DATA 1010
Problem Set 7
Due 26 October 2018 at 11 PM

Problem 1

Find the mean and variance of a continuous random variable U whose distribution is uniform over the interval $[a, b]$.

Solution

We have $\frac{1}{b-a}\int_a^b x\,\mathrm{d}x = \frac{b^2-a^2}{2(b-a)} = \frac{a+b}{2}$, and $\int_a^b x^2\,\mathrm{d}x = \frac{b^3-a^3}{3(b-a)} = \frac{a^2+ab+b^2}{3}$, so

$$\operatorname{Var} U = \mathbb{E}[U^2] - \mathbb{E}[U] = \frac{a^2+ab+b^2}{3} - \left(\frac{a+b}{2}\right)^2 = \frac{(b-a)^2}{12}.$$

Problem 2

The *skewness* of a distribution ν is a measure of its asymmetry about its mean. It is defined to be

$$\mathbb{E}\left[\left(\frac{X-\mu}{\sigma}\right)^3\right],$$

where X is a random variable with distribution ν, μ is the mean of X, and σ is the standard deviation of X. Find the skewness of the exponential distribution with parameter 1. You should set up the integrals on your own, but feel free to evaluate them using a symbolic computation system.

Solution

We begin by calculating

$$\mathbb{E}[X] = \int_0^\infty x\lambda \mathrm{e}^{-\lambda x}\,\mathrm{d}x = 1/\lambda,$$

and

$$\mathbb{E}[X^2] = \int_0^\infty x^2\lambda \mathrm{e}^{-\lambda x}\,\mathrm{d}x = 2/\lambda^2,$$

so $\sigma = \sqrt{\mathbb{E}[X^2] - \mathbb{E}[X]^2} = 1/\lambda$. Finally,

$$\mathbb{E}\left[\left(\frac{X-\mu}{\sigma}\right)^3\right] = \int_0^\infty \frac{(x-\mu)^3}{\sigma^3}\lambda \mathrm{e}^{-\lambda x}\,\mathrm{d}x = 2$$

Incidentally, one way to evaluate these integrals is to use SymPy (but you can also use Wolfram Alpha or Symbolab):

```
using SymPy
@vars x λ
integrate(x*λ*exp(-λ*x),(x,0,oo))
integrate(x^2*λ*exp(-λ*x),(x,0,oo))
integrate((x-1/λ)^3/(1/λ)^3*λ*exp(-λ*x),(x,0,oo))
```

Problem 3

Consider an increasing* function $F : \mathbb{R} \to [0, 1]$ whose limits at $-\infty$ and $+\infty$ are 0 and 1, respectively. (*Note: this means that $F(x) \le F(y)$ whenever $x \le y$).

(a) Consider the following algorithm: begin with a random variable Y whose distribution is uniform in $[0, 1]$, and identify the x-value where the graph of F hits the horizontal line $y = Y$. Show that this random variable X has CDF F.

(b) Using (a), show that `-log(rand())/λ` returns exponential random variable with parameter λ.

Solution

(a) We need to show that $\mathbb{P}(X \leq x) = F(x)$ for any $x \in \mathbb{R}$, since $x \mapsto \mathbb{P}(X \leq x)$ is the CDF of F, by definition. Since F is an increasing function, we know that the event $\{X \leq x\}$ is equal to the event $\{Y \leq F(x)\}$. Since Y is uniform on $[0,1]$, the probability of this event is $F(x)$.

(b) In part (a), we learned that we can generate a sample from any distribution specified by its CDF F: we apply the inverse function F^{-1} to a uniform random variable. Since the inverse function of $F(x) = 1 - e^{-\lambda x}$ is $y \mapsto -\log(1-y)/\lambda$, we know that $-\log(1-Y)/\lambda$ is exponentially distributed with parameter λ if Y is a uniform random variable. By symmetry of the uniform distribution, $1 - Y$ is also a Unif$([0,1])$ if Y is a Unif$([0,1])$. Therefore, setting $U = 1 - Y$, we can say that $-\log(U)/\lambda$ is an exponential random variable with parameter λ.

