16. total events $= 2^4$

$= 16$

all events:

$\{\emptyset\}$	$\{0, 1\}$	$\{1, 2, 3\}$	$\{0, 1, 2, 3\}$
$\{0\}$	$\{0, 2\}$	$\{0, 1, 2\}$	
$\{1\}$	$\{0, 3\}$	$\{0, 1, 3\}$	
$\{2\}$	$\{1, 2\}$	$\{0, 2, 3\}$	
$\{3\}$	$\{1, 3\}$		
	$\{2, 3\}$		

event outcomes

	$\{0\}$	$\{1\}$	$\{2\}$	$\{3\}$	$\{\emptyset\}$
$P_B(A)$	0.4	0.3	0.2	0.1	0
$P_G(A)$	0.25	0.35	0.25	0.15	0
$\lvert P_B(A) - P_G(A)\rvert$	0.15	0.05	0.05	0.05	0

	$\{0, 1\}$	$\{0, 2\}$	$\{0, 3\}$	$\{1, 2\}$
$P_B(A)$	$0.4 + 0.3$	$0.4 + 0.2$	$0.4 + 0.1$	$0.3 + 0.2$
$P_G(A)$	$0.25 + 0.35$	$0.25 + 0.25$	$0.25 + 0.15$	$0.35 + 0.25$
$\lvert P_B(A) - P_G(A)\rvert$	$\lvert 0.7 - 0.6\rvert$	$\lvert 0.6 - 0.5\rvert$	$\lvert 0.5 - 0.4\rvert$	$\lvert 0.5 - 0.6\rvert$
	0.1	0.1	0.1	0.1

	$\{1, 3\}$	$\{2, 3\}$	$\{1, 2, 3\}$	$\{0, 1, 2\}$
$P_B(A)$	$0.3 + 0.1$	$0.2 + 0.1$	$0.3 + 0.2 + 0.1$	$0.4 + 0.3 + 0.2$
$P_G(A)$	$0.35 + 0.15$	$0.25 + 0.15$	$0.35 + 0.25 + 0.15$	$0.25 + 0.35 + 0.25$
$\lvert P_B(A) - P_G(A)\rvert$	$\lvert 0.4 - 0.5\rvert$	$\lvert 0.3 - 0.4\rvert$	$\lvert 0.6 - 0.75\rvert$	$\lvert 0.9 - 0.85\rvert$
	0.1	0.1	0.15	0.05

	$\{0, 1, 3\}$	$\{0, 2, 3\}$	$\{0, 1, 2, 3\}$
$P_B(A)$	$0.4 + 0.3 + 0.1$	$0.4 + 0.2 + 0.1$	$0.4 + 0.3 + 0.2 + 0.1$
$P_G(A)$	$0.25 + 0.35 + 0.15$	$0.25 + 0.25 + 0.15$	$0.25 + 0.35 + 0.25 + 0.15$
$\lvert P_B(A) - P_G(A) \rvert$	$\lvert 0.8 - 0.75 \rvert$	$\lvert 0.7 - 0.65 \rvert$	$\lvert 1 - 1 \rvert$
	0.05	0.05	0

events with biggest abs. difference

$P(0.15) : \{0\}, \{1, 2, 3\}$

$\therefore$ same as TVD computed in 1a

2a. When n is large, the probabilities $P(M_n = 0)$ and $P(M_n = 1)$ are very close because they depend on $P(M_n = k) = (1 - \frac{1}{1!} + \frac{1}{2!} + \dots + (-1)^{n-k} \frac{1}{(n-k)!})$. For $k = 0$, $\frac{1}{n!}$ and for $k = 1$, $\frac{1}{(n-1)!}$. These two values are similar as n grows. For example, when $n = 100$, these values approximate to 0.01 and 0.010101. So, as n approaches a large number, such as infinity, both values will approach 0.

Lab_02

February 5, 2025

Probability for Data Science

UC Berkeley, Spring 2025

Michael Xiao and Ani Adhikari

[55]:
```
# SETUP
import warnings
warnings.filterwarnings('ignore')

import numpy as np
from datascience import *
from prob140 import *

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

# Useful for probability calculations
from scipy import stats
from scipy import special
```

0.0.1 Instructions

Similar to your homeworks, your labs will generally have two components: a written portion and a portion that also involves code.
- Written work should be completed on paper, and coding questions should be done in the notebook.
- Start the work for the written portions of each section on a new page. - You are welcome to LaTeX your answers to the written portions, but staff will not be able to assist you with LaTeX related issues. - Show your work. Give reasoning. The question isn't always going to ask for it, because we assume that you will provide justification for your answers. Every answer should contain a calculation, reasoning, or diagrams that are clearly labeled to show what's going on. - It is your responsibility to ensure that both components of the lab are submitted completely and properly to

Gradescope. **Make sure to assign each page of your pdf to the correct question. - Refer to the bottom of the notebook for submission instructions.**

1 Lab 2: Total Variation (Due Monday, February 3rd at 5 PM)

1.1 Lab Resources

- `prob 140` Library Documentation
- `datascience` Library Documentation
- Prob 140 Code Reference Sheet

In Data 8, you measured the difference between two categorical distributions by calculating the total variation distance (TVD) between them. The bigger the distance, the less similar the two distributions are. But in Data 8, we didn't discuss some obvious questions about the TVD, such as:

- How big could the TVD between two distributions be? Could it be arbitrarily large, like the distance between two points on the number line?
- Is there any way to interpret the numerical value of the TVD? For example, if we know that the TVD between two distributions is 0.02, what does that tell us?

In this lab, you will start by interpreting the total variation distance in terms of probabilities. You will then use the TVD to measure the difference between two probability distributions on the non-negative integers. In Data 8 terms, you can think of each non-negative integer as a "category".

The main point of the lab is that the total variation distance will give you a precise way to quantify how well one probability distribution approximates another.

Our focus will be on *Poisson* distributions, which are often used as approximations to distributions of counts of rare events. In particular, the Poisson (1) distribution approximates the distributions of some random counts that have 0 and 1 as their most likely values.

A random variable X has the Poisson (1) distribution if

$$P(X = k) ~=~ e^{-1}\frac{1}{k!}, ~~~~ k \ge 0$$

This is a probability distribution on infinitely many possible values. We will need that infinite support if we are going to approximate distributions on the values $0, 1, 2, \ldots, n$ for arbitrarily large n.

In class we are studying two situations in which probabilities approach those in a Poisson distribution. One is counting the number of matches in the matching problem with n letters, when n is large. The other is counting the number of successes in n i.i.d. Bernoulli (p) trials, for large n and small p.

In this lab you will look at the entire distribution of the number of matches, and also the binomial $(n, 1/n)$ distribution. You will compare them with their Poisson (1) approximations, both visually and also by the TVD.

In doing so, you will find an upper bound for the amount of error you can make when you use the approximations to calculate probabilities.

