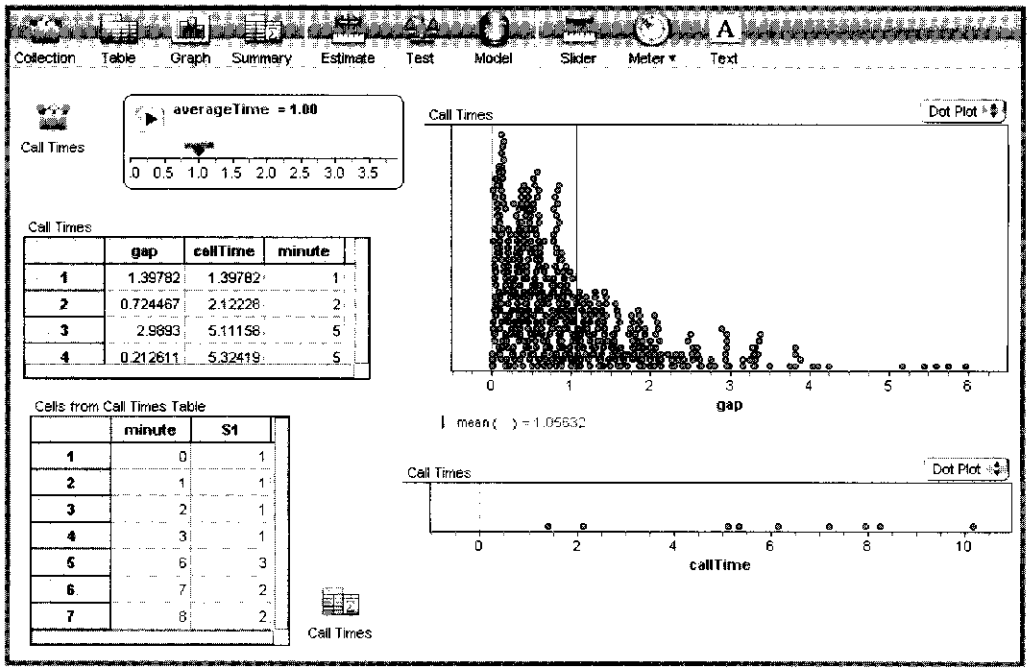
Demo 48: The Exponential and Poisson Distributions

The continuous analog to the geometric distribution • How many events happen in a given period: a Poisson distribution

Suppose phone calls come in at random times, on the average once per minute. How many seconds will it be until the first one? On the average, 60—but what is the distribution?

Based on our experience with the geometric distribution, we could say that the probability of a call during the first second is 1/60. Using that, we could simulate it one second at a time. But what if we could time the ring to the nearest 1/10 of a second? Then we would use a probability of 1/600, and use 1/10 second as our time step. Can you see where this is going? There must be a way to make this discrete phenomenon—repeated trials with a constant probability—continuous. And there is; instead of the geometric distribution, we'll use its continuous cousin, the exponential.



What To Do

▹ Open **Exponential and Poisson.ftm.** It will look something like the illustration.

Since the calls are independent, the distribution of times *between* calls is the same as the distribution of times of the *first* call: exponential.

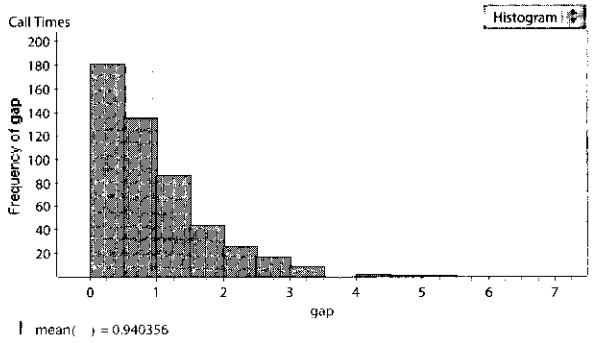
Let's concentrate on the upper half of the window. The slider **averageTime** controls the average time between calls. The **Call Times** table below it shows the first 4 of 500 calls. Attributes are the **gap** (the time between calls); the **callTime** in minutes, which is the time in decimal minutes that this call came in; and **minute,** which is the integer part of **callTime,** so it tells us *which minute* that call came in. The big graph shows the gaps, whose distribution decreases exponentially—hence the name of the distribution. Notice that, while the average time between calls is 1, most of the gaps are shorter. Below the graph is another one showing the times of the first few minutes' calls.

