solution2

October 17, 2021

1 Assignment 2 - Solutions

Created: 29 Sep. 2017 Harrison B. Prosper **Updated**: October 2021 for LPC Stats 2021

1.1 Problem 1

Starting from

$$p(t) = \int_{-\infty}^{\infty} \delta(t - xy)\, g(x)\, g(y)\, dx\, dy,$$

where g are normal densities with zero mean and unit variance, show that

$$p(t) = \frac{1}{2} \int_0^1 \frac{\exp(-|t| \,/\, \sin \pi z)}{\sin \pi z}\, dz.$$

Hints: 1. work in polar coordinates; 1. use $\delta(h(u)) = \delta(u - u_0)/|dh/du|$, where $h(u_0) = 0$ and note the restriction that the absolute value imposes.

1.2 Problem 2

Write a short well-commented Python program to complete the calculation of the interval $[\mu_L, \mu_U]$ for the Ebola problem using ROOT but neither RooFit nor RooStats. Feel free to use numpy or scipy.

1.3 Problem 3

Write a short well-commented **Python** program to determine the relative frequency with which statements of the form

$$a \in [x - \sqrt{x}, x + \sqrt{x}],$$

are true in an ensemble of 10,000 experiments, each associated with a *different* mean count a. Assume that each experiment yields a *single* count $x = N$. In the real world, unless the phenomenon being investigated does not exist - in which case the mean count is zero, it is highly unlikely that every experiment in a random collection of experiments will be associated with *exactly* the same mean count. You are to simulate such an ensemble of experiments by sampling their *mean* counts from an exponential distribution,

$$\text{Exp}(a, b) = \exp(-a/b)/b,$$

with mean $b = 5$. Repeat the exercise with $b = 10$.

1.4 Problem 4

Show that

$$\begin{aligned} p(D|s) &= \int_0^\infty p(D|s,b)\,\pi(b|s)\,db, \\ &= \frac{1}{N!M!}e^{-s}\int_0^\infty e^{-(1+\kappa)b}(s+b)^N\,(\kappa b)^M\,db \\ &= \frac{x(1-x)}{M}\sum_{r=0}^{N}\text{beta}(x,r+1,M)\,\text{Poisson}(N-r,s), \\ \text{where } x &= 1/(1+\kappa). \end{aligned}$$

2 Solutions

2.1 Problem 1

The density of a standard Gaussian variate z is $g(z) = \exp(-\frac{1}{2}z^2)\,/\sqrt{2\pi}$, therefore, the density, $p(t)$, can be written as

$$\begin{aligned} p(t) &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\delta(t-x\,y)\,g(x)\,g(y)\,dx\,dy, \\ &= \frac{1}{2\pi}\int_{-\infty}^{\infty}e^{-\frac{1}{2}x^2}\left[\int_{-\infty}^{\infty}\delta(t-x\,y)\,e^{-\frac{1}{2}y^2}\,dy\right]dx. \end{aligned}$$

Transforming

$$p(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\delta(t-x\,y)\,e^{-\frac{1}{2}(x^2+y^2)}\,dx\,dy,$$

to polar coordinates ($x = r\cos\phi, y = r\sin\phi$, $r \in (0,\infty), \phi \in (0,2\pi)$) yields

$$\begin{aligned} p(t) &= \frac{1}{2\pi}\int_0^{2\pi}d\phi\int_0^\infty\delta(t-r^2\sin\phi\cos\phi)\,e^{-\frac{1}{2}r^2}\,r\,dr, \\ &= \frac{1}{2\pi}\int_0^{2\pi}d\phi\int_0^\infty\delta(t-u\sin 2\phi)\,e^{-u}\,du, \text{ where } u = r^2\,/\,2. \end{aligned}$$

Writing $h(u) = t - u\sin 2\phi$ and using $\delta(h(u)) = \delta(u-u_0)\,/\,|dh/du|$, where $u_0 = t\,/\sin 2\phi \geq 0$ is the solution of $h(u) = 0$, we can write $\delta(h) = \delta(u-u_0)/|\sin 2\phi|$. Therefore,

$$p(t) = \frac{1}{2\pi}\int_0^{2\pi}\frac{e^{-t\,/\sin 2\phi}}{\sin 2\phi}\,d\phi,$$

subject to the constraints $\sin 2\phi \geq 0$ and $t\,/\sin 2\phi \geq 0$. These constraints imply that ϕ is restricted to the 1st and 3rd quadrant in the x, y plane (that is, $\phi \in (0,\pi/2) \cup (\pi, 3\pi/2)$) and t must be

replaced by $|t|$. Moreover, since the integral in the 1st and 3rd quadrants have the same value we can limit the integration to the 1st quadrant and multiple the integral by 2. This leads to

$$p(t) = \frac{1}{\pi} \int_0^{\pi/2} \frac{e^{-|t| \,/\, \sin 2\phi}}{\sin 2\phi} \, d\phi,$$

which, with the substitution, $2\phi = \pi z$, yields,

$$p(t) = \frac{1}{2} \int_0^{1} \frac{e^{-|t| \,/\, \sin \pi z}}{\sin \pi z} \, dz.$$

2.2 Problem 2

We shall use the root finder in scipy.