What you will learn: - The definition of total variation distance and its interpretation in terms of the amount of error in approximating probabilities - For large n, properties of the TVD between the distribution of the number of matches and the Poisson (1) distribution - For large n, properties of the TVD between the binomial $(n, 1/n)$ and Poisson (1) distributions

2 Lab 2 starts here. It is due by 5 PM on Monday, February 3rd.

2.1 Identify Your Lab Partner

This is a multiple choice question. Please select **ONE** of following options that best describes how you complete Lab 2.

- I am doing this lab by myself and I don't have a partner.
- My partner for this lab is [PARTNER'S NAME] with email [berkeley.edu email address]. [SUBMITTER'S NAME] will submit to Gradescope and add the other partner to the group on Gradescope after submission.

Please copy and paste **ONE** of above statements and fill in blanks if needed. If you work with a partner, make sure only one of you submit on Gradescope and that the other member of the group is added to the submission on Gradescope. Refer to the bottom of the notebook for submission instructions.

Type your answer in this cell.

I am doing this lab by myself and I don't have a partner.

2.2 Section 1: Total Variation Distance

Suppose you have two probability distributions on the same set of possible values $x_1, x_2, \dots, x_n$. Let the two distributions be $b_1, b_2, \dots, b_n$ and $g_1, g_2, \dots, g_n$, where for each i the b-distribution assigns probability b_i to the value x_i and the g-distribution assigns probability g_i.

The *total variation distance* between the two distributions is defined by

$$tvd(b, g) = \frac{1}{2}\sum_{i=1}^{n} |b_i - g_i|$$

The choice of notation comes from the blue and gold colors you will see in overlaid histograms below.

2.2.1 1a) [CODE] Computing TVD

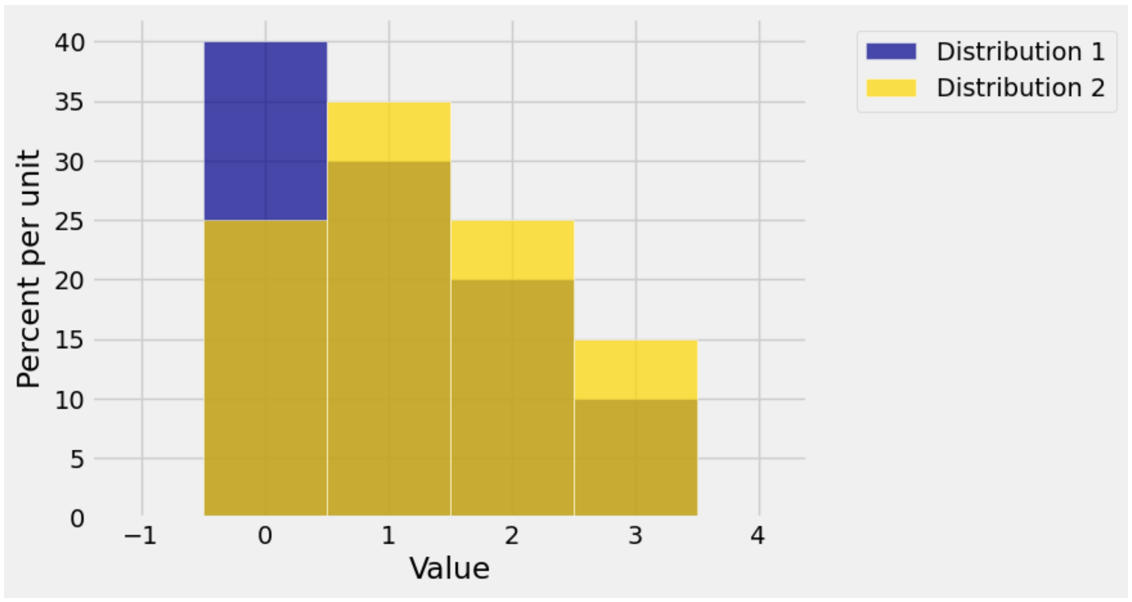
We will use the term *probability array* to mean an array of non-negative numbers that sum to 1.

Define a function `tvd` that takes two probability arrays of the same size as arguments and returns the total variation distance between them. You should just assume that both of the input arrays will be probability distributions on the same set of possible values. You don't have to include code to check that.

```
[56]: def tvd(b, g):
          return 0.5 * sum(abs(b-g))
```

When the two arrays are $b = [0.4, 0.3, 0.2, 0.1]$ and $g = [0.25, 0.35, 0.25, 0.15]$, the histograms look like this:

[57]:
```
b = make_array(0.4, 0.3, 0.2, 0.1)
g = make_array(0.25, 0.35, 0.25, 0.15)

k = np.arange(4)

blue_dist = Table().values(k).probabilities(b)
gold_dist = Table().values(k).probabilities(g)

Plots('Distribution 1', blue_dist, 'Distribution 2', gold_dist)
```



Calculate the TVD by mental math. Then run the cell below to confirm that your function `tvd` is working correctly.

[58]:
```
tvd(b, g)
```

[58]:
```
0.14999999999999999
```

The total variation distance between the two distributions is the total amount by which the areas of the blue bars exceed those of the corresponding gold bars. That's exactly equal to the total amount by which the gold bars exceed the blue.

This is almost apparent from the definition of total variation distance, and you will prove it later in this lab. Just assume it for now as you did in Data 8. It is an intuitively reasonable measure of the difference between the two distributions.

2.2.2 1b) [On Paper] Another Way of Interpreting TVD

Thus far, our interpretation of total variation distance has been essentially geometric: the amount by which the blue bars exceed the gold. There is an equivalent interpretation in terms of probabilities that makes it easier to understand what the numerical value of the distance is telling us.

Suppose you have a finite set of possible values, and a choice of two probability distributions to use for finding probabilities. For example, the choices might be the exact distribution of a random variable and an approximate distribution.

The total variation distance between the two distributions is the biggest difference you can possibly get if you compute the probability of an event using each of the two distributions.

Formally, if S is the space of all possible values, then the total variation distance between the blue and gold distributions is equal to

$$\max\left\{\left|P_{blue}(A) - P_{gold}(A)\right| : A \subseteq S\right\}$$

This can be shown in a few straightforward steps and you can do that at the end of this lab. For now, confirm that it is true for the distributions in **1a**, in the following steps.

- Figure out how many events can be created out of the outcomes $\{0, 1, 2, 3\}$.
- List all the events.
- For each event, compute the absolute difference between the blue and gold probabilities of the event. Your goal is to find the biggest possible absolute difference, so you might not even need to compute each one.
- See which event or events correspond to the biggest absolute difference, and compare the value of that absolute difference with the TVD that you computed in **1a**.

2.3 Section 2: Fixed Points of a Random Permutation

Let M_n be the number of fixed points in a random permutation of the values $1, 2, 3, \ldots, n$. You can think of M_n as the number of matches when n letters labeled 1 through n are permuted randomly into n envelopes labeled 1 through n.