Problem 4

(a) Run this code block to sample 200 points uniformly at random from the unit square and plot them.

```
using Random; Random.seed!(123)
using Plots
points = [rand(2) for i=1:200]
scatter([x for (x,y) in points],[y for (x,y) in points])
```

(b) Subdivide the square into 100 smaller squares and determine the number of samples contained in each. Store the results in a 10×10 matrix called `countmatrix`. (Hint: you can return the least integer greater than `x` with `ceil(Int,x)`.)

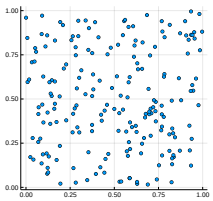
(c) For each k, let $p(k)$ be the proportion of the 100 boxes which contain exactly k samples. Show that p closely matches a Poisson distribution. Use the code below to get started.

```
using StatsBase
sorteddict = sort(countmap(countmatrix[:]))
xs = collect(keys(sorteddict))
ys = collect(values(sorteddict))
sticks(xs,ys/sum(ys),label="tally proportions")
poisson(λ,k) = exp(-λ)*λ^k/factorial(k)
sticks!(xs.+0.1,[poisson(λ,x) for x in xs],label="Poisson(2)")
```

(d) Explain why it's reasonable to expect the proportions to match a Poisson distribution.

Solution

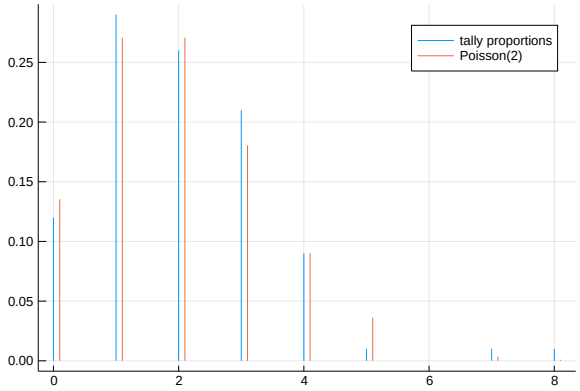
(a) We get points sprinkled randomly throughout the square, as expected:



(b) We loop through the points and increment the appropriate entry in `countmatrix` as we go:

```
countmatrix = zeros(Int,10,10)
for point in points
    countmatrix[ceil(Int,10point[1]),ceil(Int,10point[2])] += 1
end
```

(c) We can experiment with various λ values and find that $\lambda = 2$ is a good fit:



(d) The number of points which fall in the top left square can be written as a sum of indicator random variables: the one which indicates whether the first point falls in the top left square, the one which indicates whether the second point falls in the top left square, etc. Each of these random variables is a Bernoulli random variable with success probability $1/100$. Therefore, the sum of all 200 is Binomial with $n = 100$ and $p = 1/200$. Since the Poisson approximation says that $\text{Bin}(n, \lambda/n) \sim \text{Poiss}(\lambda)$, this suggests that the number of points in the top left square has distribution approximately $\text{Poiss}(2)$ (since $1/100 = 2/200$).

If we let X_i be the number of points in the ith square, then the random variables $X_1, \ldots, X_{100}$ represent 100 *non*-independent samples from a distribution which is approximately $\text{Poiss}(2)$ (since, for example, the conditional distribution of X_{100} given the other 99 X_i's is a point mass at $200 - (X_1 + \cdots + X_{99})$. However, the random variables $X_1, \ldots, X_{100}$ are only weakly dependent, so it's reasonable to expect that the tally count of the 100 results will be approximately reflective of the common distribution.

Problem 5

Let X be the first digit of the number of residents of a randomly selected world city. What would you expect the distribution of X to look like? What about the *last* digit Y?

Load the associated world city populations CSV as a DataFrame and check your predictions. Compare to the distribution with probability mass function

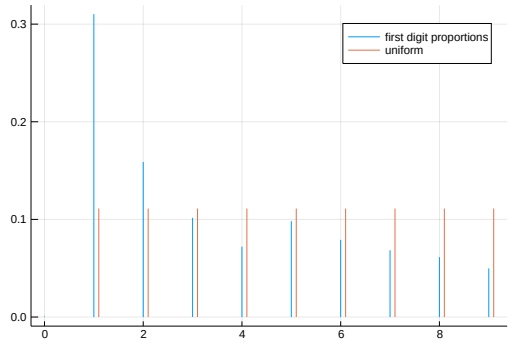
$$m(d) = \log_{10}(d+1) - \log_{10}(d) \quad \text{for } d \in \{1, 2, \ldots, 9\}.$$

```
using StatsBase, Plots, FileIO, DataFrames
D = DataFrame(load("cities.csv"))
D[:Population]
tallydict = # you fill in this part
sticks(1:9,collect(values(tallydict)))
```

