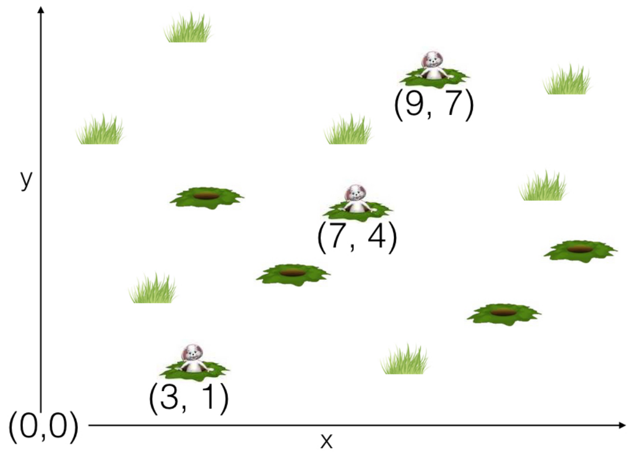
Midterm Review



1. Suppose we are scientists who study rabbits (bunny biologists), and we're interested in the burrowing behavior of the rabbits in a field. Specifically, we would like to know the *smallest distance* between any of the rabbit holes, among all pairs of holes. The holes are hidden, so we can't see them directly, but we can see the rabbits when they poke their heads out. By spraying the field with a carrot-based perfume, we entice some of the rabbits to poke their heads out of their holes, and we note the locations of those holes. Suppose these locations constitute a random sample without replacement of 30 of the hole locations, out of 100 total holes.

   We record these locations in a table called `hole_location_sample`. It has two columns: `x`, the `x` coordinate of each hole, and `y`, the `y` coordinate of each hole. The distance between two holes can be calculated using their coordinates and the Pythagorean theorem.

   (a) Using only our sample, what would be an appropriate statistic to use as an estimate of the smallest distance between any two rabbit holes in the field? (For example, "the average distance between holes in the sample" would be a (poor) estimate.)

   (b) If you wanted to know how well your estimate worked, you could repeat our sampling process 1,000 times and make a histogram of the resulting 1,000 estimates. What would that histogram be called?

   (c) Suppose the smallest distance between any two holes in the field is 10 meters. Would you expect the histogram in (b) to have its mean at roughly 10 meters, or less than 10 meters, or more than 10 meters? Draw what you think it would look like.

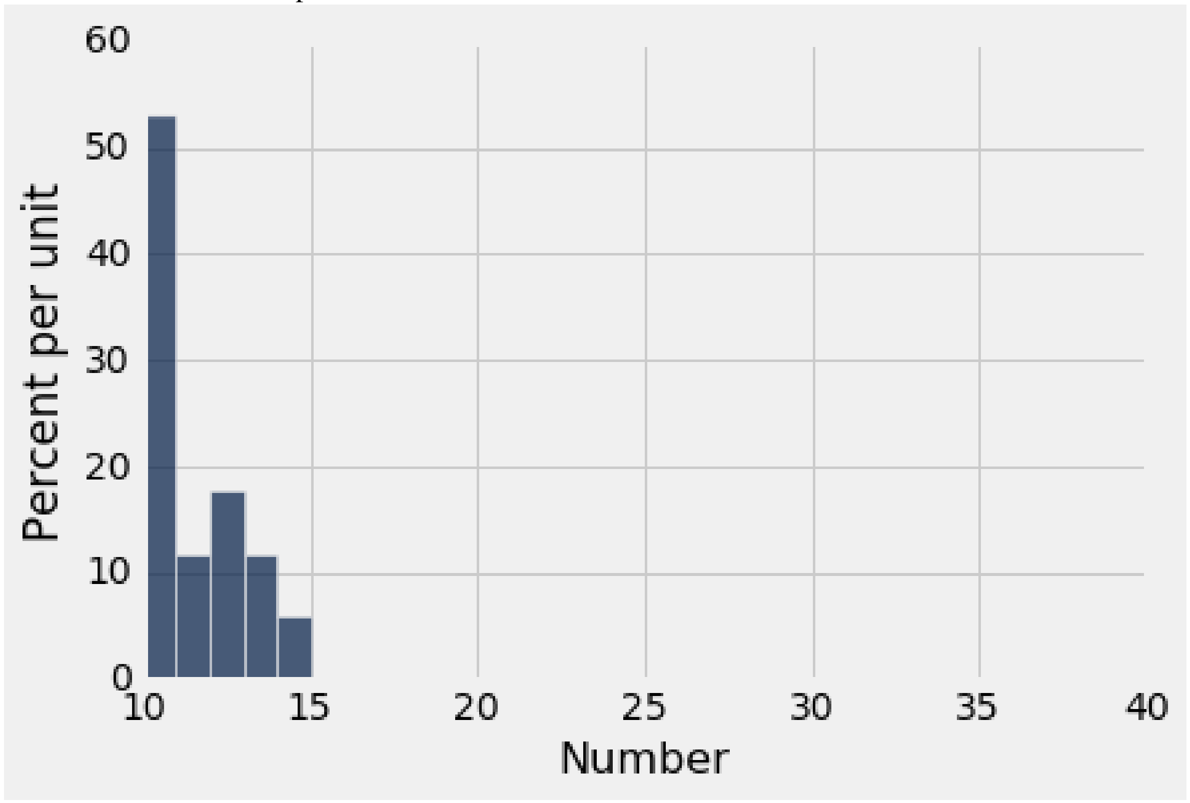
   (d) As Patrick Star once said, "We're not cavemen! We have technology!" Suppose we have access to *all 100* of the hole locations in a table called `hole_locations`, and a function called `estimate_smallest_distance` that computes your proposed estimate on 1 sample (which is comprised of picking 30 random holes). Write code that will generate the histogram in (b).