In class, we found the exact and approximate probability of $P(M_n = 0)$. Before attempting this section of the lab, please read about how to find the rest of the probabilities in the distribution of M_n.

In the same textbook section, you will also find the approximation that for large n:

$$P(M_n = k) \approx \frac{e^{-1}}{k!}$$

These are the terms in the Poisson (1) distribution.

In this part of the lab you will compare the distribution of M_n with its Poisson (1) approximation.

2.3.1 2a) [CODE] Computing $P(M_n = k)$

Complete the definition of the function `prob_matches` that takes k and n as its arguments and returns $P(M_n = k)$. Use as many lines of code as you need.

Make sure you compute the exact distribution of M_n, not its Poisson approximation. To do this, use array operations. The `special` module of SciPy has been imported and contains a useful `factorial` method:

`special.factorial(integer_array)` evaluates to an array consisting of the factorials of all the integers in `integer_array`.

Be careful about signs. Follow the lead of the code provided, and tackle all the positive terms separately from all the negative terms.

[59]:
```python
def prob_matches(k, n):
    x_even = np.arange(0, n - k + 1, 2)
    x_odd = np.arange(1, n - k + 1, 2)

    even = (1 / special.factorial(x_even))
    odd = (1 / special.factorial(x_odd))

    return (1 / special.factorial(k)) * (np.sum(even) - np.sum(odd))
```

To confirm that your function is working correctly, think about what $P(M_n = n - 1)$ should be, and then run the cell below to check.

[60]:
```python
prob_matches(99, 100)
```

[60]:
```
0.0
```

Use the formula for $P(M_n = k)$ to explain why $P(M_n = 0)$ is very close to $P(M_n = 1)$ when n is large.

written on paper

Now run the cell below and confirm that your function is working correctly.

```
[61]: prob_matches(0, 100), prob_matches(1, 100)
```

```
[61]: (0.36787944117144233, 0.36787944117144233)
```

2.3.2 2b) [CODE] The Bulk of the Distribution

Use `prob_matches` to define a function `match_dist` that takes n as its argument and returns an array consisting of the probabilities $P(M_n = k)$ for $0 \leq k \leq n$.

```
[62]: def match_dist(n):
          return np.array([prob_matches(k, n) for k in range(n + 1)])
```

The expression `match_dist(100)[0:11]` evaluates to an array consisting of the elements 0 through 10 of the array `match_dist(100)`.

Explain what the output of the cell below tells you about the distribution of M_n. As with most questions about random variables, start by thinking about the possible values of M_n.

```
[63]: sum(match_dist(100)[0:11])
```

```
[63]: 0.9999999899522336
```

The output of the cell tells me the probability of correctly matched letters in a random permutation of n letters and envelopes. It represents the chance that the number of matches falls between 0 and 10. By adding up these probabilities, it is almost 100% chance that the number of matches will be within this range.

2.3.3 2c) [CODE] Visualization

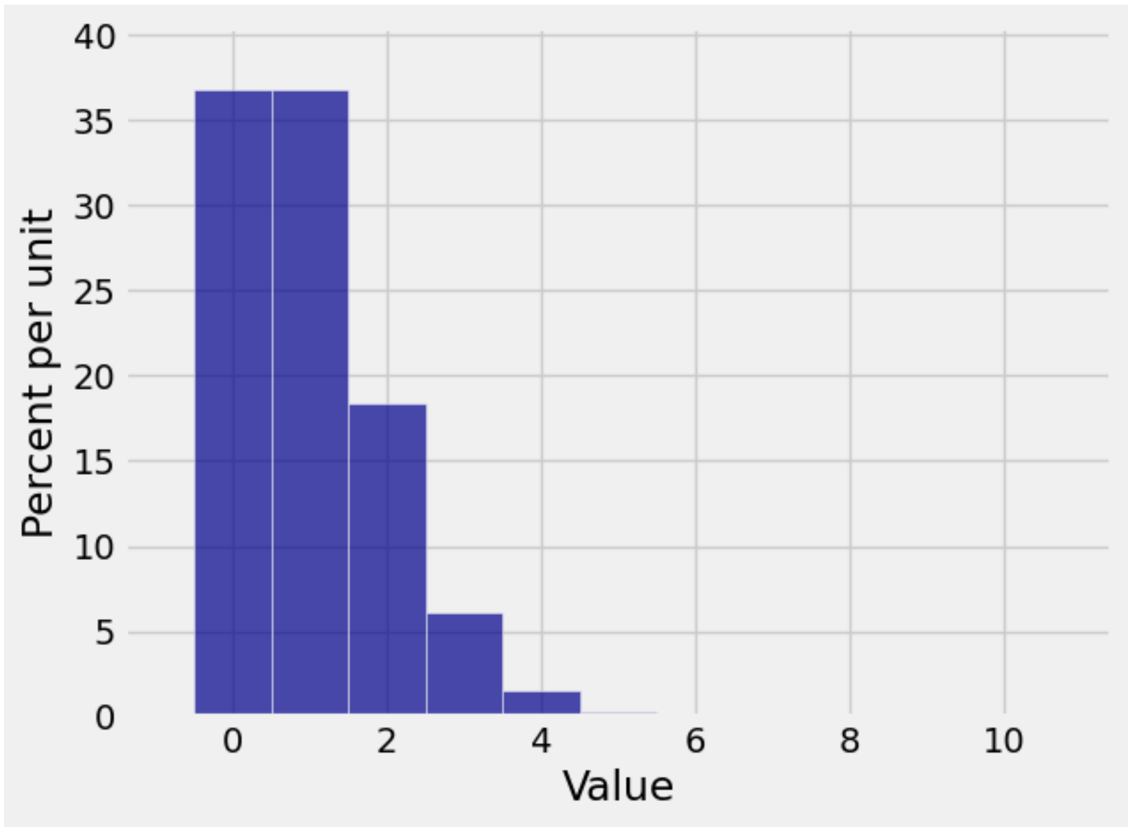
Plot the distribution of M_{100}, the number of matches in the matching problem with 100 letters. Use `Plot(dist)` where `dist` is a distribution object created by either:

`Table().values(array_of_values).probabilities(array_of_probabilities)`

or

`Table().values(array_of_values).probability_function(name_of_function)` where `name_of_function` takes a possible value as its argument and returns the probability of that value.

```
[64]: k = np.arange(11)

      array_of_probabilities = [prob_matches(i, 100) for i in k]

      matches_100 = Table().values(k).probabilities(array_of_probabilities)
      Plot(matches_100)
```


40
35
30
25
20
15
10
5
0
Percent per unit
0
2
4
6
8
10
Value

2.4 Section 3: Poisson Approximation to the Matching Distribution

Recall from the introduction to the lab that X has the Poisson distribution with parameter 1 if

$$P(X = k) \ = \ e^{-1}\frac{1}{k!}, \quad k \geq 0$$

The `stats` module of SciPy can be used to calculate the probabilities in this distribution. If `values` is an array of non-negative integers, then

