Programs for The Probability Lifesaver

Steven Miller (sjm1@williams.edu)

```
Print["Here are some suggestions for programming projects. "]
Print["Links below are clickable."]
Print["Look at the template and introductory
   lecture to Mathematica which I've posted on my homepage:"]
Print[Hyperlink["Miller LaTeX and Mathematica Handouts",
  "http://web.williams.edu/Mathematics/sjmiller/public_html/math/handouts/latex.
    htm"]]
Print[Hyperlink["Miller YouTube lecture on Mathematica",
  "http://www.youtube.com/watch?v=g1oj7CIqGM8"]]
Print[Hyperlink["Miller Mathematica template",
  "http://web.williams.edu/Mathematics/sjmiller/public_html/math/LaTexMathematica/
    MathematicaIntroVer6.nb"]]
Print[
  "Try some of the Project Euler problems; while they can get challenging (from
    a coding perspective) quickly, the first few are not that bad and
    I've provided solutions in Mathematica to a smattering of them."];
Print[Hyperlink["Project Euler Homepage", "https://projecteuler.net/"]]
Print[Hyperlink["My Project Euler Mathematica Solutions",
  "http://web.williams.edu/Mathematics/sjmiller/public_html/pbook/"]]
Print["\nHere are some general problems to test your coding skills.\n
In general you want to write a series of lines
    to do a certain task once, and then do a For loop many
    times to numerically approximate the true probability.\n
(1) Write code to numerically approximate the probability a randomly
    chosen hand of five cards has five different numbers.
(2) Write code to flip a fair coin T times and determine the longest run of
    heads, the longest run of tails, and the longest run of heads or tails.
(3) Assume each team in a best of seven series has a 50% chance of
    winning each game. Write code to numerically approximate
    the probability that a team first wins four games
    after G games have been played, for G =  4, 5, 6 or 7.
```

(4) Flip a fair coin T times; for each head take one step to the right, for each tail take one step to the left. Write code to numerically approximate the probability that you return to your starting point at least once in the T tosses.
(5) Generalize the above problem to 2-dimensions, and now at each toss you have a 25% chance of moving left, right, up or down.
(6) Generalize the generalization so you now move in 3-dimensions.
(7) Fix an irrational number x, such as π, e, Euler's γ or the Golden mean. For a large L, look at the leading digit of x^n base 10, and see how often it is a 1, a 2, ..., a 9. What do you think those probabilities should be? Are you surprised?
(8) Write code to numerically approximate the probability that in a hand of five cards, all cards equally likely to be chosen and all cards chosen independently of each other, that you have four cards of the same number.
(9) Redo the previous problem, but for at least three cards of the same suit.
(10) Redo the previous problem, but for at least two cards of the same suit. Are you surprised?
"];

Here are some suggestions for programming projects.

Links below are clickable.

Look at the template and introductory
lecture to Mathematica which I've posted on my homepage:

Miller LaTeX and Mathematica Handouts

Miller YouTube lecture on Mathematica

Miller Mathematica template

Try some of the Project Euler problems; while they can get challenging
(from a coding perspective) quickly, the first few are not that bad
and I've provided solutions in Mathematica to a smattering of them.

Project Euler Homepage

My Project Euler Mathematica Solutions

Here are some general problems to test your coding skills.

In general you want to write a series of lines to do a certain task once, and then do a For loop many times to numerically approximate the true probability.

(1) Write code to numerically approximate the probability a randomly chosen hand of five cards has five different numbers.
(2) Write code to flip a fair coin T times and determine the longest run of heads, the longest run of tails, and the longest run of heads or tails.
(3) Assume each team in a best of seven series has a 50% chance of winning each game. Write code to numerically approximate the probability that a team first wins four games after G games have been played, for G = 4, 5, 6 or 7.
(4) Flip a fair coin T times; for each head take one step to the right, for each tail take one step to the left. Write code to numerically approximate the probability that you return to your starting point at least once in the T tosses.
(5) Generalize the above problem to 2-dimensions, and now at each toss you have a 25% chance of moving left, right, up or down.
(6) Generalize the generalization so you now move in 3-dimensions.
(7) Fix an irrational number x, such as π, e, Euler's γ or the Golden mean. For a large L, look at the leading digit of x^n base 10, and see how often it is a 1, a 2, ..., a 9. What do you think those probabilities should be? Are you surprised?
(8) Write code to numerically approximate the probability that in a hand of five cards, all cards equally likely to be chosen and all cards chosen independently of each other, that you have four cards of the same number.
(9) Redo the previous problem, but for at least three cards of the same suit.
(10) Redo the previous problem, but for at least two cards of the same suit. Are you surprised?

