

3.2. Random Sampling, and The Birthday Problem

Purpose: Learn to sample with and without replacement using Minitab. Learn how to simulate a probability (hard to obtain by direct calculation) in Minitab by studying the famous "**Birthday Problem**." That is, simulate the probability that at least two students in your class have the same birthday. There are two basic sampling procedures: samplings without **or** with replacement. The first one is used in most of cases, but the birthday problem needs sampling with replacement. In this lab, we will get familiar with the sampling first and then study the birthday problem.

Reading Assignment: Read through Section 3.7.

Caution: Stored Minitab commands are used in this lab. For them to work, you must start Minitab from **your own Novell account**. This can be done by double-clicking on the file "Start Minitab.MTB" from your Novell account window when Minitab is *not* already running.

Part I: Illustrate the ease of drawing random samples using Minitab:

Section 3.7 of the textbook concerns how to use a table of random digits to obtain a random sample of size n from a population. It is much easier to obtain a random sample using Minitab. Suppose we have a population consisting of N elements, which have been identified with the numbers 1 through N . (That is the hard part, knowing how many elements are in the population and being able to identify them as element 1, element 2, and so on to element N . If you can do that, then to draw a random sample of size n from the population is easy.)

Sampling without replacement: According to Definition 3.10 (page 150), if n elements are selected from a population in such a way that every set of n elements in the population has an equal probability of being selected, the n elements are said to be a **random sample**. For such a sample, one obtains n **different** elements of the population. One way you could think of obtaining such a sample is to select one element of the population at random, then select one more at random from those not yet selected, and so on until n have been selected. This is called "sampling without replacement," since each element selected is not put back in (so can't be selected more than once).

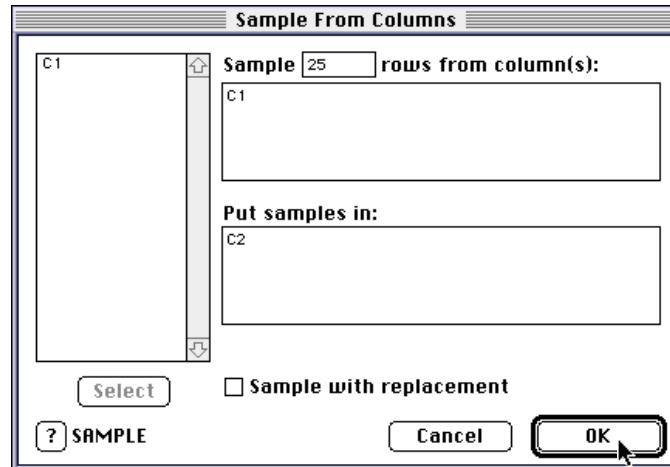
Step 1: Start Minitab from your Novell account by double clicking on the "Start Minitab.MTB" icon.

Step 2: Take a random sample of size $n=25$ from a population of $N=365$ elements labeled 1 to 365, sampling **without replacement**. To do this, enter the following Minitab commands in the Session window; (**DO NOT** enter the comments (which begin with "#", which are provided for your information, describing what each command does.)

```
MTB > set c1          # set the integers from 1 to 365 into c1
DATA> 1:365
DATA> end
MTB > sample 25 from c1, put into c2
```

(The "sample" command tells Minitab to take a random sample of size 25, without replacement, from the integers in c1, then to put the sample in c2.)

(The sample could also be obtained using Minitab menus by selecting "Calc -> Random Data -> Sample From Columns..." and completing the resulting dialog box as shown below.)



To see the elements of the sample in the order selected, print column 2 by giving the following command.

```
MTB > print c2
```

To see the elements of the sample sorted by magnitude, do the following.

```
MTB > sort c2, put into c3    # sort the sample, putting the results into c3
MTB > print c3               # print the sorted sample
```

STOP AND THINK: Look at the resulting sorted sample. Were any elements included in the sample more than once? Could there be?

Sampling with replacement: It is also possible to draw a random sample of size n from a population of size N sampling **with** replacement. To do so, one element is drawn at random then put back into the pool of elements. Then a second is drawn at random and replaced. This is repeated until n elements have been drawn at random with replacement. When sampling **with** replacement, **it is possible** for an element of the population to be included in the sample multiple times.