`stats.poisson.pmf(values, 1)`

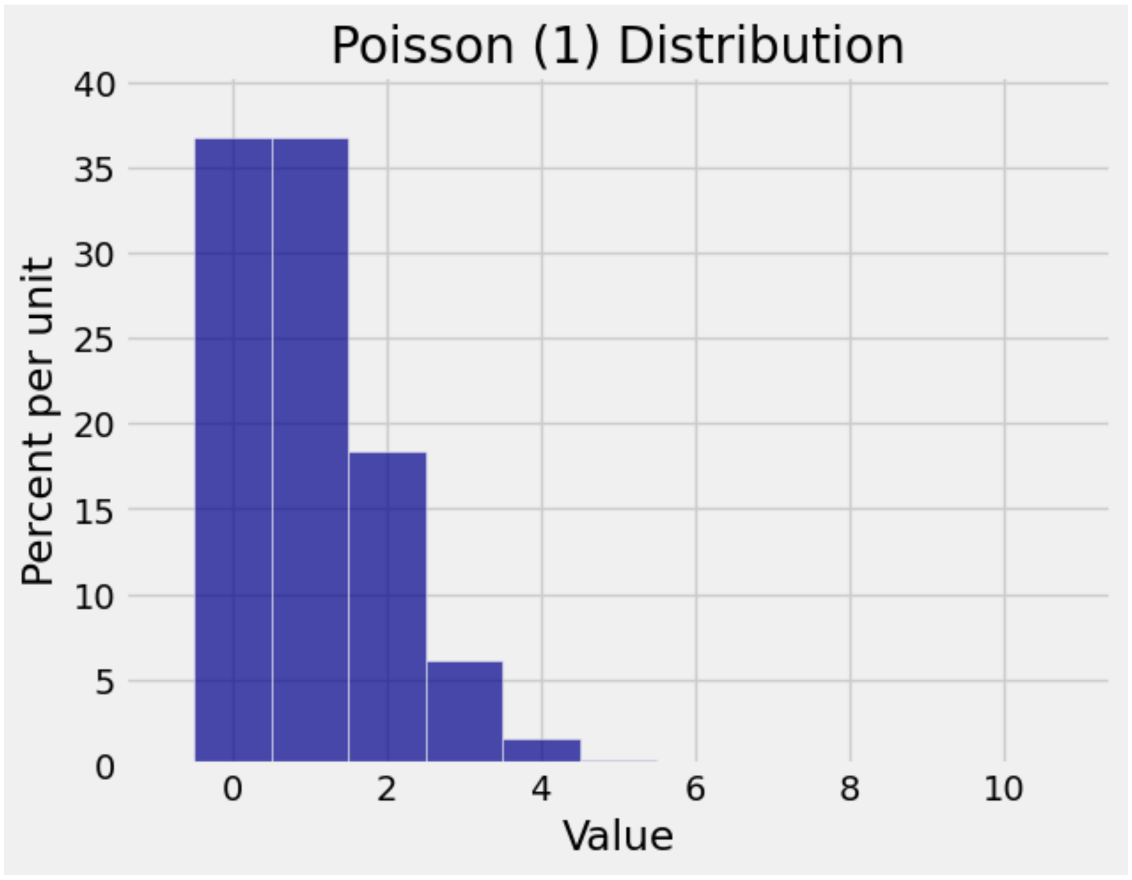
evaluates to an array of probabilities of the entries in `values`, determined by the Poisson (1) formula above.

The acronym `pmf` stands for "probability mass function". It is common for probabilists to think of the probabilities in the distribution of a discrete random variable as masses attached to the random variable's possible values.

2.4.1 3a) [CODE] The Poisson (1) Distribution

Draw a histogram of the Poisson (1) distribution.

[65]:
```
k = np.arange(11)         # selected possible values
poisson_1_probs = stats.poisson.pmf(k,1)     # array of corresponding Poisson␣
↪(1) probabilities
poisson_1_dist = Table().values(k).probabilities(poisson_1_probs)
Plot(poisson_1_dist)
plt.title('Poisson (1) Distribution');
```



Notice:

- The distribution has two modes, at 0 and 1. In exercises you will see why.
- Though there are infinitely many possible values, the set of *probable* values is very small — there is hardly any probability visible beyond the value 4.
- The distribution is extremely similar to the distribution of the number of matches in **2c**.

2.4.2 3b) [CODE] TVD Between the Matching Distribution and its Poisson Approximation

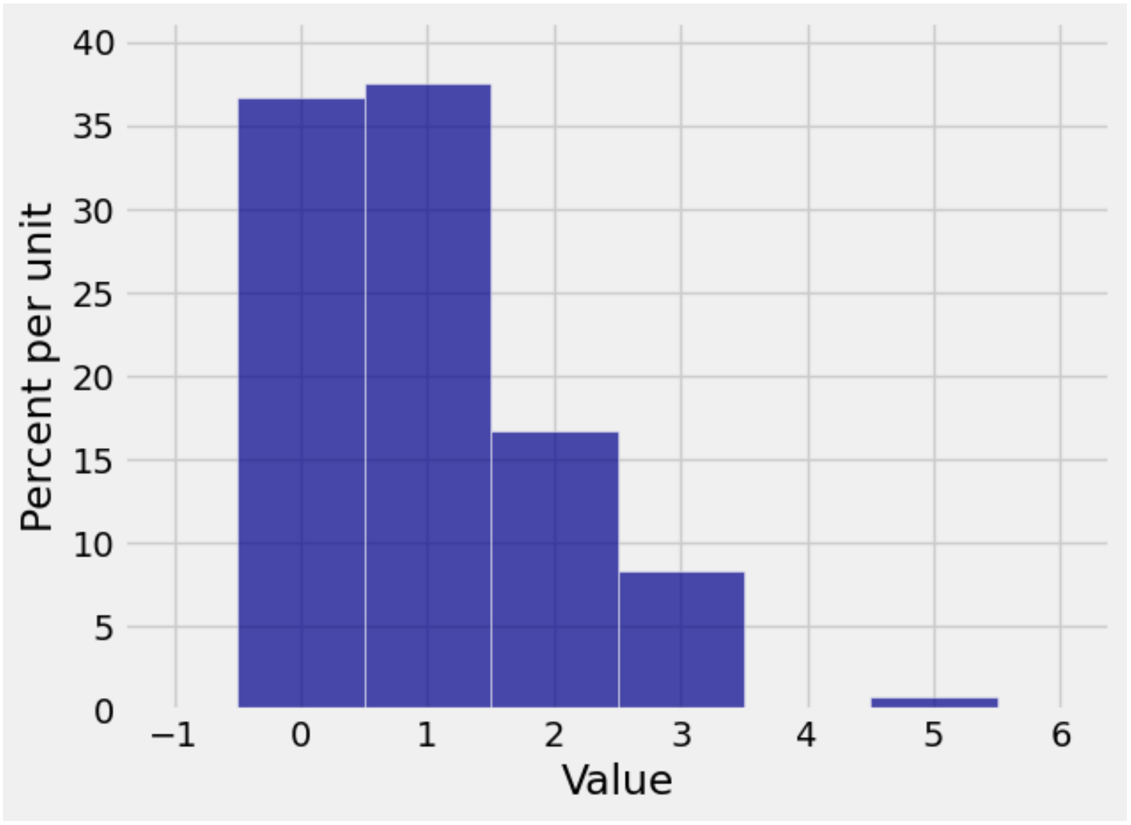
Use the `stats` module and functions you have already defined in this lab to define a new function `matches_Poisson_tvd` that takes n as its argument and returns the total variation distance between the distribution of the number of matches M_n and the Poisson (1) distribution.

