DATA 8, FALL 2016 # MIDTERM EXAM **A. Adhikari**

NAME (FIRST, LAST): ______________________ **SID:** ____________

TIME AND CONDITIONS: 45 minutes; closed book/notes/internet; no calculator/computer

QUESTIONS AND ANSWERS

• There are 8 questions, some of which have two parts.

• Not all questions will take the same amount of time. Some will go very fast.

• As in homework, some questions involve a short paragraph that you have to read. Reading quickly and accurately is highly recommended. It might help to mark key phrases as you read.

• You may answer any part of any question. If the answer to one part depends on another that you couldn't do, you can still provide an answer such as "The answer to part (a), divided by 2."

• When answers involve calculations that can't easily be done by mental arithmetic, we have asked you to leave the arithmetic unsimplified. Just leave them in any form that can be typed (perhaps laboriously) into a calculator to get the decimal answer.

• If a question asks for one line of code, you can take the space you need to write out the line by hand. You don't have to squeeze your handwritten code into one line of space on the exam page. But it should execute correctly if we type it into one line in a notebook cell.

• Explanations are expected to be concise. One or two clear sentences should be enough. Calculations and code are sufficient as explanations.

GRADING

• The exam is worth 50 points.

• You get 1 point for turning in a test with your name and SID on it, and 1 more point if you write each of your answers within the space provided. See **FORMAT** below.

• Each of the 8 questions is worth 6 points. If a question has two parts, each part is worth 3 points.

• We will give partial credit, but only for substantial progress towards a correct answer. We get to decide what "substantial progress" means.

• Commit yourself to a single answer for each part of each question. If you give multiple answers (such as both True and False), please don't expect credit, even if the right answer is among those that you gave.

FORMAT

• There is space for your answer below each question. **Please do not write outside the black boundary**; the scanner and Gradescope won't read it.

• If you need scratch paper, please use the backs of the pages of the exam, but be aware that they will not be graded.

• A reference sheet of class content will be provided. But it does not contain everything that was covered in class.

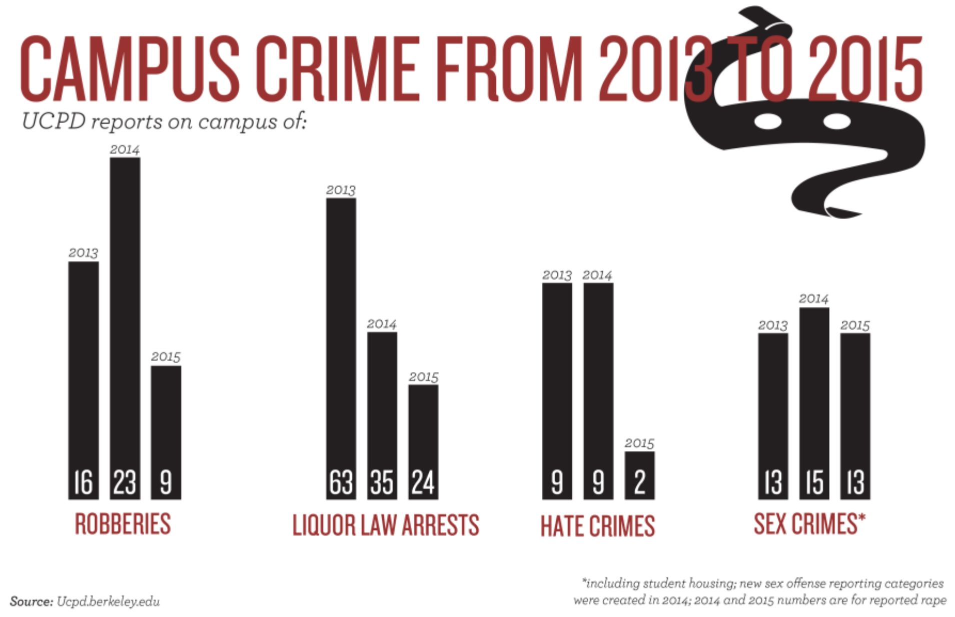
HONOR CODE

Data Science and the entire academic enterprise are based on one quality – integrity. We are all part of a community that doesn't fabricate evidence, doesn't fudge data, doesn't steal other people's work, doesn't lie and cheat. You trust that we will treat you fairly and with respect. We trust that you will treat us and your fellow students fairly and with respect. **Please abide by UC Berkeley's Honor Code:**

"As a member of the UC Berkeley community, I act with honesty, integrity, and respect for others."