One last word of warning before we begin: Fathom is doing a lot of computation here, so be patient. It can take *several seconds* (it seems much longer) for the display to update, even on a fast computer. Don't think prematurely that you've crashed. Also: You will save yourself a lot of grief in this simulation if, when you

want to change the slider value **averageTime**, you do so by editing the value, *not* dragging the pointer with the hand.

- Click on the **Call Times** collection in the upper-left corner to select it.

- Choose **Rerandomize** from the **Collection** menu a few times (unless you have a fast computer, waiting for the display to update) to see how the distribution looks.
- Change the distribution to a histogram by choosing **Histogram** from the pop-up menu in the upper-right corner of the graph. Note: You can always rescale the axes by rechoosing **Histogram.**

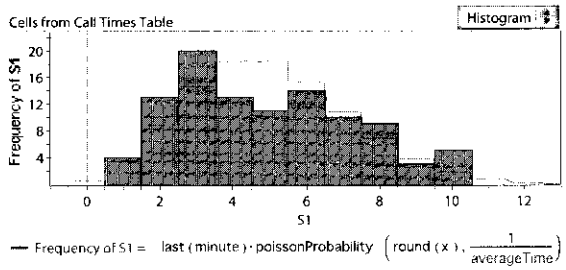


Now let's look at the machinery in the lower left. There is an iconified summary table and case table derived from it (**Cells from Call Times Table**), which shows what we want to see. It shows how many calls came in during each minute. In the illustration on the previous page, you can see that one call came in during the first minute (minute zero), and one during the second, and so forth. Note that no calls came in during minute 4. You can see that gap in the short, wide **callTime** graph and, if you wish, in the upper case table (you will have to scroll down). By this point your particular display will be different from the illustration; find analogous places in your current data.

- Change the value of **averageTime** to **2.0.** Remember to edit the value and then press **Tab** or **Enter,** given the slow response. Note how the distribution changes. Also see how the call times in the lower graph get farther apart.
- Now change the value of **averageTime** to **0.5.** Again, see how the distribution changes—and how the calls are bunched together. Try other values as you see fit.
- **Rerandomize** a few more times. (The upper collection must be selected.) Watch the lower graph and the table below to see how the gaps map onto the positions of the calls.

Let's move on. Now we're wondering, if calls come in on the average 5 per minute, how often will I get 10 calls in the same minute?

- Set **averageTime** at 0.20. Since we have 500 calls, that will be about 100 minutes.
- Now—for screen space—delete that lower graph. (Delete the graph by clicking once, then press the **Delete** key or choose **Delete Graph** from the **Edit** menu.)
- Choose **Show Hidden Objects** from the **Object** menu. A new graph appears, looking (after rescaling) something like the one in the illustration.



This graph shows something hard to grasp: the distribution of the numbers of calls that came in during the minutes we were studying. That is, during 20 of the minutes, exactly 3 calls came in. During 5 of the minutes, 10 calls came in. There may even have been a minute or two when zero calls came in, but since those minutes never appeared in our data, Fathom cannot display them.

We have plotted the theoretical distribution for this situation—a *Poisson* distribution—which also gives you an idea of how many minutes had zero calls.

- Change **averageTime** to various values such as 0.5, 1.0, and 2.0. You may need to rescale both histograms by choosing **Histogram** from their respective pop-up menus.

Questions

1 Where did the 0.20 come from when we were simulating five calls per minute?

2 In the situation of 5 calls per minute, after what fraction of the calls (out of 500 calls) was there a gap—the time between calls—of six seconds or less? (Six seconds is 0.1 minute.) **Sol**

3 If we average one call per minute (so we'll be simulating about 500 minutes), in about how many minutes do you expect to get zero calls? Remember, you'll have to think theoretically—or be sneaky—as Fathom will not display the data. **Sol**

4 Same question at one call every two minutes.

Challenges

5 Explain why the theoretical distribution for this situation—the Poisson distribution—has to have a tail that goes off to the right.

6 Study the Fathom simulation and explain why it's so much faster when **averageTime** is small than when it's large.