Though the Poisson distribution has infinite support, you have seen that almost all of the probability is concentrated on just a few small values. So it's fine to compute the Poisson (1) probabilities only for $0, 1, 2, \dots, n$, the possible numbers of matches.

[66]:
```
def matches_Poisson_tvd(n):
    k = np.arange(n + 1)
    return tvd(match_dist(n), stats.poisson.pmf(k,1))
```

To see if the value that your function is returning make sense, start by looking at the distribution of M_5, the number of matches if you just have 5 letters. Make sure you understand why there is a gap in the probability histogram.

[67]:
```
matches_5 = Table().values(np.arange(6)).probabilities(match_dist(5))
Plot(matches_5)
```



Which do you think should be larger: `matches_Poisson_tvd(5)` or `matches_Poisson_tvd(100)`? Why?

`matches_Poisson_tvd(5)` should be larger than `matches_Poisson_tvd(100)` because as n approaches infinity, `Poisson(1)` approximation improves. The binomial distribution is more suited when n is large and p is small, which slowly turns into the `Poisson(1)` distribution. So, when n is small, the TVD will show that the binomial and Poisson distributions differ more, leading to a bigger TVD. As n grows, the binomial slowly resembles the Poisson. Hence, the TVD will get smaller.

Run the cell below to see that your function behaves consistently with your answer above.

[68]:
```
matches_Poisson_tvd(5), matches_Poisson_tvd(100)
```

[68]:
```
(0.03411123088143108, 2.0236739346356693e-17)
```

2.4.3 3c) [CODE] Error in the Approximation

Let a and b be two integers with $0 \le a < b \le 100$. Suppose you approximate $P(a \le M_{100} \le b)$ by $\sum_{k=a}^{b} e^{-1}\frac{1}{k!}$. What is the largest possible error you could make?

Note: Your answer should be a number that works as an upper bound no matter which a and b are chosen according to the description above. If you are not sure how to proceed, re-read the introduction to the lab and think about the main point of **Section 1**.

```
[69]: matches_Poisson_tvd(100) * 2
```

```
[69]: 4.0473478692713387e-17
```

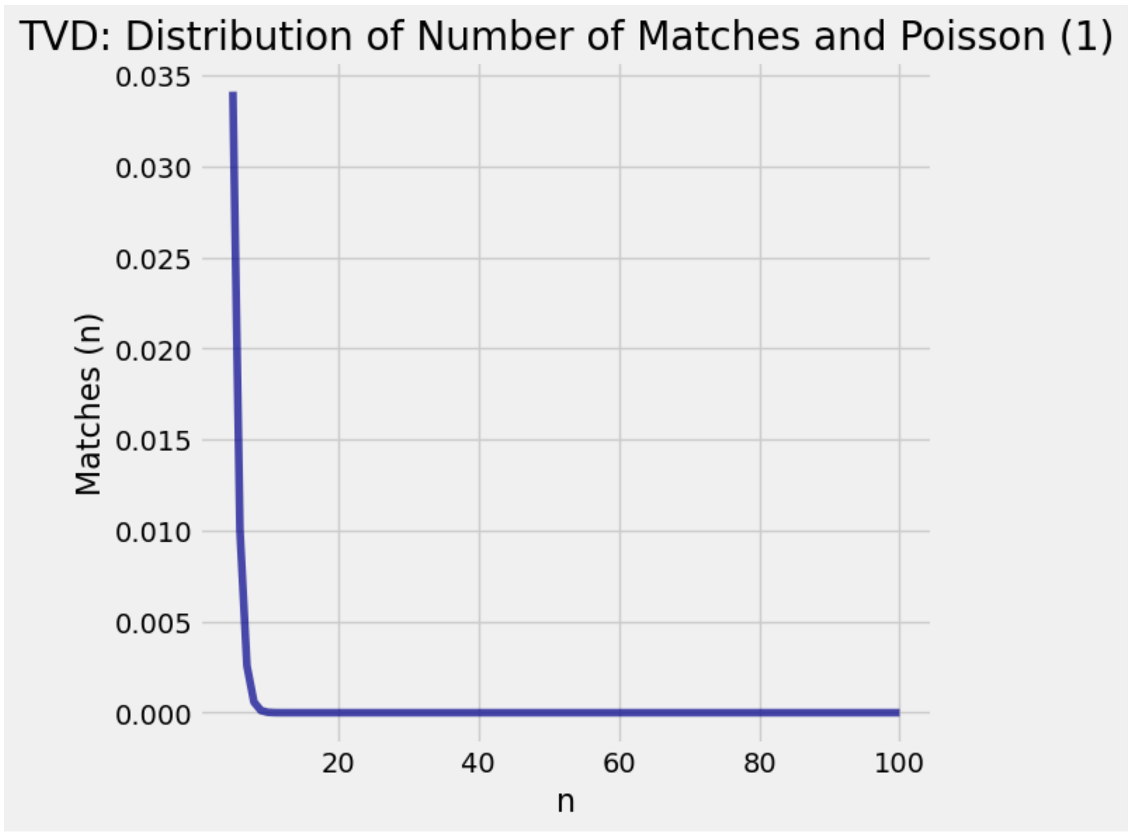
The more letters you have, the better the approximation will be. To visualize this, extend `tvd_table` below with a column labeled `Matches (n)` that contains the total variation distance between the exact distribution of the number of matches and its Poisson (1) approximation. In each row, the number of letters n is given by the entry in Column 0.

```
[70]: tvd_table = Table().with_column('n', np.arange(5, 101))

      matches_tvds = tvd_table.apply(matches_Poisson_tvd, 'n')

      tvd_table = tvd_table.with_column("Matches (n)", matches_tvds)

      tvd_table
```

```
[70]: n    | Matches (n)
      5    | 0.0341112
      6    | 0.0100777
      7    | 0.00258417
      8    | 0.000585866
      9    | 0.000119094
      10   | 2.19485e-05
      11   | 3.70026e-06
      12   | 5.74911e-07
      13   | 8.28377e-08
      14   | 1.11286e-08
      … (86 rows omitted)
```

Use the Table method `plot` to draw a line plot of the TVDs as a function of n.

```
[71]: tvd_table.plot('n')
      plt.title('TVD: Distribution of Number of Matches and Poisson (1)');
```



Note how sharply the graph falls. For $n \geq 10$ or so, there is almost no error at all in using the Poisson (1) distribution to approximate the distribution of M_n.

