times \ldots \times 2! = n$$

Recursion

When we define a function in terms of itself, in this case, the factorial of n in terms of the factorial of (n-1), we are using **recursion**. Recursive functions are functions that can call themselves in order to loop until a condition is met. In the next lab, you'll get a glimpse on how to write a recursive function in Python, but in the Appendix to this Module, we go over recursive functions in Python in much more detail.

Summary

Now you're well on your way to calculate all sorts of permutations using factorials - both for understanding the sample space, subsets, etc! Let's move on for some practice!

How do you feel about this lesson?

Have specific feedback?

Tell us here! (https://github.com/learn-co-curriculum/dsc-permutations-and-factorials-v2-1/issues/new/choose)

Chat