Solution

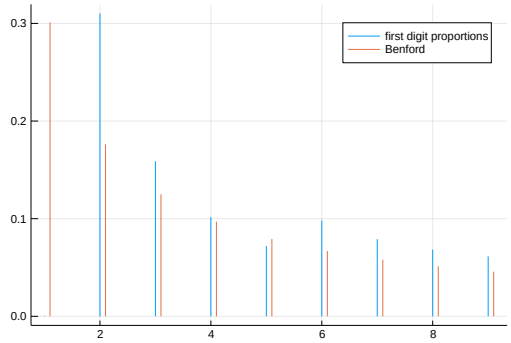
One natural prediction is that both distributions should be uniform (on 1:9 for the first digit, 0:9 for the last). Let's investigate.

```
using StatsBase, Plots, FileIO, DataFrames
D = DataFrame(load("cities.csv"))
D[:Population]
tallydict = sort(countmap([string(n)[1] for n in D[:Population]]))
ys = collect(values(tallydict))
sticks(0:9,ys/sum(ys),label="first digit proportions")
sticks!((1:9).+0.1,[1/9 for d=1:9],label="uniform")
```



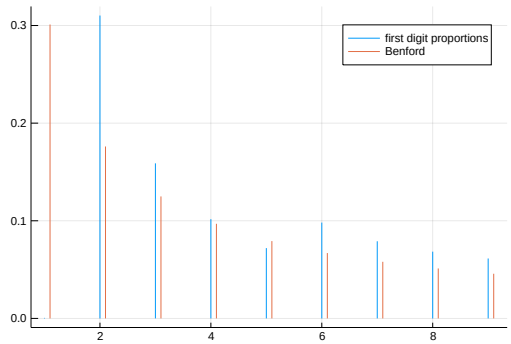
This not seem like a good fit. Let's try the distribution suggested in the problem statement.

```
sticks(1:9,ys/sum(ys),label="first digit proportions")
sticks!((1:9).+0.1,[log10(d+1)-log10(d) for d=1:9],label="Benford")
```



This fit seems better. This distribution is called **Benford's distribution**, and it fits a variety of real-world leading-digit data, for reasons that are not completely well understood (Wikipedia article recommended, for fun).

```
tallydict = sort(countmap([string(n)[end] for n in D[:Population]]))
ys = collect(values(tallydict))
sticks(0:9,ys/sum(ys),label="last digit proportions")
sticks!((0:9).+0.1,[1/10 for d=0:9],label="uniform")
```



We can see that our prediction was pretty accurate for the last digit, except that this data set contains quite a few rounded numbers which causes 0 to be overrepresented.

Problem 6

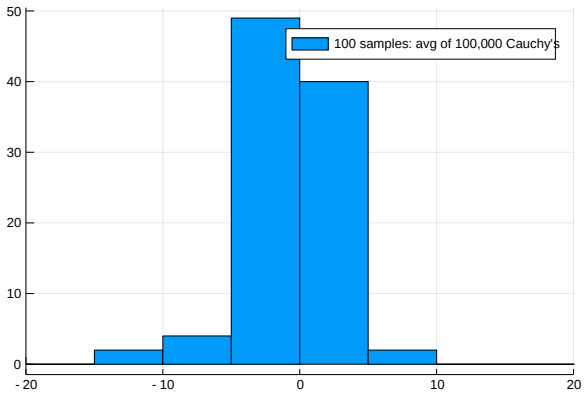
The finite-variance assumption is necessary for the law of large numbers to hold. Repeat the following experiment 100 times: sample from the Cauchy distribution 100,000 times and calculate the sample mean of these samples. Make a histogram of the 100 resulting means. Are these means tightly concentrated?

Note: you can sample from a Cauchy distribution using `tan(π*(rand()-1/2))`. Make a histogram of `samples` with `using Plots; histogram(samples,nbins=20)`.