Chapter I : Introduction

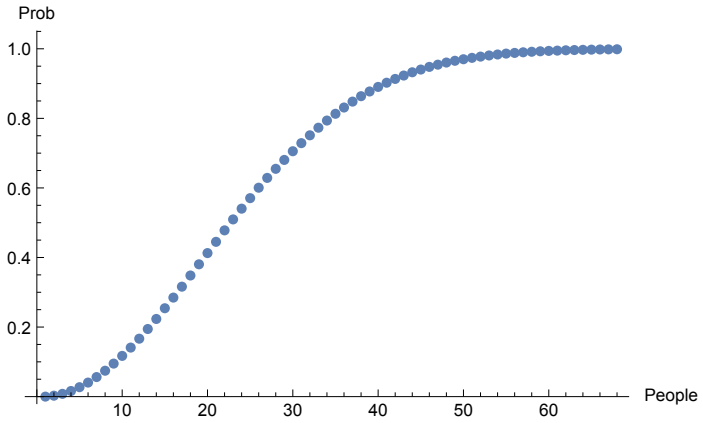
Assuming each day in the year is equally likely to be someone' s birthday, how many people do we need to have a 50 % chance two share a birthday (all birthdays are assumed to be independently chosen).

```
birthdaycdf[num_, days_] := Module[{},
   (* num is the number of times we do it *)
   (* days is the number of days in the year *)
   For[d = 1, d ≤ days, d++, numpeople[d] = 0];
   (* initializes to having d people be where the share happens
    to zero *)
```

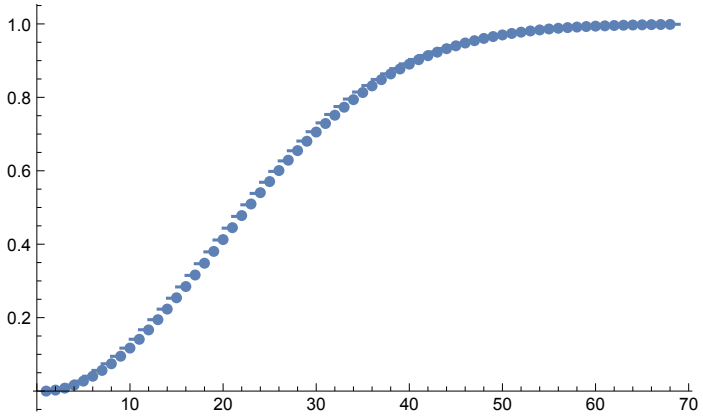
```
For[n = 1, n ≤ num, n++,
 { (* begin n loop *)
  share = 0;
  bdaylist = {}; (* will store bdays of people in room here *)
  k = 0; (* initialize to zero people *)
  While[share == 0,
   {
    (* randomly choosee a new birthday *)
    x = RandomInteger[{1, days}];
    (* see if new birthday in the set observed *)
    (* if no add, if yes won and done *)
    If[MemberQ[bdaylist, x] == False,
     bdaylist = AppendTo[bdaylist, x],
     share = 1];
    k = k + 1; (* increase number people by 1 *)
    (* if just shared a birthday add one from that person
     onward *)
    If[share == 1, For[d = k , d ≤ days, d++,
       numpeople[d] = numpeople[d] + 1];
    ]; (* records when had match *)
    (*  as doing cdf do from that point onward *)
   }]; (* end while loop *)
 }]; (* end n loop *);

bdaylistplot = {};
max = 3 * (.5 + Sqrt[days Log[4]]);
For[d = 1, d ≤ max, d++,
 bdaylistplot =
  AppendTo[bdaylistplot, {d, numpeople[d] 1.0 / num}]
]; (* end of d loop *)
(* prints obs prob of shared birthday as a function of people*)
Print[ListPlot[bdaylistplot, AxesLabel → {People, Prob}]];
Print["Observed probability of success with 1/2 + Sqrt[D log(4)] people is ",
 numpeople[Floor[.5 + Sqrt[days Log[4.]]] * 100.0 / num, "%."];
(* this is our theoretical prediction *)
f[x_] := 1 - Product[1 - k / days, {k, 0, Floor[x]}];
(* this prints our obseerved data and our predicted at
 the same time using show *)
Print[Show[Plot[f[x], {x, 1, max}],
  ListPlot[bdaylistplot, AxesLabel → {People, Prob}]
 ]];
theorybdaylistplot = {};
For[d = 1, d ≤ max, d++,
```

