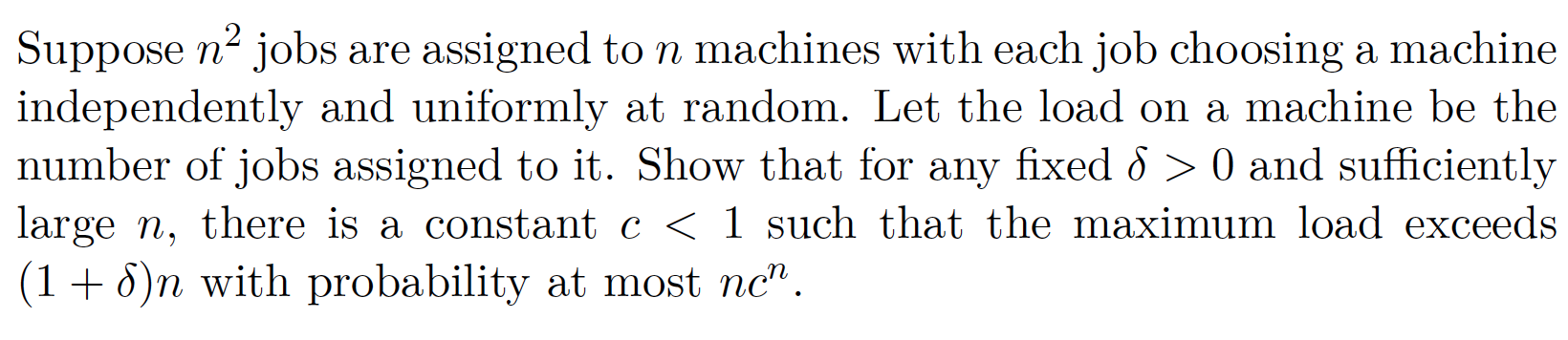
CS/Math 335: Activity 16, October 3

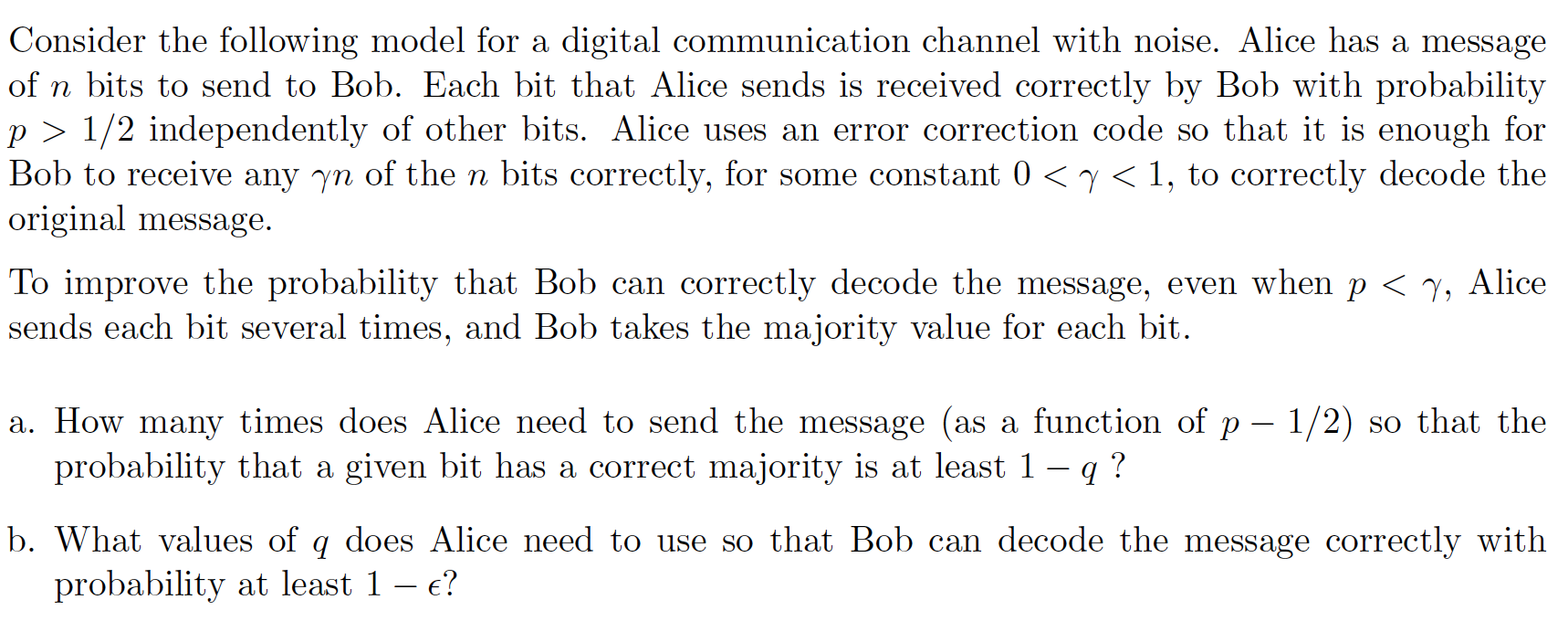
Before class please do Example 1, 5-9, 11, and (1) – (10) on the last page.

