# Asmt 1: Experimenting with Statistical Principles

Turn in through Canvas by 5pm:
Wednesday, January 29

## Overview

In this assignment you will experiment with random variation over discrete events.

It will be very helpful to use the analytical results and the experimental results to help verify the other is correct. If they do not align, you are probably doing something wrong (this is a very powerful and important thing to do whenever working with real data).

*As usual, it is highly recommended that you use LaTeX for this assignment. If you do not, you may lose points if your assignment is difficult to read or hard to follow. Find a sample form in this directory:* `http://www.cs.utah.edu/~jeffp/teaching/latex/`

## 1 Birthday Paradox (7 points)

Consider a domain of size $n = 2000$.

**A: (1 points)** Generate random numbers in the domain $[n]$ until two have the same value. How many random trials did this take? We will use $k$ to represent this value.

**B: (3 points)** Repeat the experiment $m = 250$ times, and record for each how many random trials this took. Plot this data as a *cumulative density plot* where the $x$-axis records the number of trials required $k$, and the $y$-axis records the fraction of experiments that succeeded (a collision) after $k$ trials. The plot should show a curve that starts at a $y$ value of 0, and increases as $k$ increases, and eventually reaches a $y$ value of 1.

**C: (1 points)** Calculate the empirical expected value of the number of $k$ random trials in order to have a collision. That is, add up all values $k$, and divide by $m$.

**D: (2 points)** Describe how you implemented this experiment and how long it took for $m = 250$ trials.

Show a plot of the run time as you gradually increase the parameters $n$ and $m$. Try to reach $n = 1{,}000{,}000$ and $m = 10{,}000$. If your implementation cannot finish at these values, explain why, and what would be needed to fix it.

## 2 Coupon Collectors (7 points)

Consider a domain of size $n = 50$.

**A: (1 points)** Generate random numbers in the domain $[n]$ until every value $i \in [n]$ has had one random number equal to $i$. How many random trials did this take? We will use $k$ to represent this value.

**B: (3 points)** Repeat step $A$ for $m = 250$ times, and record for each the value $k$ of how many random trials we required to collect all values $i \in [n]$. Make a cumulative density plot as in 1.B.

**C: (1 points)** Use the above results to calculate the empirical expected value of $k$.

**D: (2 points)** Describe how you implemented this experiment and how long it took for $n = 60$ and $m = 250$ trials.

Show a plot of the run time as you gradually increase the parameters $n$ and $m$. Try to reach $n = 20{,}000$ and $m = 5{,}000$. If your implementation cannot finish at these values, explain why, and what would be needed to fix it.

## 3 Comparing Experiments to Analysis (4 points)

**A: (2 points)** Calculate analytically (using the formulas from class) the number of random trials needed so there is a collision with probability at least $0.5$ when the domain size is $n = 2000$. (Show your work.)

How does this compare to your results from Q1.C?

**B: (2 points)** Calculate analytically (using the formulas from class) the expected number of random trials before all elements are witnessed in a domain of size $n = 50$? (Show your work.)

How does this compare to your results from Q2.C?

## 4 Random Numbers (2 points)

**A: (2 points)** Consider when the only random function you have is one that choses a bit at random. In particular `rand-bit()` returns `0` or `1` at uniformly random. How can you use this to create a random integer number between 1 and $n = 1000$? How many calls does this take in terms of $n$ (say I were to increase the value $n$, how does the number of calls to `rand-bit()` change)?

In many settings generating a random bit is much more expensive that a standard CPU operation (like an addition or a memory reference), so it is critical to minimize them. Say I wanted to generate many random integers between 1 and $n = 1000$, then how could I reduce the number of calls to `rand-bit()`?

## 5 BONUS (2 points)

Consider a domain size $n$ and let $k$ be the number of random trials run. Let $f_i$ denote the number of trials that have value $i$. Note that for each $i \in [n]$ we have $\mathbf{E}[f_i] = k/n$. Let $\mu = \max_{i \in [n]} f_i/k$.

Consider some parameter $\delta \in (0, 1)$. How large does $k$ need to be for $\mathbf{Pr}[|\mu - 1/n| \geq 0.01] \leq \delta$? That is, how large does $k$ need to be for *all* counts to be within $1\%$ of the average with probability $\delta$?

How does this change if we want $\mathbf{Pr}[|\mu - 1/n| \geq 0.001] \leq \delta$ (for $0.1\%$ accuracy)?

*(Make sure to show your work)*