Data 8 Final Study Guide — Page 1

The 80th percentile is the value in a set that is at least as large as 80% of the elements in the set

For `s = [1, 7, 3, 9, 5]`, `percentile(80, s)` is 7

The 80th percentile is ordered element 4: `(80/100) * 5`

(Percentile) (Size of set)

For a percentile that does not exactly correspond to an element, take the next greater element instead

`percentile(10, s)` is 1 `percentile(20, s)` is 1
`percentile(21, s)` is 3 `percentile(40, s)` is 3

Inference: Making conclusions from random samples
Population: The entire set that is the subject of interest
Parameter: A quantity computed for the entire population
Sample: A subset of the population
In a **Random Sample**, we know the chance that any subset of the population will enter the sample, in advance
Statistic: A quantity computed for a particular sample

Estimation is a process with a random outcome
Population (fixed) → Sample (random) → Statistic (random)

A 95% **Confidence Interval** is an interval constructed so that it will contain the parameter for 95% of samples
For a particular sample, the interval either contains the parameter or it doesn't; the process works 95% of the time
Resampling: When we wish we could sample again from the population, instead sample from the original sample

---

Using a confidence interval to test a hypothesis:

- Null hypothesis: **Population mean = *x***
- Alternative hypothesis: **Population mean ≠ *x***
- Cutoff for P-value: *p*%
- Method:
  - Construct a (100-*p*)% confidence interval for the population average
  - If *x* is not in the interval, reject the null
  - If *x* is in the interval, fail to reject the null

Permutation test for comparing two samples

- **E.g.**: Among babies born at some hospital, is there an association between birth weight and whether the mother smokes?
- **Null hypothesis**: The distribution of birth weights is the same for babies with smoking mothers and non-smoking mothers.
- **Inferential Idea:** If maternal smoking and birth weight were not associated, then we could simulate new samples by replacing each baby's birth weight by a randomly picked value from among all babies.
  - Permute (shuffle) the outcome column K times. Each time:
    - Create a shuffled table that pairs each individual with a random outcome.
    - Compute a sampled test statistic that compares the two groups, such as the difference in mean birth weights.
  - Compare the observed test statistic to these sampled test statistics to see whether it is typical under the null.

The Central Limit Theorem (CLT)

If the sample is large, and drawn at random with replacement,
Then, *regardless of the distribution of the population,*
the probability distribution of the sample average (or sample sum) is roughly bell-shaped

- Fix a large sample size
- Draw all possible random samples of that size
- Compute the mean of each sample
- You'll end up with a lot of means
- The distribution of those is the *probability distribution of the sample mean*
- It's roughly normal, centered at the population mean
- The SD of this distribution is the (population SD) $/\sqrt{\text{sample size}}$

Choosing sample size so that the 95% confidence interval is small

- CLT says the distribution of a sample proportion is roughly normal, centered at population mean
- **95%** confidence interval:
  - Center **± 2 SDs** of the sample mean
- **CI Width** is 4 SDs of the sample mean
  = 4 x (SD of population)$/\sqrt{\text{sample size}}$
- If you know the max possible value of (SD of population), then you can solve for the sample size that gives you a small width

---

Distance between two examples

- Two attributes *x* and *y*: $D = \sqrt{(x_0 - x_1)^2 + (y_0 - y_1)^2}.$
- Three attributes *x, y,* and *z*:

$$D = \sqrt{(x_0 - x_1)^2 + (y_0 - y_1)^2 + (z_0 - z_1)^2}$$

To find the *k* nearest neighbors of an example:

- Find the distance between the example and each example in the training set
- Augment the training data table with a column containing all the distances
- Sort the augmented table in increasing order of the distances
- Take the top *k* rows of the sorted table

To classify a point:

- Find its *k* nearest neighbors
- Take a majority vote of the *k* nearest neighbors to see which of the two classes appears more often
- Assign the point the class that wins the majority vote

Expression	Description
`minimize(fn)`	Return an array of arguments that minimize a function.
`np.median(array)`	Return the median of an array.
`np.std(array)`	Return the standard deviation of an array of numbers
`table.row(i)`	Return the row of a table at index *i*.
`table.rows`	All rows of a table; Used in `for row in table.rows:`

Computing a confidence interval for an estimate from a sample:

- Collect a random sample
- Resample K times from the sample, with replacement
  - Compute the same statistic for each resampled sample
  - Take *percentiles* of the resampled estimates
  - 95% confidence interval: [2.5 percentile, 97.5 percentile]

```python
def bootstrap_mean(original_sample, label, replications):
    means = make_array()
    for i in np.arange(replications):
        bootstrap_sample = original_sample.sample()
        resampled_mean = np.mean(bootstrap_sample.column(label))
        means = np.append(means, resampled_mean)
    return means
resampled_means = bootstrap_mean(some_table, some_label, 5000)
right = percentile(97.5, resampled_means)
left = percentile(2.5, resampled_means)
confidence_interval = [left, right]
```