First, a review of the Chernoff bounds from 4.2.1…

**Example 1**:



**Example 2**:



Application to Confidence Intervals (Section 4.2.3):

Computer Science is often concerned with **approximation algorithms**, which approximately compute some quantity of interest. Statistics, as a field, is concerned with the same problem. In statistics, there is a **population** of size N >> 0, and it is too costly to study every individual of the population. Thus, a **sample** of size n << N is taken, and statistics about the sample (e.g. its mean, median, standard deviation) are used to approximate the corresponding unknown parameters of the population.

**Example 3**: In a random sample of n = 100 American male adults, the average height in the sample was 178cm (about 5’10’’). This number approximates the true average height of all American male adults. Computing exactly would require measuring N = 120,000,000 men.

We call a **point estimate** of . Even better is to have a **confidence interval** that we are fairly certain lies within. Specifically, we seek an interval of the form (where MOE stands for “margin of error”) such that we can be 95% confident that lies inside [ - MOE, + MOE]. The interval will depend on the sample, how much variability it has, and the sample size n.

|  |  |
| --- | --- |
| To be **95% confident** means that if we took *k*  samples and computed *k* confidence intervals,  then as , we expect 95% of the intervals  to contain the true mean. Here is a picture  of k = 100 confidence intervals for a true mean  of . Each interval is centered around  its own value, from its own sample. Green  intervals contain , while red intervals fail to  contain . Notice that 95 of the 100 intervals  contain . We could also choose other levels  of confidence, e.g. 98%, 99%, or in general  () x 100% confident (meaning we expect  an interval to be wrong with probability ).  Note: the book uses instead of , and uses  to denote the margin of error. | ../../../Desktop/Screen%20Shot%202016-09-25%20at%209.06.52%20AM |

We can make confidence intervals for any population parameter, e.g. the median, the standard deviation, the proportion of successes in the population, a difference in means between two subgroups of the population (e.g. male vs. female salaries), a difference in proportion, etc. Let denote the true population standard deviation, and let *s* be the sample approximation to . For populations that can be classified as success vs. failure, let *p* denote the true proportion of

successes in the population, and let (denoted in the book) denote the sample proportion, i.e. the number *x* of successes (in the sample) divided by the sample size n.

Three algorithms to compute confidence intervals

Sampling Distribution Algorithm:

1. Select *k* sufficiently large, e.g. k = 10,000
2. Select *n* large enough that sample statistics won’t be overly affected by outliers, e.g. n=50
3. Take k many samples of size *n* from the population, and write down the sample statistic of each (e.g. the of each, or the of each).
4. Make a dotplot with *k* dots, one for each sample statistic value (e.g. each value). This is called the **sampling distribution**.
5. Select the end-points *a* and *b* that leave 2.5% of the dots less than *a*, 2.5% of the dots greater than *b*, hence 95% of the dots within *[a,b]*. Output *[a,b]*.