Answers:

(a) Remember that when we estimate a population mean using a sample, one straightforward approach is to use the sample's mean. Analogously, here we will use compute the minimum distance between any two holes *in our sample* and take that as our estimate. The idea of taking what you'd do if you had the whole population and doing that with a sample is called the "plug-in principle." It's not always the best idea (imagine if you wanted to estimate the population size!), but it's simple and sometimes it works okay.

(b) This would be an empirical histogram.

(c) Since the smallest distance between any pair of holes in the field is 10, then the smallest distance between any pair of holes in a sample can't be less than that. So if we use the plug-in estimator described in 1(a), we can't have estimates below 10, so the histogram can't have anything below 10. Sometimes our answers will be wrong, so some mass will go above 10, so the center can't be at 10. We could draw a histogram with a peak at 10 and a small right tail. *But* the shape of the histogram actually depends on the field. For example, imagine all of the holes are 100 or more meters apart from each other, except one pair that are 10 meters apart. If we don't sample the close pair, then our estimate will be 100 or more. So our histogram will look very spiky – there will be a little mass at 10, and nothing from 10 to 100, and then a lot of mass at 100 and higher. If the holes are more regularly spaced, the histogram will look less spiky. The point is that our estimate is not a sample mean, so the central limit theorem doesn't apply, and we shouldn't assume the empirical histogram of estimates will be bell-shaped, or even have some other nice shape.



(d)
```
# Make an empty table to hold the estimates:
# Repeat 1000 times:
estimates = make_array()
for i in np.arange(0, 1000, 1):
    # Simulate one sample from the 100 holes:
    simulated_sample = hole_locations.sample(30)
    # Compute our estimate for that sample, using the function
    # we assumed we'd already written:
    estimate = estimate_smallest_distance(simulated_sample)
    estimates = np.append(estimates, estimate)
estimates = Table().with_column("estimate", estimates).hist()
```

2. The Data 8 staff decide that they would like to get a sense of whether or not tutoring is an effective way to raise the scores of students. They keep track of the students who do two on one tutoring and compare their midterm scores to those that do not participate in tutoring. They find that on average, students who participated in tutoring sessions did 25 percent better on the midterm than students who did not.

   (a) Does this study have a treatment group and a control group? If so, name them.

   (b) Is this an observational study, a randomized controlled experiment, or neither?

   (c) The staff are convinced that the tutoring sections are causing the increase in midterm scores. Do you agree or disagree? Why or why not? If you agree, justify your answer. If not, give a specific reason as to why.

   Answers:

   (a) Yes, the treatment group is those who are receiving the mentoring and the control group is those who are not.

   (b) Observational Study, the staff is simply watching people and is not messing with the experience at all.

   (c) No, many confounding factors exist. One could be that students who are going to tutoring are more motivated to study and try and do well in the class, so they spend more hours outside of tutoring studying for the midterm. In general, causation can not be determined from an observational study.

3. Assume we are given a table called `scores` which has 10 columns , the first one containing the names of students, and the next nine containing their scores on quizzes 1-9 (in order). Design a function called `performance` which takes in a person's name and outputs a table with two columns, one for the quiz number and the other with a boolean value of either `true` or `false` indicating whether the student performed better than average on that quiz.
   Hint: Try using a `for` loop.

Answers:

```
def performance(name):
    quizes = make_array()
    scores = make_array()
    personal = scores.where('name', name)
    for x in range(1,10):
        score = personal.column(x).item(0)
        avg = np.mean(scores.column(x))
        better = score > avg
        quizes = np.append(quizes, x)
        scores = np.append(scores, better)
    t = Table.with_columns([
        'Quiz Number', quizes,
        'Better than Avg?', scores])
    ])
    return t
```

4. Let's say we have used the above function and assigned `Marissa` to the call `performance('Marissa')`. Using the table we now have, how can we use code to figure out how many tests Marissa did better than average on? There should be a relatively easy and concise solution.
   Answers:
   `np.count_nonzero(marissa.column(1))`