Conditional Probability

This lab is an introduction to visualizing conditional probabilities. We will cover *icon arrays*. These do not appear in the textbook and will not appear on any exam, but they will help you gain intuition about conditional probability.

Administrative details

This lab will not be collected. Conditional probability will appear on the final exam, and this is an opportunity to understand it better. **We recommend going through at least part 2.** You can complete the rest later as an exercise when you're studying.

In [1]:
```
# Run this cell to set up the notebook, but please don't change it.

# These lines import the Numpy and Datascience modules.
import numpy as np
from datascience import *

# These lines do some fancy plotting magic.
import matplotlib
%matplotlib inline
import matplotlib.pyplot as plt
plt.style.use('fivethirtyeight')
import warnings
warnings.simplefilter('ignore', FutureWarning)

# This line loads the visualization code for this lab.
import visualizations

# These lines load the tests.
from client.api.assignment import load_assignment
tests = load_assignment('lab10.ok')
```

1. What is conditional probability good for?

Suppose we have a known population, like all dogs in California. So far, we've seen 3 ways of *predicting* something about an individual in that population, given incomplete knowledge about the identity of the individual:

- If we know nothing about the individual dog, we could predict that its speed is the *average* or *median* of all the speeds in the population.
- If we know the dog's height but not its speed, we could use *linear regression* to predict its speed from its height. The resulting prediction is still imperfect, but it might be more accurate than the population average.
- If we know the dog's breed, height, and age, we could use *nearest-neighbor classification* to predict its speed by comparing to a collection of dogs with known speed.

Computing conditional probabilities is a different way of making predictions. It differs in at least two important ways from the methods we've seen:

1. Rather than producing a single answer that might be wrong, we just figure out how likely each possible answer is.
2. In the simple (but important) cases we'll look at today, conditional probabilities can be calculated exactly from assumptions rather than estimated from data. By contrast, there are many techniques for classification, and even once we choose `k`-Nearest Neighbors, we get different results for different values of `k`.

2. Icon arrays

Parts 3 and 4 of this lab are about cancer, but first let's start with a simple, contrived example.

Imagine you are a marble. You don't know what you look like (since you obviously have no eyes), but you know that Samantha drew you **uniformly at random** from a bag that contained the following marbles:

- 4 large green marbles,
- 1 large red marble,
- 6 small green marbles, and
- 2 small red marbles.

Question 2.1. Knowing only what we've told you so far, what's the probability that you're a large green marble?

```
In [2]: probability_large_green = ...
```

```
In [3]: _ = tests.grade("q21")
```

Here's a table with those marbles:

```
In [4]: marbles = Table.read_table("marbles.csv")
        marbles
```

We've included some code to display something called an *icon array*. The functions in the cell below create icon arrays from various kinds of tables. Refer back to this cell later when you need to make an icon array.

In [5]:
```
# Run this cell.

#######################################################################
# The functions you'll need to actually use are in here.  Each is a
# way of making an icon array from a differently-formatted table.
#######################################################################

def display_icon_array(table, groups, individuals_name):
    """
    Given a table and some columns to group it on, displays an icon arra
y
    of the groups.

    groups should be an array of labels of columns in table.

    individuals_name is your name for the individual rows of table.
    For example, if we're talking about a population of people,
    individuals_name should be "people".

    For example:

    display_icon_array(marbles, make_array("color", "size"), "marbles")
    """
    display_grouped_icon_array(table.groups(groups), individuals_name)

def display_grouped_icon_array(grouped_data, individuals_name):
    """
    Given a table with counts for data grouped by 1 or more categories,
    displays an icon array of the groups represented in the table.

    grouped_data should be a table of frequencies or counts, such as
    a table created by calling the groups method on some table.

    individuals_name is your name for the individual members of the
    dataset.  For example, if we're talking about a population of
    people, individuals_name should be "people".

    For example:

    display_grouped_icon_array(marbles.groups(make_array("color", "siz
e")), "marbles")
    """
    visualizations.display_combinations(grouped_data, individuals_name=i
ndividuals_name)

def display_crosstab_icon_array(crosstabulation, x_label, individuals_na
me):
    """
    Given a crosstabulation table, displays an icon array of the groups
    represented in the table.

    crosstabulation should be a table of frequencies or counts created b
y
    calling pivot on some table.

    x_label should be the label of the categories listed as columns (on
    the "x axis" when the crosstabulation table is printed).

    individuals_name is your name for the individual members of the
    dataset.  For example, if we're talking about a population of
    people, individuals_name should be "people".

    For example:

    display_crosstab_icon_array(marbles.pivot("color", "size"), "color",
 "marbles")
    """
    display_grouped_icon_array(visualizations.pivot_table_to_groups(cros
stabulation, x_label), individuals_name)
```

Here's an icon array of all the marbles, grouped by color and size:

```
In [10]: # Run this cell.
         display_grouped_icon_array(marbles.groups(make_array("color", "size")),
         "marbles")
```

Note that the icon colors don't correspond to the colors of the marbles they represent.

