Data 8 Final Reference Sheet — Fall 2016

Python Basics

- **Functions:** called with parentheses (i.e., `np.mean(arr)`), defined with `def` statement

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

- **Comparators:** `==, !=, >, <, >=, <=` compare two values and return `True` or `False`.
- **Conditionals:** A structure to execute different lines of code based on whether certain conditions are true

```
if <if expression>:
    <if body>
elif <elif expression 0>:
    <elif body 0>
else:
    <else body>
```

- **Loops:** a `for` loop iterates through the elements of a sequence

```
two_three_four = make_array()
for x in make_array(1, 2, 3):
    two_three_four = np.append(two_three_four, x + 1)
```

Distance Between Two Points

- Two attributes x and y: $\sqrt{(x_0 - x_1)^2 + (y_0 - y_1)^2}$
- Three attributes x, y, and z: $\sqrt{(x_0 - x_1)^2 + (y_0 - y_1)^2 + (z_0 - z_1)^2}$
- and so on...

Probability

Probabilities are between 0 and 1.

- If all outcomes are equally likely, then $P(\text{event happens})$ = proportion of outcomes that make the event happen
- P(event happens) = 1 - P(the event doesnt happen)
- Chance that two events A and B both happen = P(A happens) $\times$ P(B happens given that A has happened)
- If event A can happen in exactly one of two ways, then P(A) = P(first way) + P(second way)
- **Bayes' rule:** $P(\text{first event happens given the second happens}) = \frac{P(\text{first event happens}) \times P(\text{second event happens given the first happens})}{P(\text{second event happens})}$

Descriptive Statistics

- **Median:** 50th percentile, where
  p-th percentile = smallest value on list that is at least as large as p% of the values
- **Mean** of 5, 7, 8, 8 $= (5 + 7 + 8 + 8)/4 = 5 \times 0.25 + 7 \times 0.25 + 8 \times 0.5$
- Mean depends on all the values; smoothing operation; center of gravity of histogram; if histogram is skewed, mean is pulled away from median towards the tail
- The mean of a 0/1 population is the proportion of 1s in the population
- **Standard deviation (SD):** The root mean square of deviations from average.
- The SD of a 0/1 population is less than or equal to 0.5
- **Chebychev's Bound:** No matter what the distribution looks like, the proportion in the range average $\pm z$ SDs is at least $1 - \frac{1}{z^2}$.
- If the distribution is normal, about 68% of values are within the range [average $\pm$ 1 SD] and about 95% of values are within the range [average $\pm$ 2 SDs].
- **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.
- **Standard units (s.u.):** To convert a value to standard units: $z = \frac{\text{value}-\text{average}}{\text{SD}}$.
  To convert standard units to original units: value $= z \times$ SD $+$ average.
- **Correlation (r):** $r = \text{mean}(x \text{ in s.u. } \times y \text{ in s.u. })$
- **Estimate of** y **=** $r \times x$, when both variables are measured in standard units
- **Slope of the regression line** $= r \times \frac{\text{SD of } y}{\text{SD of } x}$
- **Intercept of the regression line** = mean of $y -$ slope $\times$ mean of x
- **Residual** $=$ observed $y -$ regression estimate of y
- **Average of residuals** $= 0$
- **SD of residuals** $= \sqrt{1 - r^2} \times$ SD of y

With-Replacement Random Sample Means

The mean of a random sample with replacement is expected to be the population mean.

- **SD of Sample Mean** $= \frac{\text{Population SD}}{\sqrt{\text{Sample Size}}}$
- **Square Root Law:** If you multiply sample size by a factor, the accuracy of the sample mean goes up by the square root of the factor.
- **Central Limit Theorem:** If a sample is large, and drawn at random with replacement, then, regardless of the distribution of the population, the probability distribution of the sample sum (or of the sample mean) is roughly bell-shaped.

Code examples on the other side of this sheet.

Tables

Tables are a data structure used to store tabular (row and column) data. You may assume that these functions exist in Python. `tbl` refers to a table.

