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 $[a, b]$.

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.

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 $F(x) \leq F(y)$ whenever $x \leq y$).

(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.

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.

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)))
```

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)`.

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 $n \to \infty$?

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? (Warning: the recursive convolution function below is very computationally expensive, so stick with a dozen or fewer for the second argument.)

(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.

Problem 8 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

"""
Return the distribution of the sum of
an m-distributed random variable and an
independent n-distributed random variable.
"""
function convolve(m::PMF,n::PMF)
    D = Dict()
    for (value1,mass1) in zip(m.values,m.masses)
        for (value2,mass2) in zip(n.values,n.masses)
            newvalue = value1 + value2
            if newvalue in keys(D)
                D[newvalue] += mass1*mass2
            else
                D[newvalue] = mass1*mass2
            end
        end
    end
    sorteddict = sort(D)
    PMF(collect(keys(sorteddict)),collect(values(sorteddict)))
end

"""
Return the distribution of k independent
m-distributed random variables
"""
function convolve(m::PMF,k::Integer)
    if k == 1
        m
    else
        convolve(convolve(m,k-1),m)
    end
end

import Plots.plot
plot(m::PMF) = sticks(m.values,m.masses)
function compareplot(m::PMF)
    plot(m)
    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

function standardize(m::PMF)
    PMF((m.values .- mean(m))./sqrt(var(m)),m.masses)
end

m = PMF([0,1],[0.5,0.5])
compareplot(standardize(convolve(m,8)))
```