Step 3: Take a random sample of size 25 from the population of 365 elements labeled 1 to 365, **with replacement**, then sort and print the sample. We use:

```
MTB > random 25 c1;
SUBC> integers 1 to 365.
MTB > sort c1, put into c2
MTB > print c2
```

With sorting, we can easily see if an element is selected twice into the sample. In the birthday problem, we want to see if a day (the element) is "selected" by two students which means the two students have the same birthday.

(The sample could be also obtained using menus by "Calc -> Random Data -> Sample From Columns...", but in the dialog box shown above, one would click in the box "Sample with replacement".)

STOP AND THINK: Look at the resulting sorted sample. Were any elements included in the sample more than once? Could there be?

Step 4: Repeat step 3. [The easiest way to repeat a set of commands is to copy them (i.e. select them in the Session window, then select "Edit -> Copy" from the menu), then paste them at the last Minitab prompt, then press the "return" key. (Before pressing the "return" key, you could also edit the commands if you wanted.)]

STOP AND THINK: Again, were any elements included in the sample more than once? What do you think the chances are that all 25 sampled elements will be distinct? This is related to a famous probability problem called the Birthday Problem, which we explore in Part II.

Part II: The Birthday Problem. Suppose a class contains 25 people. What are the chances that no two of the 25 people share the same birthday? (Venture a guess!)

STOP AND THINK: There are 365 days in the year (most years). Given 25 people, are their 25 birthdays independent of one another? For each person, is each day of the year equally likely to be his or her birthday?

If the answer to both questions is "yes", and if none of the 25 people have a birthday on February 29th, then the birthdays of the 25 people are like a random sample of size 25, sampling **with** replacement from 365 days. The question we want to explore is, "**What are the chances that no two of the people share the same birthday?**" In Part I of this lab, you simulated this situation twice. (Were there matching birthdays in 1, 2, or neither of the simulations? Correspondingly, one might estimate the chances of a match to be closer to 50%, 100%, or 0%. However, based on Lab 3.1, where we explored the law of averages, we would want to run the simulation **many times** to get a good estimate of the chances! Thus, we need some systematic (i.e. easy) way to get Minitab to do the work for us! First we will explore how to have Minitab check for matches. Then we will learn to **store** a set of Minitab commands and **execute** it many times.

We will now illustrate and use one way to "program" Minitab to simulate the birthday problem and check for matches. Recall, **we need to take random samples of size 25 from a population of size 365, sampling with replacement, and check for matches for many times (100).**

Step 1: Enter the following commands.

```
MTB > erase c1-c5
MTB > let k1=0
MTB > print c1-c5
MTB > print k1
```

This erases the contents of columns c1-c5, stores the value zero in k1, and then prints the contents of columns c1-c5 (which are empty, having been erased) and the constant k1 (which is zero). (Minitab lets you store values

in two places: in the columns c_1, c_2, \dots , and in the locations k_1, k_2, \dots . While each column can contain many values (i.e. one in each row), the locations k_1, k_2, \dots can each hold only one value. These locations are called **constants** in Minitab, perhaps because each can contain only one value, but it is better to think of each as being a single variable, since its value can be changed.

Step 2: Next, enter the following commands. (When storing commands, type especially carefully! If you mess up, you may need to enter the "end" command then start over again, copying and pasting what you can.)

```
MTB > store
STOR> name c1 'sample' c2 'sorted' c3 'lag' c4 'diff' c5 'MinDiff'
STOR> random 25, put into c1;
STOR> integers from 1 to 365.
STOR> sort c1, put into c2
STOR> lag c2, put into c3
STOR> let c4 = c2 - c3
STOR> let k1 = k1 + 1
STOR> let c5(k1) = minimum(c4)
STOR> end
MTB >
```

Nothing happens! The commands are stored by **not executed!** They can now be executed any number of times very easily.