Name: ____

1. The graphic below appeared on the front page of the Daily Cal on Tuesday of last week.



(a) One important choice made by the graphic designers makes it difficult to visually compare the numbers of crimes in various categories. What was that choice, and why does it make visual comparisons difficult?

(b) The data have been entered into a table called **crime**, shown below.

Year	Robberies	Liquor Law Arrests	Hate Crimes	Sex Crimes
2013	16	63	9	13
2014	23	35	9	15
2015	9	24	2	13

Use the table **crime** in one line of Python code that produces a graphic that corrects the problem in (a), and **explain** why the problem will be fixed.

Name: ______________________________

2. The table **colors** is created as follows:

colors = Table().with_column(
    'Color', make_array('blue', 'green', 'red', 'purple')
)

Find a numerical expression for the chance that all the entries of **colors.sample(3).column(0)** are the same. Don't simplify the arithmetic.

Name: ____

3. Fill in each blank with the best choice from the table of items below. **No explanations are necessary.**

You can use items more than once. Don't write out the words or phrases you picked; just enter the item number in the answer table.

1.	approximate	2.	empirical
3.	sample	4.	population
5.	random sample	6.	estimate
7.	parameter	8.	statistic
9.	low	10.	high
11.	bias	12.	confounding
13.	likely	14.	unlikely
15.	variability	16.	distribution
17.	approximation	18.	probability
19.	convenience	20.	control

• Important criteria for a statistic to be a good estimate of a population ____I____ include ____II____ bias and low ____III____.

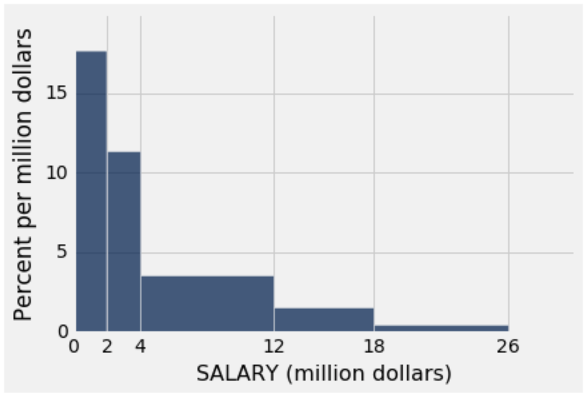
• The ____IV____ distribution of a large ____V____ is likely to resemble the ____VI____ of the population.

Answer Table

Blank	**Item Number**
I	
II	
III	
IV	
V	
VI	

Name: ____

4. The table **nba** has a column labeled **SALARY** containing the 2015-2016 salaries of NBA players. Here is the output of **nba.select('SALARY').hist(bins = make_array(0, 2, 4, 12, 18, 26))** along with the heights of the bars.



bin (million dollars)	[0, 2)	[2, 4)	[4, 12)	[12, 18)	[18, 26)
height (percent per million dollars)	17.63	11.39	3.60	1.60	0.45

The interval $[a, b)$ contains all values that are greater than or equal to a and less than b.

(a) Which bin contains more players: [2, 4) or [4, 12)? **Explain** your choice.

b) To see some more detail in the [4, 12) range, the histogram will be redrawn with bins as shown below. The display includes the heights that are available from above.

bin (million dollars)	[0, 2)	[2, 4)	[4, 6)	[6, 12)	[12, 18)	[18, 26)
height (percent per million dollars)	17.63	11.39	(i)	(ii)	1.60	0.45

The expression **nba.num_rows** evaluates to 417.
The expression **nba.where('SALARY', are.between(4, 6)).num_rows** evaluates to 56.

If possible, provide a numerical expression for each missing height (do not simplify the arithmetic). If this is not possible, explain why not.

(i)

(ii)

Name: ____

5. In a Public Health Service study of the effect of smoking, participants were separated into groups according to gender and age. In each subgroup, the observation was the same – those who had never smoked were on average somewhat more healthy than the current smokers, but the current smokers were on average quite a bit more healthy than those who had recently stopped smoking.