2.5 Section 4. Poisson (1) Approximation to Binomial Distributions

Roughly stated, a theorem we proved in class says that if n is large and p is small, then the binomial (n, p) probabilities are close to Poisson (np) probabilities.

Therefore if n is large then the binomial $(n, 1/n)$ probabilities should be close to Poisson (1) probabilities.

2.5.1 4a) [CODE] Binomial $(10, 1/10)$ Distribution

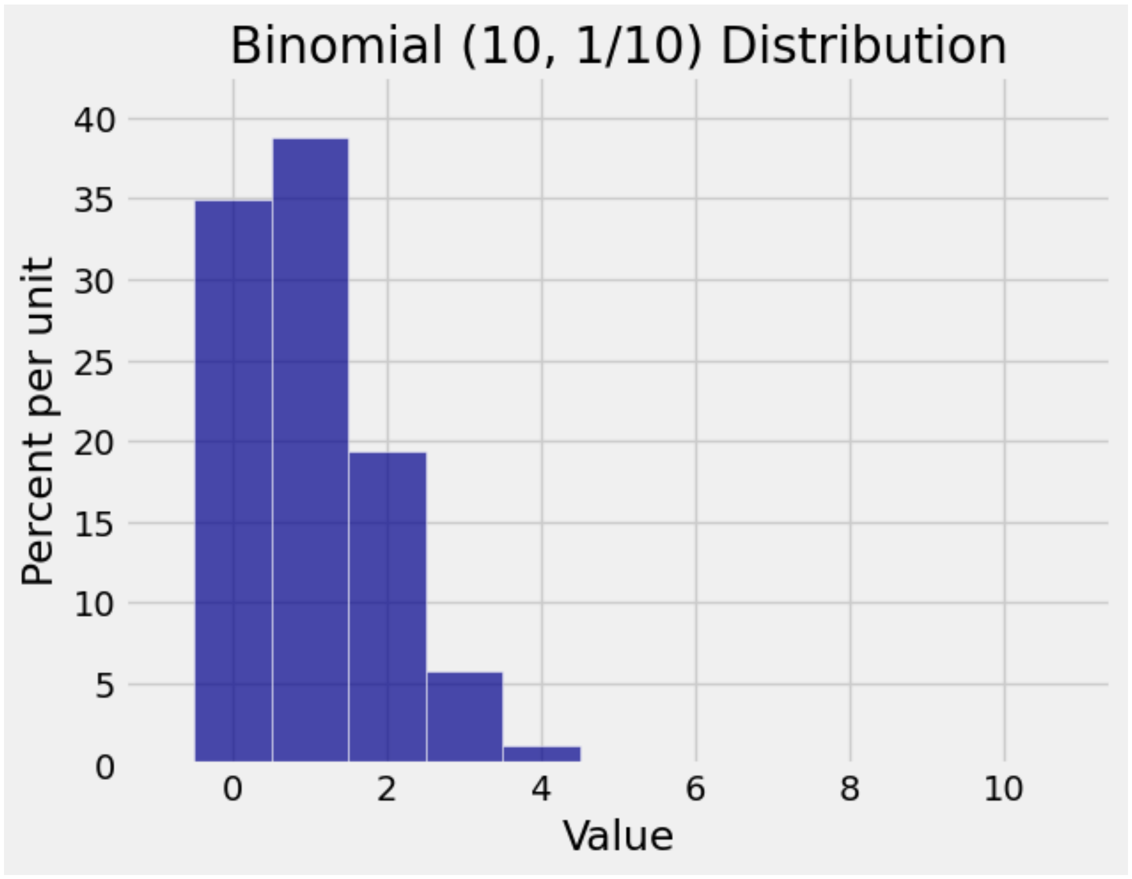
If `values` is an array of integers in the range 0 through n, then

`stats.binom.pmf(values, n, p)`

evaluates to an array of the binomial (n, p) probabilities of the entries in `values`.

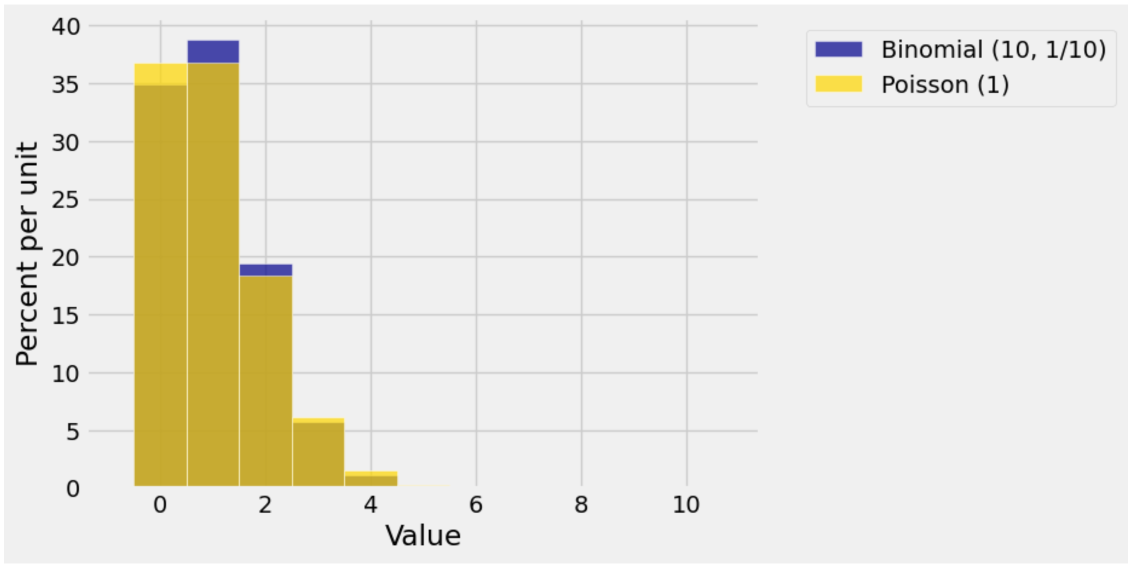
Display a histogram of the binomial $(10, 1/10)$ distribution. Notice how the probabilities are concentrated on the low values. This is a signal to start thinking about Poisson approximations, even though the number of trials $(n = 10)$ isn't very large.

[72]:
```
# binomial (n, 1/n) distribution

n = 10                              # number of trials

k = np.arange(n+1)                             # possible values
binom_probs = stats.binom.pmf(k,n,1/n)                   # binomial (n, 1/n)␣
 ↪probabilities



binom_dist = Table().values(k).probabilities(binom_probs)

Plot(binom_dist)
plt.title('Binomial (10, 1/10) Distribution');
```



Since $n = 10$ isn't very large, this distribution doesn't look exactly like the Poisson (1) distribution though the two have many similarities. Run the cell below to see the overlaid histograms.