Solution

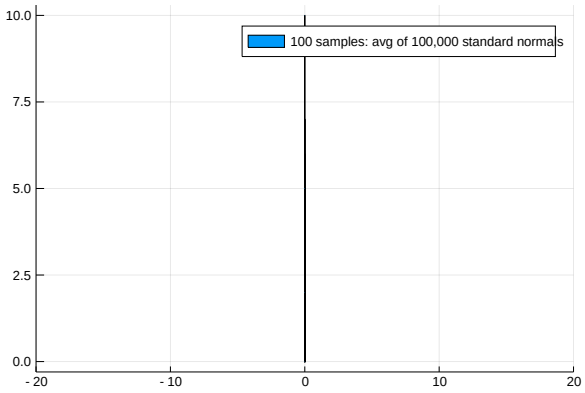
The means are not tightly concentrated. Many samples are fairly close to 0, but many aren't.

```
using Plots
samples = [mean(tan(π*(rand()-1/2)) for i=1:100_000) for j=1:100]
histogram(samples,bins=60,xlims=(-15,25),label="100 samples average of 100,000 Cauchy's")
```



For constrast, let's replace the Cauchy distribution with the normal distribution:

```
samples = [mean(randn() for i=1:100_000) for j=1:100]
histogram(samples,bins=60,xlims=(-15,25),label="100 samples: average of 100,000 standard normals")
```



Problem 7

Suppose that we draw six cards (without replacement) from a standard deck of 52 cards, and that we repeat this experiment n times independently. Does the law of large numbers ensure that the total number of red cards drawn is between $3n - 1000$ and $3n + 1000$ with probability converging to 1 as $n \to \infty$?

Solution

It does not. It implies that, for $\epsilon > 0$, the *average* number of red cards is between $3 - \epsilon$ and $3 + \epsilon$ for large enough n. This implies that the total number of red cards is between $3n - 3n\epsilon$ and $3n + 3n\epsilon$. For large n, $3n\epsilon$ is not less than 1000, so the law of large numbers does not imply anything about the probability that the number of red cards is between $3n - 1000$ and $3n + 1000$.

In fact, the central limit theorem is sharp enough to answer this question. It says that the number of red cards is roughly normally distributed with mean $3n$ and standard deviation given by $\sigma\sqrt{n}$, where σ is the standard deviation of the number of red cards on a single run of the experiment. Therefore, if n is large enough that $\sigma\sqrt{n}$ is much larger than 1000, then the typical fluctuations of the total number of red cards will be much larger than 1000. So the proportion of probability mass in the interval $[3n - 1000, 3n + 1000]$ actually converges to 0, not 1.

Problem 8

In this problem we will perform a computational exploration of the central limit theorem.

(a) Comment on each section of the code printed below. For the two functions which are already commented, explain how they work.

(b) When you run the code, you will find that the graph of the probability mass function of the standardized distribution of the sum of n independent samples from `m` does not line up with the standard normal density. However, there is no mistake in the code. Explain this discrepancy and make a plot for which the two graphs *do* approximately coincide (you could, for example, make suitable edits to the `compareplot` function).

(c) Using your code from (b), examine central limit theorem convergence for the Bernoulli(p) distribution for $p \in \{0.5, 0.75, 0.99\}$. For which value of p is the convergence slowest?

(d) Investigate CLT convergence for the following approximately-Poisson(1) distribution:

```
λ = 1
masses = [exp(-λ)*λ^k/factorial(k) for k=1:8]
masses /= sum(masses)
m = PMF(collect(1:8),masses)
```

Describe qualitatively how the distributions of the standardized sums converge to the normal distribution. (Warning: the recursive convolution function is very computationally expensive, so stick with a dozen or fewer for the second argument.)