Mean (or average): Balance point of the histogram

- ***Not*** the "half-way point" of the data; the mean is not the median unless the histogram is symmetric
- If the histogram is skewed, then the mean is pulled away from the median in the direction of the tail

Standard deviation (SD) =

root	mean	square of	deviations from	average
5	4	3	2	1

Measures roughly how far off the values are from average

Most values are with the range "average ± z SDs"

- *z* measures "how many SDs above average"
- If *z* is negative, the value is below average
- *z* is a value in **standard units**
- Chebyshev: At most $1/z^2$ are more than z SDs from the mean
- Almost all standard unit values are in the range (-5, 5)
- Convert a value to standard units: (value - average) / SD

Percent in Range	All Distributions	Normal Distribution
average ± 1 SD	at least 0%	about 68%
average ± 2 SDs	at least 75%	about 95%
average ± 3 SDs	at least 88.888...%	about 99.73%

Correlation Coefficient (*r*) =

average of	product of	x in standard units	and	y in standard units

Measures how clustered the scatter is around a straight line

- -1 ≤ *r* ≤ 1 ; r = 1 (or -1) if the scatter is a perfect straight line
- *r* is a pure number, with no units
- *r* is not affected by changing units of measurement
- *r* is not affected by switching the horizontal and vertical axes

Regression to the mean: a statement about x and y pairs

- Measured in *standard units*
- Describing the deviation of x from 0 (the average of x's)
- And the deviation of y from 0 (the average of y's)

On average, y deviates from 0 less than x deviates from 0

Regression Line

$$y_{(su)} = r \times x_{(su)}$$

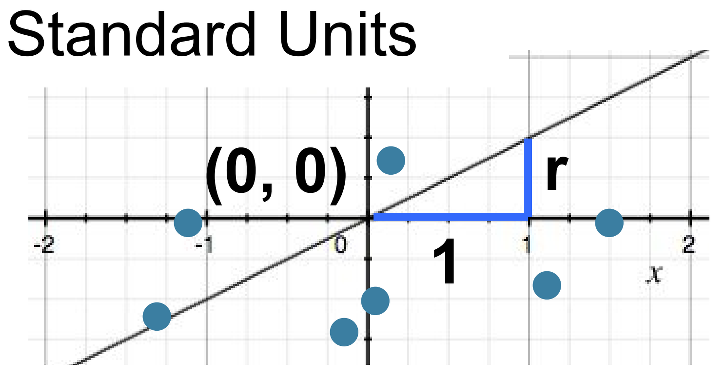
Correlation

In original units, the regression line has this equation:

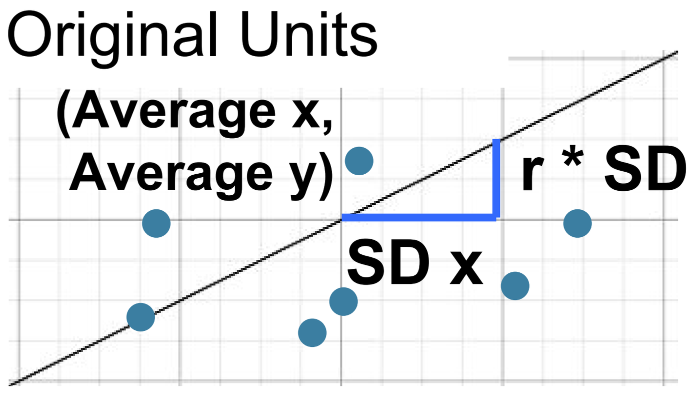
$$\frac{\text{estimate of } y - \text{average of } y}{\text{SD of } y} = r \times \frac{\text{the given } x - \text{average of } x}{\text{SD of } x}$$

y in standard units | x in standard units

Standard Units



Original Units



The regression line is the one that minimizes the (root) mean squared error of a collection of paired values

The slope and intercept are unique for linear regression

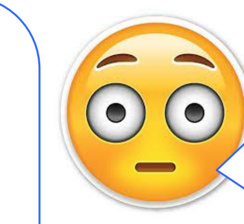
estimate of *y* = slope * *x* + intercept

$$\textbf{slope of the regression line} = r \cdot \frac{\text{SD of } y}{\text{SD of } x}$$

$$\textbf{intercept of the regression line} = \text{average of } y - \text{slope} \cdot \text{average of } x$$

We observed a positive slope and used it to make our predictions.

But what if the scatter plot got its positive slope just by chance?



What if the true line is actually FLAT?

- **Bootstrap the scatter plot & find the slope of the regression line through the bootstrapped plot many times.**
- Draw the empirical histogram of all the resampled slopes.
- Get the "middle 95%" interval: that's an approximate 95% confidence interval for the slope of the true line.
- **Null hypothesis:** The slope of the true line is 0.
  - Construct a bootstrap confidence interval for the true slope.
  - If the interval doesn't contain 0, reject the null hypothesis.
  - If the interval does contain 0, there isn't enough evidence to reject the null hypothesis.