[73]:
```
poisson_1_probs = stats.poisson.pmf(k, 1)
poisson_1_dist = Table().values(k).probabilities(poisson_1_probs)

Plots('Binomial (10, 1/10)', binom_dist, 'Poisson (1)', poisson_1_dist)
```



At this point it will come as no surprise that the Poisson (1) approximation gets better as n increases. You will quantify this in the remaining exercises below.

2.5.2 4b) [CODE] TVD between the Binomial $(n, 1/n)$ Distribution and its Poisson Approximation

Define a function `binomial_Poisson_tvd` that takes n as its argument and returns the total variation distance between the binomial $(n, 1/n)$ and Poisson (1) distributions. As before, it's fine to compute the Poisson (1) probabilities only on 0 through n, the possible values of the binomial.

[74]:
```
def binomial_Poisson_tvd(n):
    k = np.arange(n + 1)
    return tvd(stats.binom.pmf(k, n, 1/n), stats.poisson.pmf(k, 1))
```

As a check to see if your function is working correctly, run the cell below. The output should be about 1%.

[75]:
```
binomial_Poisson_tvd(30)
```

[75]:
```
0.009379738441887376
```

Fill in the blank with the name of a distribution and its numerical parameter or parameters.

The output of the code cell above is the biggest possible error in using the Poisson (1) distribution to approximate the ___________ distribution.

binomial

2.5.3 4c) [CODE] Error in Approximation

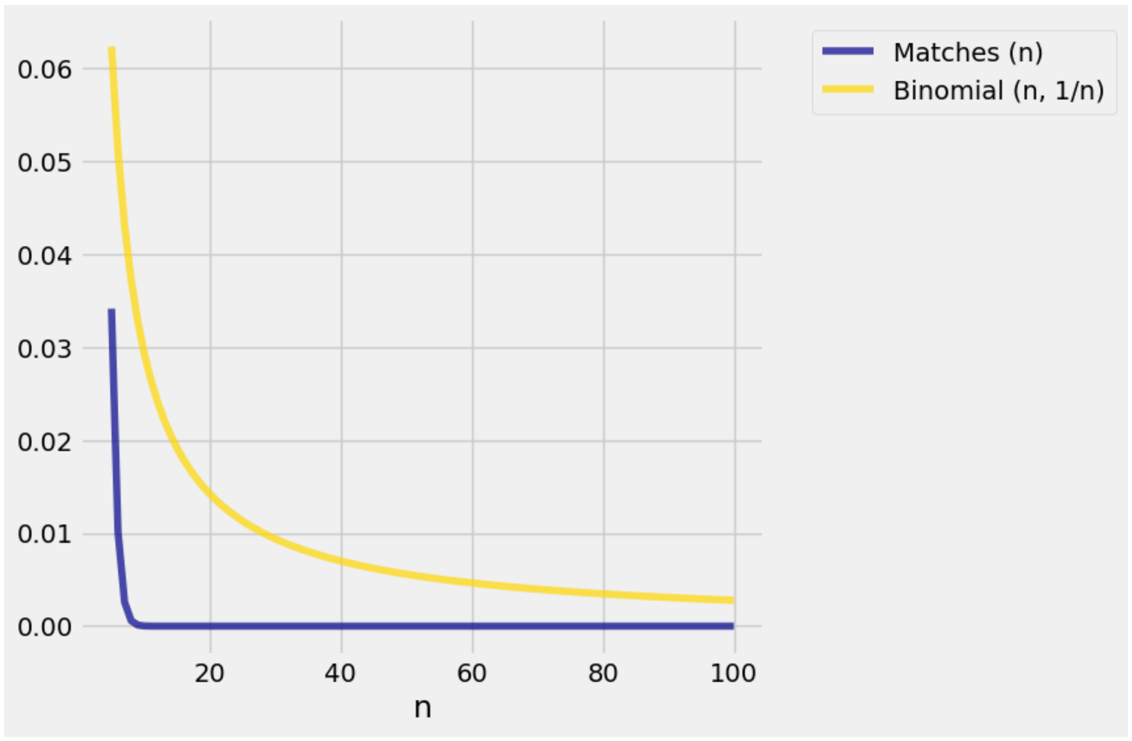
Extend `tvd_table` defined earlier with a column labeled `'Binomial (n, 1/n)'` that contains the TVD between the binomial $(n, 1/n)$ and Poisson (1) distributions, where in each row n is given by the entry in Column 0.

[76]:
```
binom_tvds = tvd_table.apply(binomial_Poisson_tvd, 'n')

tvd_table = tvd_table.with_column("Binomial (n, 1/n)", binom_tvds)

tvd_table
```

[76]:
```
n    | Matches (n) | Binomial (n, 1/n)
5    | 0.0341112   | 0.0622837
6    | 0.0100777   | 0.0509556
7    | 0.00258417  | 0.0430299
8    | 0.000585866 | 0.0372241
9    | 0.000119094 | 0.0327973
10   | 2.19485e-05 | 0.0293116
11   | 3.70026e-06 | 0.0264958
12   | 5.74911e-07 | 0.0241737
13   | 8.28377e-08 | 0.0222259
14   | 1.11286e-08 | 0.0205686
… (86 rows omitted)
```

Run the cell below to get overlaid line plots of the two TVD columns as functions of n.

[77]:
```
tvd_table.plot(0)
```



The graph of the binomial-Poisson TVDs drops sharply, though not as sharply as the matches-Poisson TVD graph.

Fill in the blanks (use the code cell below if you need to):

For values of n that are about ____________ or more, Poisson (1) approximations to binomial $(n, 1/n)$ probabilities will be off by at most 0.5%.

56

[78]:
```
value = []
counter = 0
for x in tvd_table.column('Binomial (n, 1/n)'):
    if x <= 0.005:
        value += (tvd_table.column('n')[counter], i)
        break
    counter += 1

value
```

[78]:
```
[56, 0.062283745834045667]
```

Now you can use total variation distance to help answer the question, "How large does n have to be before I can use the Poisson (1) approximation to the binomial $(n, 1/n)$ distribution?"

- First decide how much error you are prepared to tolerate in your approximations.

- Then use `tvd_table` (or an extended one with larger values of n) to find the smallest n for which the TVD is below your threshold error.
- For that n or larger, the error in your Poisson (1) probability approximations will be below your threshold.

2.6 Conclusion

What you have learned in this lab: - Let X be a discrete random variable. If you use an approximation to the distribution of X, then the total variation distance between the exact and approximate distributions measures the worst error you can make in approximating probabilities of events determined by X. - Many random variables in this lab have a large number of possible values, and the approximating Poisson distribution has infinitely many possible values. But no matter how large n is, the *probable* values of all the variables are in a very small range — 0 through about 8 or 10 — because all of the distributions are roughly Poisson (1). - The total variation distance between the distribution of the number of matches in a random permutation of n elements and its Poisson (1) approximation falls very sharply. By about 10 elements or so, you might as well use the Poisson distribution for the number of matches. - The total variation distance between the binomial $(n, 1/n)$ and Poisson (1) distributions falls sharply as a function of n and is below 1% even for moderate values of n.