Problem 9 code

```
using Statistics, LinearAlgebra
using Plots, StatsBase

struct PMF
    values
    masses
end

import Statistics: mean, var
mean(m::PMF) = m.values ⋅ m.masses
function var(m::PMF)
    μ = mean(m)
    (m.values .- μ).^2 ⋅ m.masses
end

```

```
16  """
17  Return the distribution of the sum of
18  an m-distributed random variable and an
19  independent n-distributed random variable.
20  """
21  function convolve(m::PMF,n::PMF)
22      D = Dict()
23      for (value1,mass1) in zip(m.values,m.masses)
24          for (value2,mass2) in zip(n.values,n.masses)
25              newvalue = value1 + value2
26              if newvalue in keys(D)
27                  D[newvalue] += mass1*mass2
28              else
29                  D[newvalue] = mass1*mass2
30              end
31          end
32      end
33      sorteddict = sort(D)
34      PMF(collect(keys(sorteddict)),collect(values(sorteddict)))
35  end
36
37  """
38  Return the distribution of k independent
39  m-distributed random variables
40  """
41  function convolve(m::PMF,k::Integer)
42      if k == 1
43          m
44      else
45          convolve(convolve(m,k-1),m)
46      end
47  end
48
49  import Plots.plot
50  plot(m::PMF) = sticks(m.values,m.masses)
51  function compareplot(m::PMF)
52      plot(m)
53      xs = range(-4,stop=4,length=1000)
54      ys = [1/sqrt(2π)*exp(-x^2/2) for x in xs]
55      plot!(xs,ys;xlims=(-4,4))
56  end
57
58  function standardize(m::PMF)
59      PMF((m.values .- mean(m))./sqrt(var(m)),m.masses)
60  end
61
62  m = PMF([0,1],[0.5,0.5])
63  compareplot(standardize(convolve(m,8)))
```

Solution

(a) In lines 4-7, we are defining a data type called `PMF`. Objects of this type will represent probability mass functions. In lines 9 through 14, we are defining methods for computing the mean and variance of a probability mass function. The function `convolve` function takes two PMFs and returns the PMF of the sum of independent random variables X and Y with those two PMFs. It does so naively: for each location where the distribution of (X, Y) has positive mass, we identify the corresponding $X + Y$ value and add the appropriate product mass there. The dictionary `D` keeps track of where these masses are.

The second `convolve` method uses recursion to find the distribution of `k` independent samples from `m`.

The `plot` method for the PMF type simply makes a spike graph representing the PMF. The `compareplot` function puts the spike graph alongside a graph of the standard normal density. `standardize` standardizes

the PMF by shifting and scaling to get mean 0 and variance 1. Finally, the last two lines define a fair-coin Bernoulli distribution and make a normal distribution comparison with the standardized distribution of eight independent samples therefrom.

(b) The reason the graphs aren't the same is that they don't represent the same thing: the normal distribution represents probability *density*, while the spike graph represents probability *mass*. To put them on equal footing, we should divide mass by volume (which, since the probability mass is on the number line, means *length*) so that they both represent density. The simplest way to do this is to allocate to each spike the points on the number line which are closest to that spike. To find the length of that interval, we find compute the sequence of gaps between successive spikes and take half of the gap to the left and half of the gap to the right.

```
function compareplot(m::PMF)
    gaps = diff(m.values)
    volumes = [gaps[1]; [0.5(gaps[i]+gaps[i+1]) for i=1:length(gaps)-1]; gaps[end]]
    bar(m.values,m.masses./volumes,bar_width=volumes)
    xs = range(-4,stop=4,length=1000)
    ys = [1/sqrt(2π)*exp(-x^2/2) for x in xs]
    plot!(xs,ys;xlims=(-4,4))
end
```

(c) Convergence is fast for $p = 0.5$ and slow for $p = 0.99$.

```
p = 0.5 # and repeat with the other two values of
m = PMF([0,1],[1-p,p])
compareplot(standardize(convolve(m,12)))
```

Some further commentary: the **Berry-Esseen theorem** quantifies convergence in the central limit theorem: the greatest difference between the standard normal CDF and the CDF of $\frac{X_1+\ldots+X_n-n\mu}{\sigma\sqrt{n}}$ is less than $\rho/(\sigma^3\sqrt{n})$, where $\rho = \mathbb{E}[|X_1|^3]$ and $\sigma^2 = \operatorname{Var} X_1$. This suggests that a small σ value is an impediment to convergence speed, and σ is indeed smaller for $\mathrm{Ber}(0.99)$ than for $\mathrm{Ber}(0.5)$.

(d) The PMF graphs approach the graph of the standard normal density from the left. One way to think of this is that the Poisson distribution is positively skewed, and the skewness takes several convolution iterations to "wear off".