Function/Method	Description
`Table()`	Creates an empty table
`Table.read_table(filename)`	Returns a table read in from a CSV file
`tbl.labels`	Returns an array of a table's column labels
`tbl.num_rows, tbl.num_cols`	Returns the number of rows (and columns, respectively)
`tbl.column(name)`	Returns the values of a column (an array)
`tbl.with_column(name, values)`	Adds or replaces a column to a table
`tbl.with_columns(n1,v1,n2,v2)`	Adds or replaces multiple columns
`tbl.row(i)`	Returns the i-th row of a table
`tbl.append(row)`	Appends a row to a table
`tbl.select(col1,col2,...)`	A table with only the selected columns
`tbl.drop(col1,col2,...)`	A table without the specified set of columns
`tbl.take(row_indices)`	A table with only the rows at the given indices. `row_indices` is an array of indices.
`tbl.relabeled(old_lbl, new_lbl)`	Returns a copy of the table with a column label changed.
`tbl.apply(function, column)`	Returns an array where a function is applied to each item in a column
`tbl.sort(column_name)` `tbl.sort(column_name, descending)`	A table of rows sorted according to the values in a column (specified by name/index). Default order is ascending. For descending order, use argument `descending=True`.
`tbl.where(column, predicate)`	Selects rows from a table based on column values. See "`Table.where` predicates" below.
`tblA.join(colA, tblB, colB)`	Generate a table with the columns of self and other, containing rows for all values of a column that appear in both tables. `colA` is a string specifying a column name, as is `colB`. Takes the first match found in `tblB`
`tbl.group(column, func)`	Group rows by unique values in a column. Other values aggregated by count (default) or optional argument `func`.
`tbl.groups(col_names_array, func)`	Group rows by unique combinations of values in some columns. Aggregate/count other values as above.
`tbl.pivot(col1, col2)` `tbl.pivot(col1,col2,vals, collect)`	Return a pivot table where each unique value in col1 has its own column and each unique value in col2 has its own row. Count or aggregate values from a third column, collect with some function. Default vals and collect return counts in cells (`vals` and `collect` are optional arguments).
`tbl.sample()` `tbl.sample(n)` `tbl.sample(n, with_replacement)`	Returns a new table with `n` rows sampled from the original table. Default `n` is `tbl.num_rows`. Default sampling is with replacement, otherwise pass in `with_replacement = False`.
`tbl.barh(category_col)` `tbl.barh(category_col,freq_col)`	Displays a bar chart with bars for each category in the column whose name is passed in, with height proportional to the corresponding frequency. `freq_col` argument unnecessary if table consists just of a column of categories and a column of frequencies.
`tbl.hist(columns,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. Bins have the form [a, b).
`tbl.scatter(x_col, y_col)`	Draws a scatter plot consisting of one point for each row of the table
`tbl.plot(x_col, y_col)`	Draws a line plot of the data in the columns passed in

Table Predicates

Table predicates are used when a call to `tbl.where(column, predicate)` is made to filter a table by a condition. In the following examples, `x` represents a string or number and `val` represents the value a column takes for a given row. Here is a list of useful predicates:

Predicate	Description
`are.equal_to(x)`	Selects rows where `val == x`
`are.above(x)`	Selects rows where `val > x`
`are.below(x)`	Selects rows where `val < x`
`are.between(x, y)`	Selects rows where `x <= val < y`

Arrays

An array is a sequence of elements of the same type. You may assume that these functions exist in Python. In the examples below, `np` refers to the NumPy module, `tbl` refers to a Table object, `arr` refers to a NumPy array, and `num` refers to a number.

Function/Method	Description
`make_array(),` `make_array(val1,val2,...)`	Returns an array with the values passed in. If no values are passed in, returns an empty array.
`arr.item(i)`	Returns the element at the i-th index (the index of the first element is 0)
`np.append(arr, item)`	Returns a copy of `arr` with `item` appended to the end
`max(arr), min(arr)`	Returns the maximum or minimum of the sequence
`sum(arr)`	Returns the sum of all elements in the array
`len(arr)`	Returns the length (number of elements) in an array
`round(num), np.round(arr)`	Returns a number or array rounded to the nearest integer
`abs(num), np.abs(arr)`	Returns the absolute value of a number or array
`np.mean(arr)`	Returns the mean (a.k.a. average) of the values in the array
`np.median(arr)`	Returns the median of the values in the array
`np.std(arr)`	Returns the standard deviation of the values in the array
`np.arange(start, stop, step),` `np.arange(start, stop),` `np.arange(stop)`	Returns an array of numbers starting from `start`, going up in increments of `step`, and going up to but excluding `stop`. When `start` and or `step` are left out, by default, `step = 1` and `start = 0`
`np.count_nonzero(arr)`	Returns the number of nonzero elements in an array (`False` counts as 0, `True` as nonzero)
`np.random.choice(arr, n),` `np.random.choice(arr)`	Returns an array of `n` items sampled with replacement from an array. Default `n` is 1.

Additional Functions

You may assume that these functions exist in Python.

Function/Method	Description
`proportions_from_distribution(tbl,label,n)`	Takes in a table, column label corresponding to distribution proportions, and a sample size n. Returns a table augmented with the column "Random Sample" with the sampled proportions.
`percentile(n, arr)`	Returns the n-th percentile of an array
`stats.norm.cdf(x, mean, sd)`	Returns the area to the left of `x` under a normal curve with mean `mean` and standard deviation `sd`
`minimize(function)`	Returns the array of parameters that minimize `function`. If `function` takes one argument, parameter returned as number.