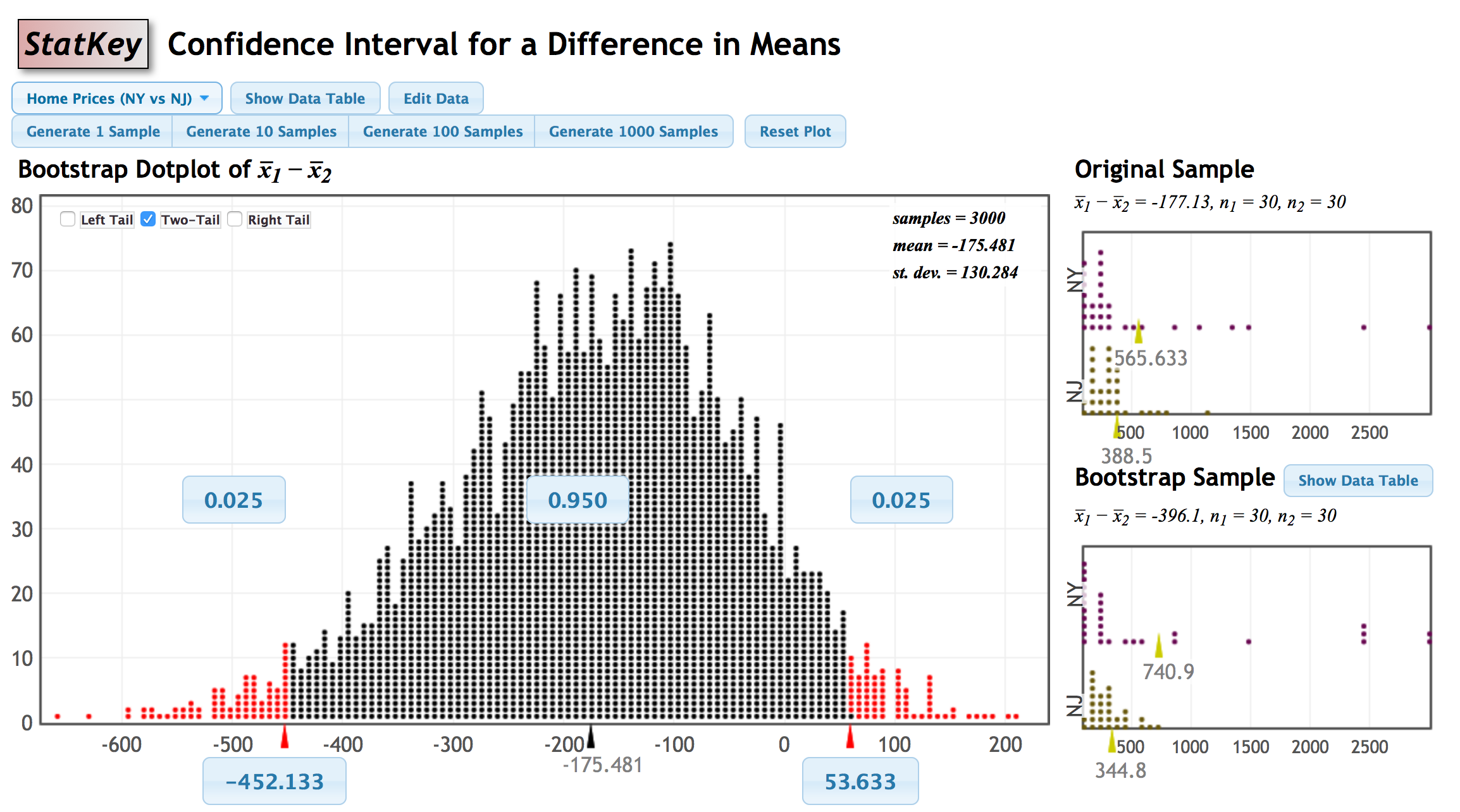
This algorithm is impractical because of Step 3, which takes too long and is too costly. A more realistic algorithm is:

Bootstrapping Algorithm:

1. Select *k* sufficiently large, e.g. k = 10,000
2. Select *n* large enough that sample statistics won’t be overly affected by outliers, e.g. n=50
3. Take 1 sample S of size *n* from the population
4. Create *k* resamples by drawing *n* items **with replacement** from S, and write down the sample statistic of each (e.g. the of each, or the of each).
5. Make a dotplot with *k* dots, one for each resample statistic value (e.g. each value). This is called the **bootstrap distribution**.
6. Select the end-points *a* and *b* that leave 2.5% of the dots less than *a*, 2.5% of the dots greater than *b*, hence 95% of the dots within *[a,b]*. Output *[a,b]*.

This algorithm works if the sample S is representative of the population. In that case, drawing from S with replacement is analogous to drawing from the true population, because each item in S can be thought of as representing many items in the population. This algorithm breaks down for n < 12, but works well for n > 24 for any reasonable population. Write out pseudocode to implement this algorithm with input S, with k = 10,000, and to compute a confidence interval for . What would you change to create a confidence interval for ?

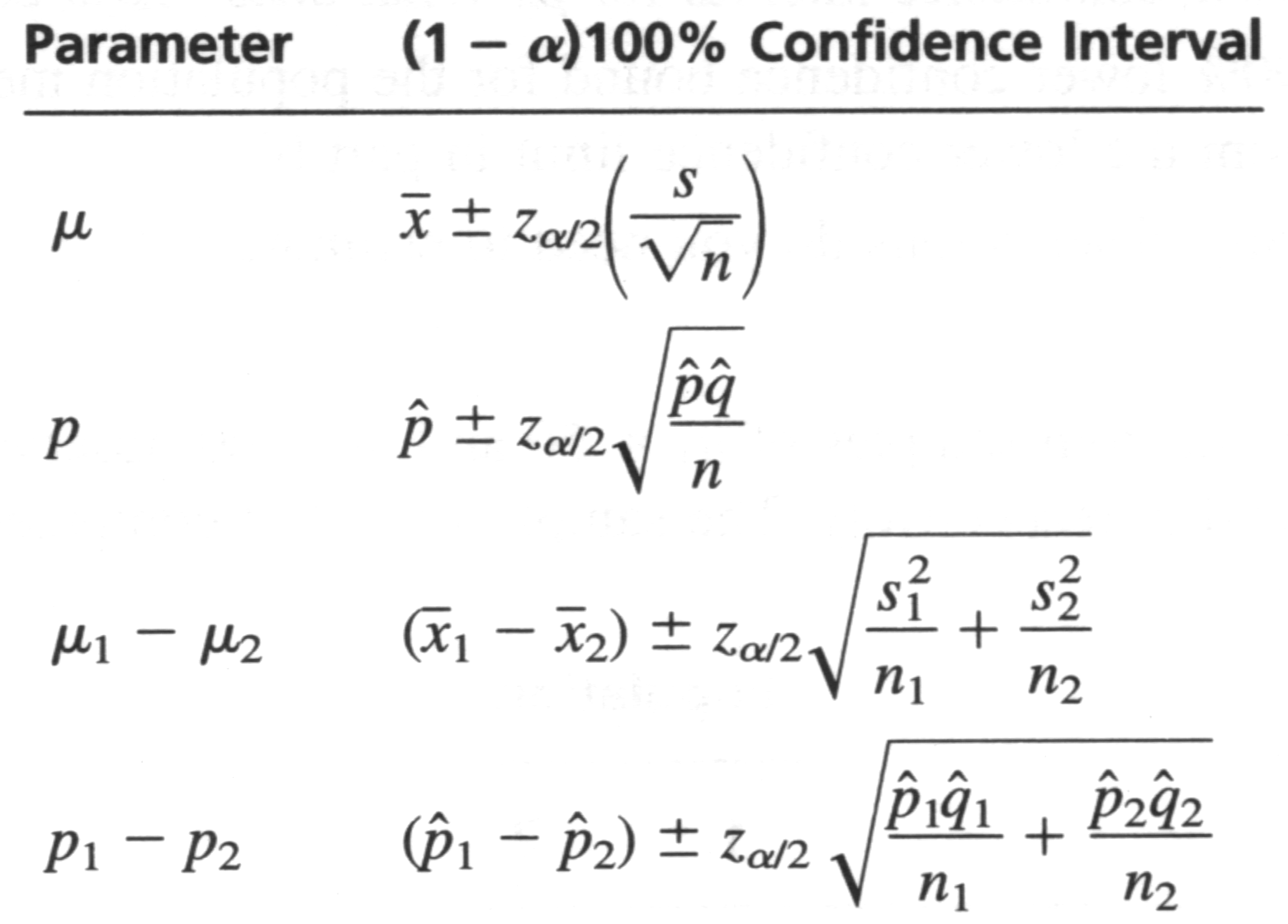
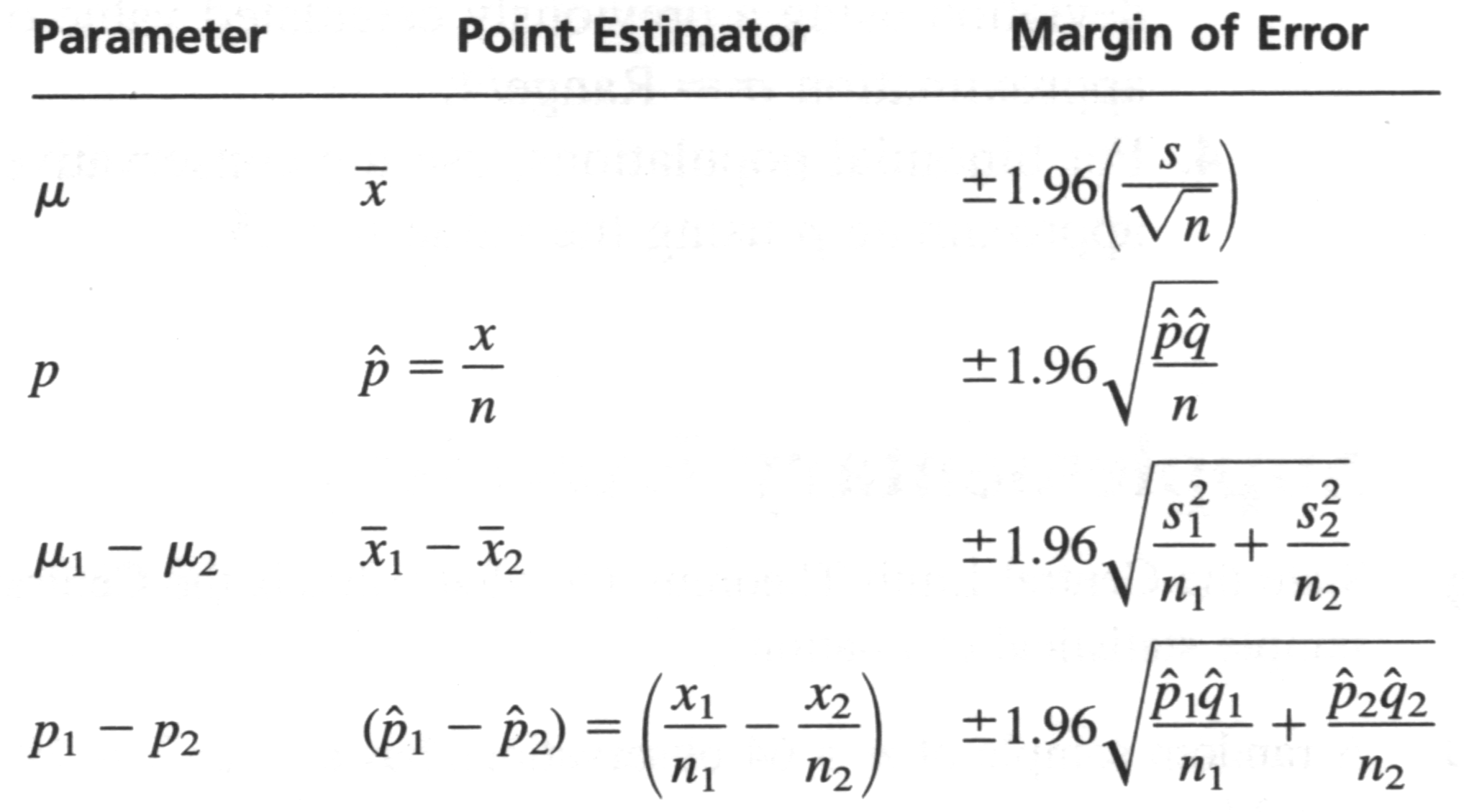
**Example 4**: for the bootstrap distribution below (with k = 3,000), we would output the interval [-452.133, 53.633] and we would be 95% confident that the true difference in home prices between NY and NJ is in that interval.