You (the marble) should imagine that you are a random draw from these 13 icons.

Question 2.2. Make an icon array of the marbles, grouped **only by color**.

```
In [6]: ...
```

Knowing nothing else about yourself, you're equally likely to be any of the marbles pictured.

Question 2.3. What's the probability that you're a green marble? Calculate this by hand (using Python for arithmetic) by looking at your icon array.

```
In [7]: probability_green = ...
```

```
In [8]: _ = tests.grade("q23")
```

2.1. Conditional probability

Suppose you overhear Samantha saying that you're a large marble. (Little-known fact: though marbles lack eyes, they possess rudimentary ears.) Does this somehow change the likelihood that you're green? Let's find out.

Go back to the full icon array, displayed below for convenience.

In [9]:
```
display_grouped_icon_array(marbles.groups(make_array("color", "size")),
"marbles")
```

In question 2.3, we assumed you were equally likely to be any of the marbles, because we didn't know any better. That's why we looked at all the marbles to compute the probability you were green.

But assuming you're a large marble, we can eliminate some of these possibilities. In particular, you can't be a small green marble or a small red marble.

You're still equally likely to be any of the remaining marbles, because you don't know anything that says otherwise. So here's an icon array of those remaining possibilities:

In [10]:
```
# Just run this cell.
display_grouped_icon_array(marbles.where("size",
"large").group("color"), "large marbles")
```

Question 2.1.1. What's the probability you're a green marble, knowing that you're a large marble? Calculate it by hand, using the icon array.

In [11]:
```
probability_green_given_large = ...
```

In [12]:
```
_ = tests.grade("q211")
```

You should have found that this is different from the probability that you're a green marble, which you computed earlier. The distribution of colors among the large marbles is a little different from the distribution of colors among all the marbles.

Question 2.1.2. Suppose instead Samantha had said you're a **green** marble. What's the probability you're large? Make an icon array to help you compute this probability, then compute it.

Hint: Look at the code we wrote to generate an icon array for question 2.1.1.

```
In [13]: # Make an icon array to help you compute the answer.
         ...

         # Now compute the answer.
         probability_large_given_green = ...
```

```
In [14]: _ = tests.grade("q212")
```

Question 2.1.3. How could you answer the last two questions just by looking at the full icon array? (You can run the cell below to see it again.)

```
In [15]: # Just run this cell.  The next cell is where you should write your answ
         er.
         display_grouped_icon_array(marbles.groups(make_array("color", "size")),
         "marbles")
```

Write your answer here, replacing this text.

3. Cancer screening

Now let's look at a much more realistic application.

Background

Medical tests are an important but surprisingly controversial topic. For years, women have been advised to get regular mammograms (tests for breast cancer). Today, there is controversy over whether the tests are useful at all.

Part of the problem with such tests is that they are not perfectly reliable. Someone without cancer, or with only a benign form of cancer, can see a positive result on a test for cancer. Someone with cancer can receive a negative result. ("Positive" means "pointing toward cancer," so in this context it's bad!) Doctors and patients often deal poorly with the first case, called *false positives*. For example, a patient may receive dangerous treatment like chemotherapy or radiation despite having no cancer or, as happens more frequently, having a cancer that would not have impacted her health.

Conditional probability is a good way to think about such situations. For example, you can compute the chance that you have cancer, given the result of a test, by combining information from different probability distributions. You'll see that the chance you have cancer can be far from 100% even if you have a positive test result from a test that is usually accurate.

3.1. Basic cancer statistics

Suppose that, in a representative group of 10,000 people who are tested for cancer ("representative" meaning that the frequencies of different things are the same as the frequencies in the whole population):

1. 100 have cancer.
2. Among those 100, 90 have positive results on a cancer test and 10 have negative results. ("Negative" means "not pointing toward cancer.")
3. The other 9,900 don't have cancer.
4. Among these, 198 have positive results on a cancer test and the other 9,702 have negative results. (So 198 see "false positive" results.)

Below we've generated a table with data from these 10,000 hypothetical people.

```
In [24]: people = Table().with_columns(
             "cancer status", make_array("sick", "sick", "healthy", "healthy"),
             "test status", make_array("positive", "negative", "positive", "negat
         ive"),
             "count", make_array(90, 10, 198, 9702))
         people
```

One way to visualize this dataset is with a contingency table, which you've seen before.

Question 3.1.1. Create a contingency table that looks like this:

cancer status	negative	positive
sick		
healthy		

...with the **count** of each group filled in, according to what we've told you above. The counts in the 4 boxes should sum to 10,000.