Based on these results, should the current smokers have been advised not to give up smoking? If not, provide at least one factor that helps explain the observation.

Name: ____

6. A clinic is trying to run a randomized controlled trial of a new appointment system. The clinic is open Monday through Saturday each week. As a way of randomization, all appointment requests that come in on Mondays, Wednesdays, and Fridays are assigned to the new system. All appointment requests that come in on Tuesdays, Thursdays, and Saturdays are assigned to the old system.

At the end of a few weeks, 212 appointment requests have been assigned to the new system and 277 to the old one.

Clinician A says, "This method is biased against the people who call on on Mondays, Wednesdays, and Fridays. If we had tossed a fair coin for each request, we wouldn't have ended up with so few requests assigned to the new system."

Clinician B says, "No, the results are just like tossing a fair coin."

(a) Provide null and alternative hypotheses that reflect the views of the two clinicians.

Null hypothesis:

Alternative hypothesis:

(b) A test of the hypotheses in part (a) will be performed. The test statistic is the **number of requests assigned to the new system**.

Fill in the blanks in the code below so that the output of the final line is the empirical P-value of the test. Make sure your code is consistent with the hypotheses in part (a).

Two arrays have been created for you. You are welcome to use their names in your code if you wish.

- The array **array2** consists of two entries, one of which is the string **'New'** and the other is **'Old'**.
- The array **array489** consists of 489 entries, 212 of which are **'New'** and 277 are **'Old'**.

Here is the code for you to complete.

```
stats = make_array()

for i in np.arange(5000):

    draws = np.random.choice(__________________________, __________________________)

    number_new = np.count_nonzero(______________________________________________________)

    stats = np.append(_______________________________, number_new)

empirical_P = np.______________________________________________________________________

empirical_P
```

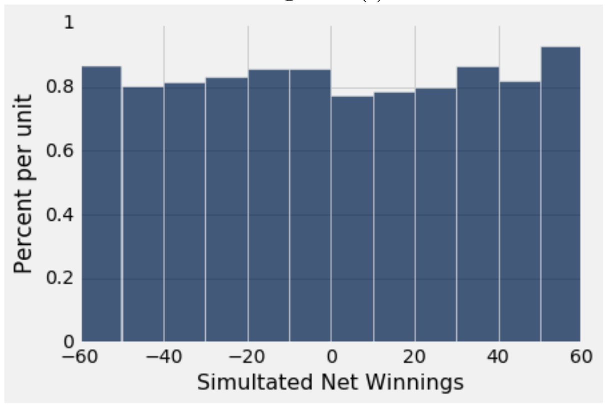
Name: ____

7. My friend and I play the following gambling game repeatedly:

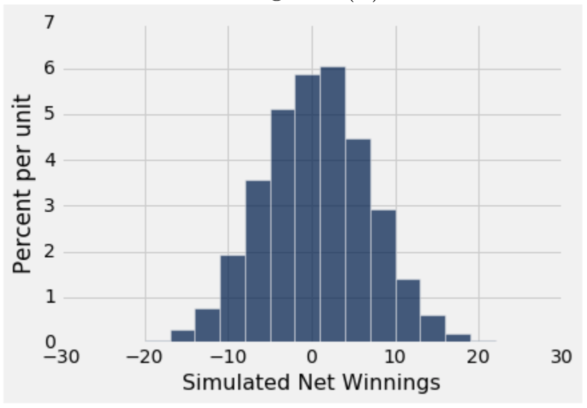
A fair die is rolled. If it shows one or six spots, I pay my friend \$1. If it shows two or five spots, my friend pays me \$1. If it shows three or four spots, neither of us pays any money.