3 Lab 2 ends here. It is due by 5PM on Monday, February 3rd.

3.0.1 Section 5, below the Submission Instructions, is optional and should not be turned in.

3.1 Submission Instructions

Many assignments throughout the course will have a written portion and a code portion. Please follow the directions below to properly submit both portions.

3.1.1 Written Portion

- Scan all the pages into a PDF. You can use any scanner or a phone using applications such as CamScanner. Please **DO NOT** simply take pictures using your phone.
- Please start a new page for each question. If you have already written multiple questions on the same page, you can crop the image in CamScanner or fold your page over (the old-fashioned way). This helps expedite grading.
- It is your responsibility to check that all the work on all the scanned pages is legible.
- If you used LaTeX to do the written portions, you do not need to do any scanning; you can just download the whole notebook as a PDF via LaTeX.

3.1.2 Code Portion

- Save your notebook using `File > Save Notebook`.
- Generate a PDF file using `File > Save and Export Notebook As > PDF`. This might take a few seconds and will automatically download a PDF version of this notebook.
  - If you have issues, please post a follow-up on the general Lab 2 Ed thread.

3.1.3 Submitting

- Combine the PDFs from the written and code portions into one PDF. Here is a useful tool for doing so.
- Submit the assignment to Lab 2 on Gradescope.
- **Make sure to assign each page of your pdf to the correct question.**
- **It is your responsibility to verify that all of your work shows up in your final PDF submission.**

If you are having difficulties scanning, uploading, or submitting your work, please read the Ed Thread on this topic and post a follow-up on the general Lab 2 Ed thread.

3.2 We will not grade assignments which do not have pages selected for each question.

3.3 [Optional Proof: Do Not Turn In] Section 5: Biggest Possible Error

These exercises guide you through a proof that the TVD can be interpreted as a maximum difference of probabilities.

We are going to compare two probability distributions P_{blue} and P_{gold} on a finite set of values S. Suppose the values are labeled $1, 2, \ldots, n$.

The *total variation distance between P_{blue} and P_{gold}* is defined as

$$\|P_{blue} - P_{gold}\|_{TVD} \ = \ \max\{|P_{blue}(A) - P_{gold}(A)| : A \subseteq S\}$$

The definition says: For every event A, compute how far off $P_{blue}(A)$ is from $P_{gold}(A)$. The TVD is the biggest value among all these differences.

That doesn't look at all like what we have been calculating as the TVD starting way back in Data 8. But in fact it's the same thing. It's your job to show how.

Before you get started, confirm your understanding of the definition. Suppose you calculate the TVD between two distributions and get 0.003. That says that if you list all possible events and compare their probabilities under the two distributions, the biggest difference you will get is 3/1000. The two distributions are pretty close.

The goal of this section is to show that this new definition of TVD is equivalent to the calculation we have been doing all along. Let's start by setting up some notation. For each i in S, let $P_{blue}(i) = b_i$ and let $P_{gold}(i) = g_i$. If you imagine a bar graph or histogram of each distribution, then b_i is the area of the blue bar at the value i, and g_i is the area of the gold bar at i.

In this notation, our familiar calculation of the TVD is

$$\frac{1}{2}\sum_{i \in S}|b_i - g_i|$$

In this question and the next you will show that

$$\max\{|P_{blue}(A) - P_{gold}(A)| : A \subseteq S\} \ = \ \frac{1}{2}\sum_{i \in S}|b_i - g_i|$$

Three events will be important in the calculations.

The set of values for which the blue bars exceed the gold:

$$B = \{i : b_i > g_i\}$$

The set of values for which the gold bars exceed the blue:

$$G = \{i : g_i > b_i\}$$

The set of values for which the blue bars and gold bars are equal:

$$E = \{i : b_i = g_i\}$$

Keep in mind that for any event A,

$$P_{blue}(A) = \sum_{i \in A} b_i \qquad \text{and} \qquad P_{gold}(A) = \sum_{i \in A} g_i$$

3.3.1 5a) Gold and Blue Bars

Find the value of

$$\sum_{i \in B} b_i ~+~ \sum_{i \in G} b_i ~+~ \sum_{i \in E} b_i$$

Repeat the calculation after replacing b_i by g_i in all three sums above.

Hence show that

$$\sum_{i \in B} (b_i - g_i) ~=~ \sum_{i \in G} (g_i - b_i)$$

This proves a statement we have made in Data 8 and in Part **1a** of this lab: "The amount by which the blue bars exceed the gold is the same as the amount by which the gold bars exceed the blue."

3.3.2 5b) Another Expression for the TVD

Our usual calculation of TVD is

$$\frac{1}{2} \sum_{i \in S} |b_i - g_i|$$

Partition the sum into two pieces to show that

$$\frac{1}{2} \sum_{i \in S} |b_i - g_i| ~=~ \sum_{i \in B} (b_i - g_i) ~=~ \sum_{i \in G} (g_i - b_i)$$

This proves another statement we made in Data 8 and Part **1a** of this lab: "The TVD is the amount by which the blue bars exceed the gold."

3.3.3 5c) An Inequality

Now let A be any event. Show that

$$P_{blue}(A) - P_{gold}(A) ~=~ \sum_{i \in AB} (b_i - g_i) ~-~ \sum_{i \in AG} (g_i - b_i)$$

Hence show that

$$P_{blue}(A) - P_{gold}(A) ~\le~ \sum_{i \in AB} (b_i - g_i) ~~~~~ \text{and} ~~~~~ P_{gold}(A) - P_{blue}(A) ~\le~ \sum_{i \in AG} (g_i - b_i)$$

3.3.4 5d) Absolute Inequalities

Use the first of the two inequalities in **c** to show that if $P_{blue}(A) - P_{gold}(A) > 0$ then

$$|P_{blue}(A) - P_{gold}(A)| ~\le~ \sum_{i \in B} (b_i - g_i)$$

Use the second of the two inequalities in **c** to show that if $P_{blue}(A) - P_{gold}(A) < 0$ then

$$|P_{blue}(A) - P_{gold}(A)| ~\le~ \sum_{i \in G} (g_i - b_i)$$

3.3.5 5e) Equivalence of the Definitions

Identify an event for which one of the inequalities in **d** is an equality.

Explain why you now have a complete proof of

$$\max\{|P_{blue}(A) - P_{gold}(A)| : A \subseteq S\} ~=~ \frac{1}{2}\sum_{i \in S} |b_i - g_i|$$

That is, our usual calculation of the TVD is equivalent to finding the biggest difference between probabilities assigned by the two distributions to any event.

```
[ ]:
```