Hint: Use `pivot` with the `sum` function.

```
In [25]: cancer = ...
         cancer
```

```
In [26]: _ = tests.grade("q311")
```

Question 3.1.2. Display the `people` data in an icon array. The name of the population members should be "people who've taken a cancer test".

```
In [27]: ...
```

Now let's think about how you can use this kind of information when you're tested for cancer.

Before you know any information about yourself, you could imagine yourself as a **uniform random sample** of one of the 10,000 people in this imaginary population of people who have been tested.

What's the chance that you have cancer, knowing nothing else about yourself? It's $\frac{100}{10000}$, or 1%. We can see that more directly with this icon array:

In [29]:
```
by_health = people.select(0, 2).group(0, sum).relabeled(1, 'count')
display_grouped_icon_array(by_health, "people who've taken a cancer tes
t")
```

Question 3.1.3. What's the chance that you have a positive test result, knowing nothing else about yourself?

Hint: Make an icon array.

In [30]:
```
# We first made an icon array in the 2 lines below.
by_test = ...
display_grouped_icon_array(by_test, "people who've taken a cancer test")

# Fill in the probabiliy of having a positive test result.
probability_positive_test = ...
```

In [31]:
```
_ = tests.grade("q313")
```

3.2. Interpreting test results

Suppose you have a positive test result. This means you can now narrow yourself down to being part of one of two groups:

1. The people with cancer who have a positive test result.
2. The people without cancer who have a positive test result.

Here's an icon array for those two groups:

In [32]:
```
# Just run this cell.
display_grouped_icon_array(people.where("test status", are.equal_to("pos
itive")).drop(1), "people who have a positive test result")
```

The *conditional probability* that you have cancer *given* your positive test result is the chance that you're in the first group, assuming you're in one of these two groups.

Question 3.2.1. Eyeballing it, is the conditional probability that you have cancer given your positive test result closest to:

1. 9/10
2. 2/3
3. 1/2
4. 1/3
5. 1/100

In [33]:
```
# Set this to one of the numbers above.
rough_prob_sick_given_positive = ...
```

In [34]:
```
_ = tests.grade("q321")
```

Question 3.2.2. Now write code to calculate that probability exactly, using the original contingency table you wrote.

In [35]:
```
prob_sick_given_positive = ...
prob_sick_given_positive
```

In [36]:
```
_ = tests.grade("q322")
```

Question 3.2.3. Look at the full icon array again. Using that, how would you compute (roughly) the conditional probability of cancer given a positive test?

In [37]:
```
# The full icon array is given here for your convenience.
# Write your answer in the next cell.
display_grouped_icon_array(people, "people who've taken a cancer test")
```

Write your answer here, replacing this text.

Question 3.2.4. Is your answer to question 3.2.2 bigger than the overall proportion of people in the population who have cancer? Does that make sense?

Write your answer here, replacing this text.

4. Tree diagrams

A tree diagram is another useful visualization for conditional probability. It is easiest to draw a tree diagram when the probabilities are presented in a slightly different way. For example, people often summarize the information in your `cancer` table using 3 numbers:

1. The overall probability of having cancer is **`X`**. (This is called the *base rate* or *marginal probability* of the disease.)
2. Given that you have cancer, the probability of a positive test result is **`Y`**. (This is called the *sensitivity* of the test. Higher values of `Y` mean the test is more useful.)
3. Given that you don't have cancer, the probability of a positive test result is **`Z`**. (This is called the *false positive rate* of the test. Higher values of `Z` mean the test is less useful.)

This corresponds to this tree diagram:

```
          /\
         /  \
    1-X /    \ X
       /      \
  no cancer   cancer
    / \1-Z     / \
Z  /   \     Y/   \ 1-Y
  /     \    /     \
 +       -  +       -
```

You already saw that the base rate of cancer (which we'll call `X` for short) was .01 in the previous section. `Y` and `Z` can be computed using the same method you used to compute the conditional probability of cancer given a positive test result.

Question 4.1. Compute `Y` and `Z` for the data in section 3. You can use an icon array or compute them only with code. You can run the tests to see the right answers.

In [41]:
```
# Hint: You may find these two tables useful:
has_cancer = cancer.where("cancer status", are.equal_to("sick"))
no_cancer = cancer.where("cancer status", are.equal_to("healthy"))

X = .01
Y = ...
Z = ...

print('X:', X, ' Y:', Y, ' Z:', Z)
```

In [42]:
```
_ = tests.grade("q41")
```

In [ ]:
```
# For your convenience, you can run this cell to run all the tests at on
ce!
import os
_ = [tests.grade(q[:-3]) for q in os.listdir("tests") if
q.startswith('q')]
```

In [ ]:
```
# Run this cell to submit your work *after* you have passed all of the t
est cells.
# It's ok to run this cell multiple times. Only your final submission wi
ll be scored.

!TZ=America/Los_Angeles jupyter nbconvert --output=".lab10_$(date +%m%d
_%H%M)_submission.html" lab10.ipynb
```