One of the histograms below is an empirical histogram of my overall net winnings in 60 games, based on 10,000 repetitions of finding the overall net gain in 60 games. **Which histogram is it, and why?**

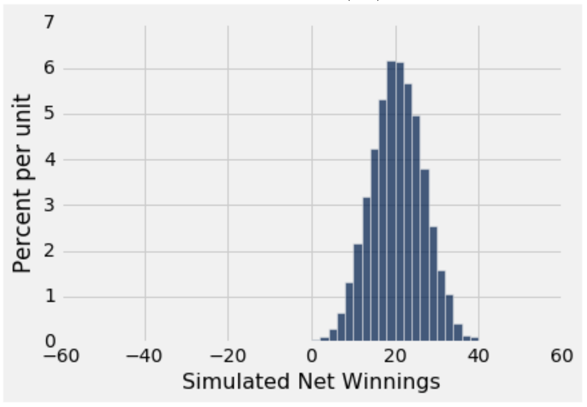
Histogram (i)



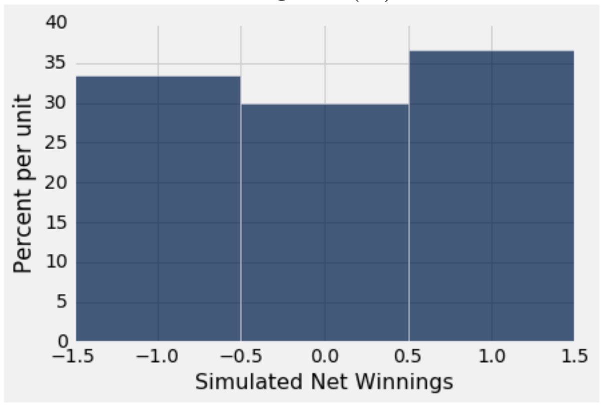
Histogram (ii)



Histogram (iii)



Histogram (iv)



Justification:

Name: ____

8. The table **voters** contains one row for each voter in a large national sample. There are two columns:

Column 0: **State** is the state where the voter is registered. The states are denoted by their two-letter abbreviations, so that California is the string **'CA'** and Florida is **'FL'**.

Column 1: **Choice** has three categories: **'Candidate A'**, **'Candidate B'**, and **'Undecided'**.

Write code to calculate the total variation distance between the **Choice** distributions of California and Florida. You can assume that the table contains at least one voter in each **Choice** category for both the states.

Use the following steps.

# A table of the data for all the California voters, containing just one column: Choice

ca_voters = voters.____

# An array containing the Choice distribution (in proportions) for California

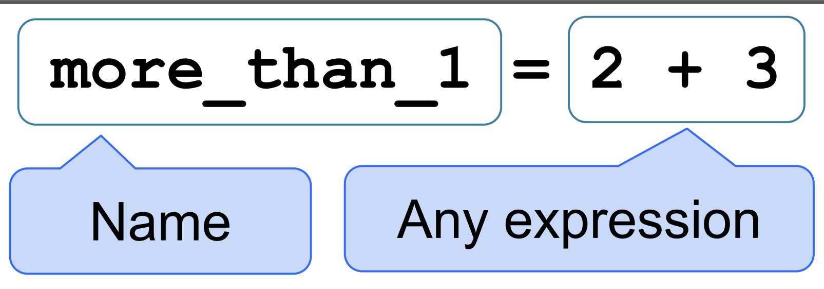
ca_dist = ca_voters.____

Now assume you have created **fl_voters** and **fl_dist** correspondingly for Florida. There is no need to write out the code for those.

Find the total variation distance:

tvd = ____

Data 8 Midterm Reference Sheet — Page 1



```
more_than_1 = 2 + 3
```

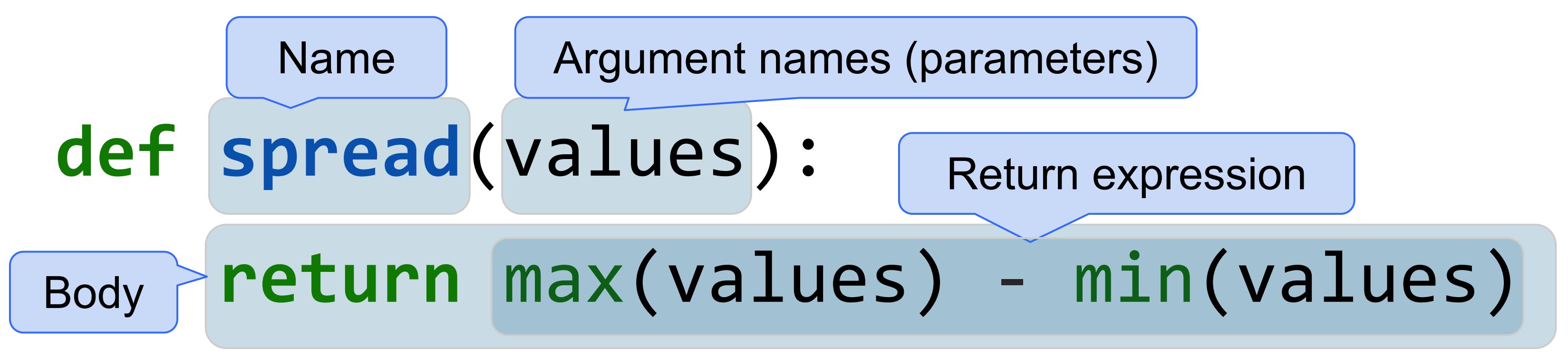
- Statements don't have a value; they perform an action
- An assignment statement changes the meaning of the name to the left of the = symbol
- The name is bound to a value (not an equation)

- < and > mean what you expect (less than, greater than)
- <= means "less than or equal"; likewise for >=
- == means "equal"; != means "not equal"
- Comparing strings compares their alphabetical order

Arrays - sequences that can be manipulated easily.

- All elements of an array should have the same type
- Arithmetic is applied to each element of an array individually
- Elementwise operations can be done on arrays of the same size

---



```
def spread(values):
    return max(values) - min(values)
```

```
for  i  in  np.arange(12)  :
    print(i)
```

The body is executed **for** every item in a sequence
The body of the statement can have multiple lines
The body should do something: print, assign, hist, etc.

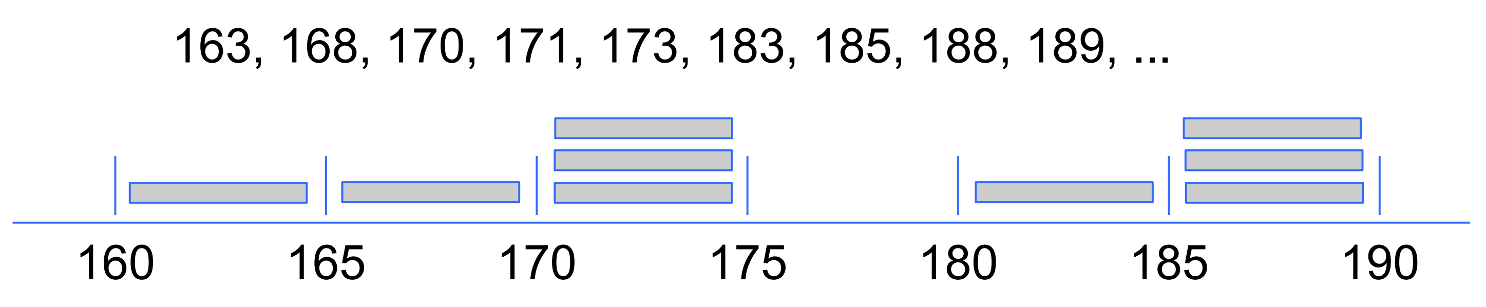
```
Conditional Statements
    if <if expression>:
        <if body>
    elif <elif expression 0>:
        <elif body 0>
    elif <elif expression 1>:
        <elif body 1>
    ...
    else:
        <else body>
```

Values in Tables: Every column of a table is an array.

- **Categorical**
  - May or may not have an ordering
  - Categories are the same or different
  - Allows grouping by value (`group, groups, pivot, join`)
- **Numerical**
  - Ordered
  - Allows binning by value (`bin, hist`)

Binning is counting the number of numerical values that lie within ranges, called bins.