- **Fitted value:** Height of the regression line at some x: a * x + b.
- **Residual**: Difference between y and regression line height at x.
- **Regression Model**: y is a linear function of x + normal "noise"
- Residual plot looks like a formless "noise" cloud under this model
- Average of residuals is always 0 for any scatter diagram

Properties of fitted values, residuals, and the correlation r:

(Variance of residuals) / (Variance of y) = $1 - r^2$

(Variance of fitted values) / (Variance of y) = r^2

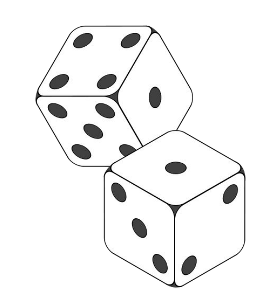
(Variance of y) = (Variance of fitted values) + (Variance of residuals)

Note: Variance is standard deviation squared.

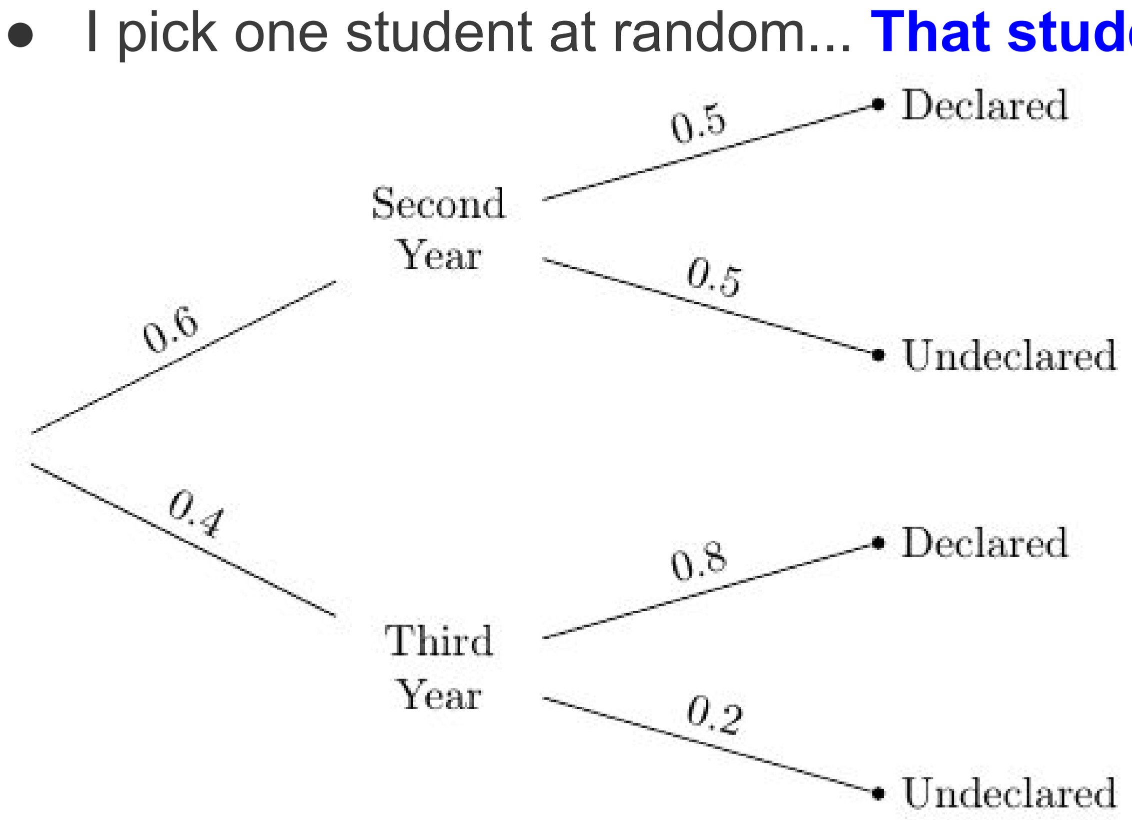
Probability Terminology

- **Experiment**: An occurrence with an uncertain outcome
- **Outcome**: The result of an experiment
- **Sample Space**: The set of all possible outcomes for the experiment
- **Event**: A subset of the possible outcomes
- **Probability**: The proportion of experiments for which the event occurs
- **Distribution**: The probability of all events

Scenario in which to apply Bayes Rule



- Class consists of second years (60%) and third years (40%)
- 50% of the second years have declared their major
- 80% of the third years have declared their major
- I pick one student at random... **That student has declared a major!**



Posterior probability:

P(Third Year | Declared)

$$= \frac{0.4 \times 0.8}{(0.6 \times 0.5) + (0.4 \times 0.8)}$$

Ways to accumulate statistics from multiple repetitions

Using an array:

```
stats_array = make_array()
for i in np.arange(1000):
    new_stat = ...
    stats_array = np.append(stats_array, new_stat)
```

Using a list:

```
stats_list = []
for i in np.arange(1000):
    new_stat = ...
    stats_list.append(new_stat)
```

Using a table:

```
stats_table = Table(['index', 'stat'])
for i in np.arange(1000):
    new_stat = ...
    stats_table.append([i, new_stat])
```