Historically, step 4 of the bootstrap algorithm was impossible due to lack of computers, so a parametric approximation (the normal distribution) to the sampling distribution was used.

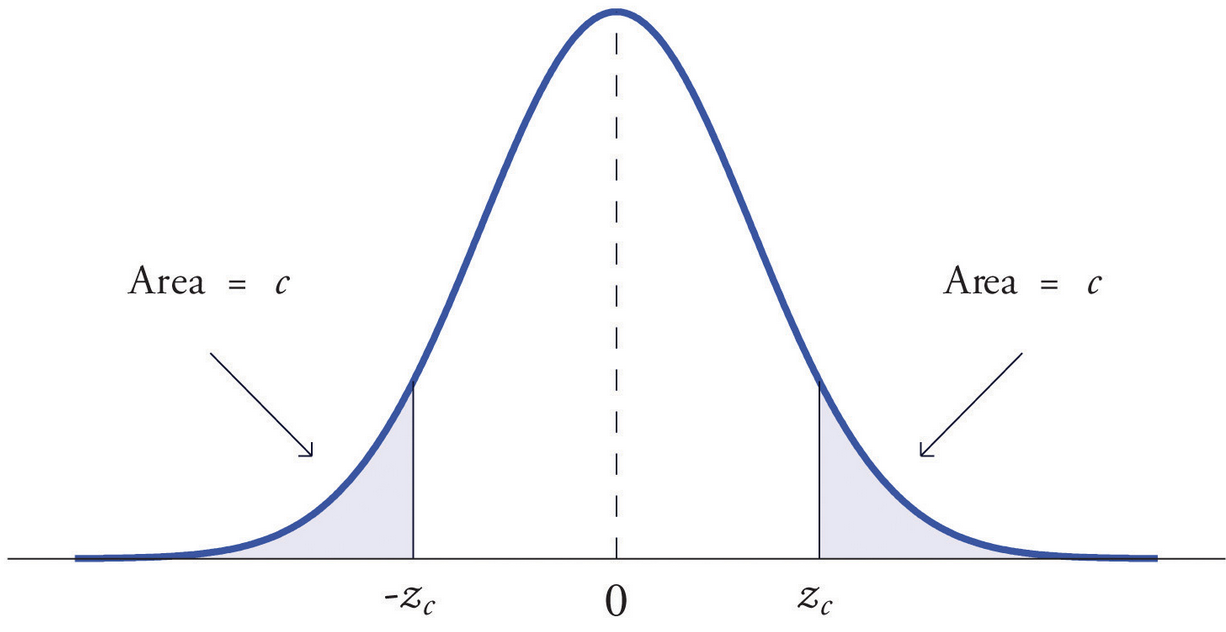
Parametric Confidence Interval Algorithm:

1. Select the sample size *n* and take a sample S of size *n*.
2. Compute the point estimate from S (e.g. , or , or *s*) for your parameter of interest.
3. Use the table below to compute the margin of error. For a 95% confidence interval, we use 1.96 as the cutoff value. For a general , we must look up the cutoff value in a table of standard normal values. The cutoff value leaves in each tail (see below).
4. Output



This algorithm only works if the sampling distribution is approximately normal. Using the central limit theorem, we need the original data to be roughly symmetric (not skewed) and we need n > 30 for confidence intervals, and we need n > 5 and n(1-) > 5 for intervals.