- Bins are defined by their lower bounds (inclusive)
- The upper bound (exclusive) is the lower bound of the next bin



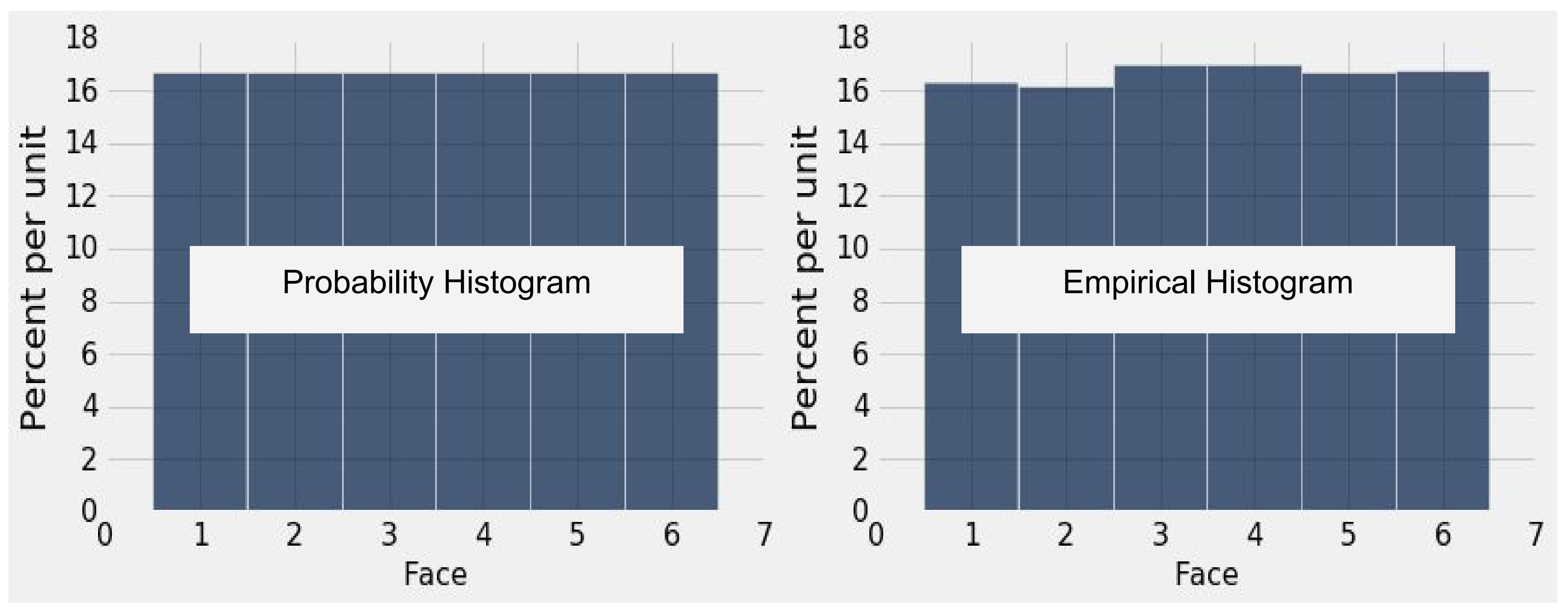
A **histogram** has two defining properties:

- The bins are contiguous (though some might be empty) and are drawn to scale
- The **area** of each bar is equal to the proportion of entries in the bin

Has total area 1 (or 100%)
Vertical axis units: Proportion / Unit on the horizontal axis

---

- A histogram of proportions of all possible outcomes of a *known* random process is called a *probability histogram*
- A histogram is a summary visualization of a *distribution*
- A histogram of proportions of actual outcomes generated by sampling or actual data is called an *empirical histogram*



Calculating Probabilities

Complement Rule: P(event does not happen) = 1 - P(event happens)
Multiplication Rule: P(two events happen) = P(one happens) * P(other happens, given the first happened)
Addition Rule: P(an event happens) = P(first way it can happen) + P(second way it can happen) IF it can happen in ONLY one of two ways

---

Total Variation Distance - a statistic measuring the difference between categorical distributions. The sum of the absolute value of the differences between proportions in each category, divided by two.

Data 8 Midterm Reference Sheet — Page 2

In the examples in the left column, `np` refers to the NumPy module, as usual. Everything else is a function, a method, an example of an argument to a function or method, or an example of an object we might call the method on. For example, `tbl` refers to a table, `array` refers to an array, and `num` refers to a number. `array.item(0)` is an example call for the method `item`, and in that example, `array` is the name previously given to some array.