Step 3: Execute the stored commands one time. To do this, you only need to enter the "execute" command **exec** below -- this executes the stored commands that follow it.

```
MTB > exec
```

Step 4: To see what has happened, print the contents of k_1 and c_1 - c_5 , using the commands shown below. (The contents of your columns will almost certainly be different from those shown, because you took a new random sample.)

```
MTB > print k1
K1      1.00000
MTB > print c1-c5
```

ROW	sample	sorted	lag	diff	MinDiff
1	55	3	*	*	1
2	205	10	3	7	
3	12	12	10	2	
4	185	19	12	7	
5	165	23	19	4	
6	33	33	23	10	
7	200	35	33	2	
8	46	41	35	6	
9	294	46	41	5	
10	145	55	46	9	
11	23	58	55	3	
12	347	86	58	28	
13	10	106	86	20	
14	106	131	106	25	
15	86	145	131	14	
16	19	153	145	8	

17	337	165	153	12
18	58	185	165	20
19	295	200	185	15
20	131	205	200	5
21	3	294	205	89
22	41	295	294	1
23	320	320	295	25
24	153	337	320	17
25	35	347	337	10

MTB >

K1=1 tells us that we have executed the stored commands one time. Here we want to see if one date appears more than once in "sorted". Therefore we generate "lag" by moving down one cell of "sorted" and then take the difference between "lag" and "sorted". A zero in "diff" indicates a match. Looking at the sorted sample, the closest values (birthdays) are days 294 and 295, with a difference of 1, which is stored in the first cell of c5 (i.e. c5(1)).

STOP AND THINK: For this sample, there are no matching birthdays, the closest being one day apart. What would be the smallest difference if the sample did contain a match? Where there any matching birthdays in your sample? How close were the two closest birthdays in your sample?

Step 5: Execute the stored commands a second time, but use the **noecho** command. This keeps the stored commands from being shown (echoed) in the Session window. Again, print the results. (Your results will most likely differ from those shown below.)

```
MTB > noecho
MTB > exec
MTB > print k1
K1      2.00000
MTB > print c1-c5
```

ROW	sample	sorted	lag	diff	MinDiff
1	74	67	*	*	1
2	360	74	67	7	0
3	244	96	74	22	
4	190	107	96	11	
5	204	122	107	15	
6	316	136	122	14	
7	299	143	136	7	
8	324	170	143	27	
9	258	171	170	1	
10	122	190	171	19	
11	249	204	190	14	
12	298	244	204	40	
13	271	249	244	5	
14	96	258	249	9	
15	316	271	258	13	
16	136	289	271	18	
17	67	298	289	9	
18	289	299	298	1	
19	143	316	299	17	
20	320	316	316	0	
21	343	320	316	4	
22	334	324	320	4	
23	170	334	324	10	
24	171	343	334	9	
25	107	360	343	17	

```
MTB >
```

In the second sample above, there are two birthdays on day 316, so the minimum difference is zero, indicating a match.

STOP AND THINK: Did you get a match in your second sample? What is the minimum difference between birthdays for your second sample?

Step 6: Execute the commands 98 more times, for a total of 100 simulations of the birthday problem. Doing it 98 times, we definitely want to use the noecho command, to avoid having the commands show up in the Session window 98 times! Print only the contents of k1 and c5. Use the **tally** command to compute the number and percentage of samples for which the closest birthdays match, for which they are one day apart, two days apart, and so on.

```
MTB > noecho
MTB > exec 98
MTB > print k1
K1      100.000
MTB > print c5
```

(The 100 values of "MinDiff", corresponding to the 100 simulations, will be printed.)

```
MTB > tally c5;
SUBC> counts;
SUBC> percents.
```

(Counts and percentages of 0's, 1's, etc. will be printed.)

It takes a few minutes for Minitab to execute the commands, so be patient. The cursor becomes a wrist watch to let you know that Minitab is working on it! The percentage of 0's is the relative frequency of a match, which is also the estimate of the probability of match.

STOP AND THINK: What would you estimate to be the chances of there being no matching birthdays in a group of 25 people (assuming birthdays are independent and all days are equally likely for each person)? Your answer is based on 100 simulations. Should your answer match the exact probability of no matches? Estimate the probability that the closest pair of birthdays in such a sample differ by three or more days.

Remember to copy your Session window into a Word document, then paginate, save and print it.

LAB REPORT: Write a report on the birthday problem. What is it? How did we study it in this lab? What assumptions have we made? What is your estimate of the chances that, given 25 people, none will have the same birthday? Why is your answer only an estimate? What do you estimate to be the chances that the two closest birthdays are 2 days apart? 3 days? 4 days? More than 4 days? or 3, or 4, or 5? (As always, annotate and append your computer output, and cross-reference it in your report.)