Picture to keep in mind for cutoff values. For us, c = . Example:



**Example 5:** Let *p* be the proportion of 2014 Macbook Pro 15’’ screen computers that crash on a given day. Suppose we collect a random sample of 300 users of these computers and 40 of them reported that their computer crashed on the day of the study. Compute a 95% confidence interval for *p*.

The upside of the parametric algorithm is that you can reverse engineer it to figure out how large to make *n*, if you want to be () x 100% certain that your sample statistic is within of the true population parameter value. As *n* grows, we have more information, so we can pin down the value of the parameter more precisely.

**Example 6**: Suppose we want to be 99% certain ( that our is within B = .5 of the true mean . This means we want MOE < .5. Suppose a pre-study told us that s = 2 is a good approximation to the standard deviation. Solve the equation to get a lower bound for n that will guarantee MOE < B. Here, you are solving 2.33 \* (2 / < .5.

**Example 7**: Now solve for *n* when we want MOE < .05. Notice the trade-off between precision of our estimate and number of people we must sample from the population.

The book suggested using Chernoff bounds to find a bound on *n* when we are computing confidence intervals for proportions, as in Example 7. Let’s compare methods. Suppose we want to be 95% certain that our sample statistic is within of the true *p*. For example, maybe *p* is the proportion of faulty Intel chips of some type, and Intel has asked us to estimate this proportion so that they can see if they need to invest in making chips that are less faulty.

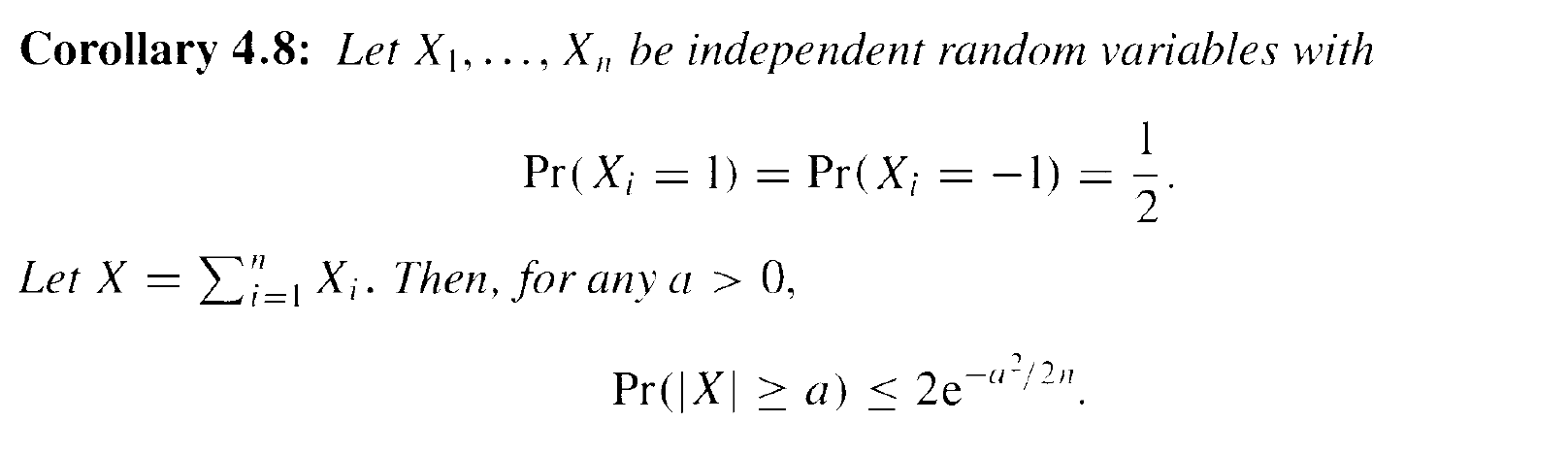
**Example 8:** In the Chernoff bound method, we use that

Pr(*p* not in [) <

For a 95% confidence interval, the probability on the left is .05. Solve this equation to find a lower bound for *n*, when , when , and when . How does the analysis change if 95% is replaced by () x 100%?