Example function call	**Value of a call to the function**
`max(array)`	Maximum or minimum of a sequence
`sum(array)`	Sum of all elements in an array
`len(array)`	Length (num elements) in an array
`round(num); np.round(array)`	Round number or array of numbers to the nearest integer
`abs(num); np.abs(array)`	Take the absolute value of number or each number in an array
`np.average(array)`	The average of the values in an array

`np.arange(start, stop, step)`
`np.arange(start, stop)`
`np.arange(stop)`
	An array of numbers starting with `start`, going up in increments of `step`, and going up to but excluding `stop`. When `start` and/or `step` are left out, default values are used in their place. Default `step` is 1; default `start` is 0.

`np.count_nonzero(array)`	Count the number of non-zero elements in an array (`False` counts as zero, `True` as non-zero)
`array.item(index)`	The item in the array at some index. `array.item(0)` is the first item of `array`.
`np.append(array, item)`	A copy of the array with item appended to the end.

`np.random.choice(array, num)`
`np.random.choice(array)`
	An array of things randomly selected with replacement from an array. `num` is the number of things selected. Default `num` is 1.

`Table()`	An empty table.
`Table.read_table(filename)`	A table with data from a file.
`tbl.num_rows`	The number of rows in a table.
`tbl.num_column`	The number of columns in a table.
`tbl.labels`	A list of the column labels of a table.

`tbl.with_column(name, values)`
`tbl.with_columns(n1, v1, n2, v2...)`
	A table with an additional or replaced column or columns. `name` is a string for the name of a column, `values` is an array.

`tbl.column(column_name)`	The values of a column (an array).
`tbl.select(col1, col2, ...)`	A table with only the selected columns. (Each argument is the name of a column.)
`tbl.drop(col1, col2, ...)`	A table without the selected columns. (Each argument is the name of a column.)
`tbl.relabeled(old_label, new_label)`	A new table with a label changed.
`tbl.relabel(old_label, new_label)`	Change the label of a column in place. (Has no value!)
`tbl.take(row_indices)`	A table with only the rows at the given indices. `row_indices` is an array of indices.
`tbl.sort(column, descending)`	A table of rows sorted according to the values in a column. Default order is ascending.
`tbl.where(column, predicate)`	A table of the rows for which the column satisfies some predicate. See "`Table.where predicates`" below.
`tbl.apply(function, column)`	Returns an array where a function is applied to each item in a column.

`tbl.group(column, func)`
`tbl.groups(column_names_array, func)`
	Group rows by unique values in a column. Other values aggregated by count (default) or optional arg `func.`
	Group rows by unique combinations of values in some columns. Aggregate/count other values as above.

`tblA.join(colA, tblB, colB)`
`tblA.join(colA, tblB)`
	Generate a table with the columns of self and other, containing rows for all values of a column that appear in both tables. Default `colB` is `colA.` `colA` is a string specifying a column name, as is `colB.`

`tbl.pivot(row, col, values, collect)`
`tbl.pivot(row, col)`
	Group rows by unique values in two columns; count or aggregate values from a third column, collect with some function. Default `values` and `collect` return counts in cells.

`tbl.sample(n, with_replacement)`	Returns a new table where k rows are randomly sampled from the original table. Default is with replacement.
`tbl.scatter(x_column, y_column)`	Draws a scatter plot consisting of one point for each row of the table.

`tbl.barh(categories)`
`tbl.barh(categories, frequencies)`
	Displays a bar chart with bars for each category in a column, with height proportional to the corresponding frequency. `frequencies` argument unnecessary if table consists just of a column of categories and a column of frequencies.

`tbl.hist(column, units, bins)`	Generates a histogram of the numerical values in a column. `units` and `bins` are optional arguments, used to label the axes and group the values into intervals (bins), respectively.

Operations: addition 2+3=5; subtraction 4-2=2; division 9/2=4.5 multiplication 2*3=6; division remainder 11%3=2; exponent 2**3=8

Data Types: string 'hello'; **boolean** True, False; **int** 1, -5; **float** - 2.3, -52.52, 7.9

Arithmetic with arrays is elementwise:

```
make_array(1,2,3) ** 2 # [1, 4, 9]
```

Table.where predicates (x is a string or number)

```
are.equal_to(x) # [2, 3, 4]
are.above(x) # val > x
are.below(x) # val < x
are.between(x, y) # x <= val < y
```