Video #15: Randomizing Things

In the last video, you learned how to rearrange things in a sorted order, using the MATLAB sort function. In this video, we'll do the opposite, rearrange things in a random order. You may be writing a program to run an experiment with lots of trials or conditions, and want to run those trials or conditions in a random order. Or you may be implementing a survey or quiz, and want to present the questions and answers in a random order. In a different context, suppose you're implementing a model or simulating a process where events occur at random, with a certain probability. Consider simulating the spread of a virus, for example - if an individual is exposed to the virus, there's a certain chance, maybe a 20% chance, that they'll become infected. To implement programs like these, we use MATLAB functions to generate random numbers or sequences.

We've seen a couple examples already - the rand function, for example, when called with no inputs, returns a random floating point number between 0 and 1. Suppose we wanted to simulate a coin flip - if it's a fair coin, then you'd expect to get heads and tails each about half the time. We expect the rand function to return a number larger than 0.5 about half the time, so we can use our random number to determine whether to display heads or tails. <run section> Note that I could have just called rand inside the if, instead of creating this separate variable called num. We can call rand with two inputs that specify the number of rows and columns of a matrix of random numbers - here generate a row vector of 5 random numbers or a 2 x 4 matrix of random numbers. If we just provide one input, the function returns a square matrix of random numbers, in this case a 3 x 3 matrix. <run section>

We can generate random integers with randi - if we provide a single input n, randi returns one random integer between 1 and n, and we can also add input rows and columns to generate a matrix of random integers. In these two examples, we generate 6 random integers between 1 and 10, or a 3 x 2 matrix of these integers. <run section>

Earlier we raised the idea of randomizing the order of things. A useful function for this is randperm. With a single positive integer n, this function returns a random permutation of the integers from 1 to n, in this example, a permutation of the integers 1 to 10. There's an optional second input, expected to be a smaller integer, returns k unique integers between 1 and 10 - here 5 - <run section> Note that if we generate a set of random integers with randi, there can be repeats, but there's no repeats with randperm (important distinction between two functions)

So let's look at some examples of using randperm to randomize the order of things. Let's say we have a set of conditions that we want to implement - maybe they're parameters for an experiment or simulation, and we want to run the experiment with each of these parameters, but in a random order. The conditions here are a set of values from 3 to 6, going up by a half. We generate a random permutation of the number of conditions, and create a new ordering of the conditions by using the order as a vector of indices, this will collect contents of conditions in this order. <run section> We can use the same idea to shuffle a deck of cards - here I represented each card as a string that starts with a number or j/q/k for jack/queen/king, and

the h/d/s/c stand for hearts, diamonds, spades, and clubs. To shuffle them, we'll create a random permutation of the integers from 1 to 52 and use that as a vector of indices for cards - we can do this for a cell array in the same way that we rearrange the content of a vector. In the final example here, we create an anagram, which is a string that contains the same letters, in different order. We create a random permutation of integers from 1 to the length of the string, use as a vector of indices to rearrange the letters in rumpelstiltskin. <run section>

Finally let's consider a problem we posed earlier, of randomizing the order of questions and answers presented in a quiz. I created an App named cityQuiz - let's first run it and see what it does. Each question consists of the name of a state in the US and three city names, one of which is a real city in that state, and we need to select the correct city <cityQuiz, all correct>. Suppose you want to randomize the order of the questions so that each time you run the App, the user sees the state names in a different order. Also within each question, suppose you want to randomize the order of the 3 answer options (3 city names). We'll just look at two snippets of the code that deal with the ordering of questions and answers. The first piece creates new properties with the information for the quiz. For simplicity, I put the strings for the states and cities directly in the code rather than reading them from a textfile. We have a cell array named states with 5 state names. Cities is a cell array with 5 cell arrays inside, and each one has three city names to use as the answer options for the corresponding state. So the first cell array has the answer options for Colorado, the second has the options for Massachusetts, and so on. In each case, the correct city is the first string in each cell array, so there's really a Loveland Colorado, Chicopee Massachusetts, Rome Georgia, Aloha Oregon, and Normal Illinois.

We'd like to rearrange the order of the state names and apply the same reordering to the five cell arrays in cities, so if Georgia ends up first, for example, then the third cell array starting with Rome should be moved to the first position of the cities cell array as well. The second thing we want to do is rearrange the order of the three city options in each inner cell array, and in the process, keep track of where the correct city gets placed - we'll record that in the vector correctAns. There's a start survey button on the GUI that was enabled when the App started, and when pushed, this callback function was invoked, startsurveyButtonPushed. I'm just showing the first part of this callback function that deals with reordering the questions and answers. First, we generate a random permutation of the integers from 1 to totalQuests, which is 5 here. Then we can use that same order to rearrange the five state names in states, and rearrange the 5 cell arrays in cities. The for loop here rearranges the three answer options within each inner cell array. We loop through the five cell arrays, and for each one, we generate a random permutation of the integers 1,2,3. We record where the correct city name ends up (location of the number 1 in the permutation), rearrange app.cities{i} with this order. At the end of the program, which you can explore on your own, we compare the cities chosen by the user with the correct answers to share the results with the user.

In this video, we reviewed two functions for generating random numbers, rand and randi, and introduced a new function for generating random permutations of integers called randperm, and illustrated how we can use this function to randomize the order of entities in a program.