**Example 9**: Use the margin of error method to find lower bounds on *n*, for the same values of the margin of error as above. Compare your bounds on *n* to the bounds above. Which method is better? How does the analysis here change if 95% is replaced by () x 100%? Do we have a closed-form expression for the values of ?

Now we turn to Section 4.3 …



**Example 10**: Let X1, X2, …, X8 be flips of a fair coin, with values 1 for heads and -1 for tails. X can take values -4, -3, -2, …, 3, 4. The mean of X is 0 and n = 8. Use (4.8) to bound P(|X|4):

P(|X| 4) < . Since Y = X+4 is binomial, in this case we can compute the true probability of Y or Y = 8 as 2\*.004 = .008. So the bound of .736 is not sharp.

**Example 11**: Let X be the sum of 100 copies of Xi as above, so 50 X -50. Determine what (4.8) says about P(X 25 or X -25):

**Example 12**: Let X be the sum of 40 independent dice rolls. Can you use (4.8) to bound the probability that |X| 20? Why or why not?

**Example 13**: Quantitative hedge funds create models for the stock market so that their algorithms for picking stocks make more money than choosing stocks randomly. We say a stock is **up** at the end of the day if it’s worth more than it was at the start of the day. A simplistic model for a particular stock S suggests that, each day,

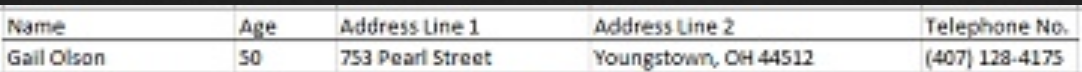
Pr(S valued higher at the end of the day than at the start of the day) = ½

Pr(S valued lower at the end of the day than at the start of the day) = ½

Furthermore, knowledge of whether or not the stock was up on previous days cannot be used to predict whether it’ll be up today. Use (4.8) to bound the probability that, over the course of a year (n = 365), the stock will be down 100 days more than it will be up (if this event occurs, it might be cause to reject the model above). Comment on whether a different Chernoff bound might be better, and why. If you have time, try to think of more realistic models for the behaviors of stocks.

Section 4.4…

When data comes to us in a spreadsheet, the rows are individuals and the columns are different variables/features we are measuring on each individual, e.g. in the following row



we have 5 features: Name, Age, …, Phone Number.

1. Give an example of a real world scenario where you might want to divide a dataset into two disjoint groups (as in 4.4) where each feature is balanced between the two groups.
2. Compute for the matrix A and the vector b below:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 1 | 0 | 1 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 |

|  |
| --- |
| 1 |
| -1 |
| -1 |
| 1 |
| 1 |
| 1 |
| -1 |
| 1 |

1. Find a vector b that you think *minimizes* for the matrix above.
2. Can you think of any non-brute force way to search for b? If not, then perhaps the problem is NP complete. If you’ve learned about complexity theory, comment on this.
3. What’s the asymptotic lower bound on ? How far is the book’s algorithm off from optimal? Hint: the answer is in the reading, right before Theorem 4.11.
4. The proof of Theorem 4.11 breaks into two cases: k small vs. k large. Intuitively, keeping in mind that the goal is to find b to minimize , why is the k small case easy?
5. Consider the proof of Theorem 4.11, where the book says “clearly.” Recalling that the entries of b are all 1 or -1, what vector b maximizes |ci|? What is that maximum value of |ci| as a function of k? Is that line in the proof clear now?
6. For the k large case, we write the bad event as a union of n events. Write out the bad event as a sentence:
7. Are the n individual events mutually disjoint? Does this matter for the union bound?
8. How did the book get rid of the m and k in the last inequality of the proof?