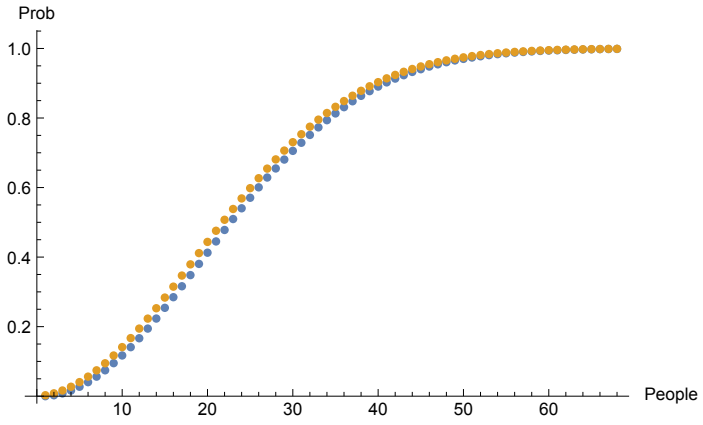
```
    theorybdaylistplot = AppendTo[theorybdaylistplot, {d, f[d]}]];
   Print[
    ListPlot[{bdaylistplot, theorybdaylistplot}, AxesLabel → {People, Prob}]];
  ];
```

```
birthdaycdf[100 000, 365]
```




```
Observed probability of success with 1/2 + Sqrt[D log(4)] people is 47.801%.
```





Three people independently draw a card from a deck (with replacement); the first to get a diamond wins. What is the probability each wins?

```
diamonddraw[num_] := Module[{},
   awin = 0; bwin = 0; cwin = 0; (* initialize win counts to 0 *)
   For[n = 1, n ≤ num, n++,
    { (* start of n loop *)
     diamond = 0;
     While[diamond == 0,
      { (* start of diamond loop, keep doing till get diamond *)
       (* randomly choose a card for each of three players, with replacement*)
       (* we'll order the deck so first 13 cards are the diamonds *)
       c1 = RandomInteger[{1, 52}];
       c2 = RandomInteger[{1, 52}];
       c3 = RandomInteger[{1, 52}];
       (* if one is a diamond we win and will stop *)
       If[c1 ≤ 13 || c2 ≤ 13 || c3 ≤ 13, diamond = 1];
       (* give credit to winner *)
       If[diamond == 1,
        If[c1 ≤ 13, awin = awin + 1,
         If[c2 ≤ 13, bwin = bwin + 1,
          If[c3 ≤ 13, cwin = cwin + 1]]]
       ]; (* end of if loop on diamond = 1 *)
      }]; (* end of while diamond loop *)
    }]; (* end of n loop *)
   Print["Here are the observed probabilities from ", num, " games."];
   Print["Percent Alice won (approx): ", 100.0 awin / num, "%."];
   Print["Percent Bob won (approx): ", 100.0 bwin / num, "%."];
   Print["Percent Charlie won (approx): ", 100.0 cwin / num, "%."];
   Print["Predictions (from our bad logic) were approx ",
    1600.0 / 37, " ", 4800.0 / 175, " ", 900.0 / 37];
  ];

diamonddraw[1 000 000]
```

```
Here are the observed probabilities from 1 000 000 games.
Percent Alice won (approx): 43.2202%.
Percent Bob won (approx): 32.4069%.
Percent Charlie won (approx): 24.3729%.
Predictions (from our bad logic) were approx 43.2432 27.4286 24.3243
```

Chapter 2 : Basic Probability Laws

Chapter 3 : Counting I : Cards

```
fullhousesearch[num_] := Module[{},
   (* creates a deck of cards, as only care about numbers *)
   (* we ignore suits and write each number four times *)
   cards = {};
   fullhouse = 0;
   For[d = 1, d ≤ 13, d++,
    For[i = 1, i ≤ 4, i++, cards = AppendTo[cards, d]]];
   (* main code here, do num times *)
   For[n = 1, n ≤ num, n++,
    {
     (* the code randomly chooses 5 from 52 cards and sorts. *)
     (* for the analysis below it's very easy to check to see *)
     (* if we have a full house if the hand is sorted *)
     (* we have a full house if the first three are the same *)
     (* and the last two are the same, or the first two are *)
     (* the same and the last three are *)
     hand = Sort[RandomSample[cards, 5]];
     If[hand[[1]] == hand[[2]] && hand[[4]] == hand[[5]],
      If[hand[[3]] == hand[[2]] || hand[[3]] == hand[[4]],
       fullhouse = fullhouse + 1]];
    }];
   Print["Percent of time got full house is ", 100.0 fullhouse / num, "."];
   (* we now print out the two predictions, and see that the *)
   (* first is very close to our numerics *)
   Print["The two predictions were 0.144058% and 0.072029%."];
  ];
```

In[51]:=
```
fullhousesearch[1 000 000]
```

```
Percent of time got full house is 0.1447.
The two predictions were 0.144058% and 0.072029%.
```

Chapter 4 : Conditional Probability, Independence and Bayes' Theorem