[1]:
```
# standard system modules
import os, sys

# standard array manipulation module
import numpy as np

# standard scientific python module
import scipy as sp
import scipy.optimize as op

import ROOT
```

```
Welcome to JupyROOT 6.24/00
```

[2]:
```
# Number of reported Ebola cases in the USA (2014 - 2016)
N  = 4

# desired confidence level
CL = 0.693
P  = (1 - CL)/2
```

The lower limit μ_L and upper limit μ_U are solutions of the equations

$$D_R(N, \mu_L) = (1 - CL)/2,$$
$$D_L(N, \mu_R) = (1 - CL)/2,$$

where the right and left distribution funcions, D_R and D_L, respectively, are given by

$$D_R(N, \mu) = \sum_{k=N}^{\infty} \text{Poisson}(k, \mu) = P(N, \mu),$$
$$\text{and } D_L(N, \mu) = \sum_{k=0}^{N} \text{Poisson}(k, \mu) = 1 - P(N + 1, \mu).$$

The function $P(\alpha, x) = \int_0^x t^{\alpha-1} \exp(-t)\, dt \,/\, \Gamma(\alpha)$ is the normalized incomplete gamma function.

[3]:
```
# The normalized incomplete gamma function is in the special functions
# module of scipy. The function can also be computed using
# ROOT.TMath.Gamma( , x) in ROOT.

def DR(n, mu):
    return sp.special.gammainc(n, mu)

def DL(n, mu):
    return 1 - sp.special.gammainc(n+1, mu)

# equations to solve
def FR(mu, n):
    return DR(n, mu) - P

def FL(mu, n):
    return DL(n, mu) - P
```

Solve equations using one of the root finders in scipy

[4]:
```
def compute_interval(n):
    dn   = 2*np.sqrt(n)

    # find muL
    amin = max(0, n - dn) # lower bound of range to search for solution
    amax = n              # upper bound of range to search for solution
    muL  = op.brentq(FR, amin, amax, args=(n,))

    # find muU
    amin = n
    amax = n + dn + 5
    muU  = op.brentq(FL, amin, amax, args=(n,))

    return (muL, muU)
```

[5]:
```
muL, muU = compute_interval(N)
print('N = 4, mu in [%3.1f, %3.1f] @ 68%s CL' % (muL, muU, '%'))
```

```
N = 4, mu in [2.1, 7.2] @ 68% CL
```

2.3 Problem 3

Generate 10,000 experiments each with a different mean a. We use **TRandom3()** to generate the sequences of random numbers. * N_{exp} number of experiments * b mean of exponential density Each experiment obtains a count N. This result is translated into the statement

$$a \in [N - \sqrt{N}, N + \sqrt{N}],$$

which is either *true* of *false*, where a is the mean count for that experiment. In the real world we do not know the mean count a (often misleadingly referred to as the *true* count) associated with an experiment. However, in a simulated world we do because we have access to it. Therefore, we can determine whether or not each statement is true or false. In the limit of an infinitely large ensemble of experiments - and therefore statements, the relative frequency with which the statements are true is called the **coverage** probability.

Note: The coverage probability is a property of the ensemble to which the statements belong. It is not a property of the statement except insofar as the statement is presumed to inhabit a particular ensemble. However, if a given statement is imagined to be a member of a different ensemble, then, in general, the coverage probability will change.

The Frequentist Principle The goal of frequentist analyses is to guarantee the following: over an (infinite) ensemble of statements, *which could be about different things*, a minimum fraction, CL, of statements are true. The CL is called the **confidence level**. The task is to invent procedures in which the CL is specified *a priori*. For Gaussian random variables x statements of the form $\mu \in [x - \sigma, x + \sigma]$, where μ is the mean of the Gaussian, are true 68.3% of the time; hence the $p = 0.683$ convention even for non-Gaussian probability functions.

[6]:
```
Nexp = 10000 # number of experiments/statements
ran  = ROOT.TRandom3() # This has a cycle of 2^19937 - 1 ~ 10^6001
```

Model the experiments

In the following, we use Python's **list comprehension** idiom to create one Python list from another. The syntax is

```
alist = [ expression for loop expression involving a list ]
```

We also use the array manipulation capability of numpy.

[7]:
```
def performExperiments(Nexp, b, ran):

    # generate Nexp mean values
    a  = np.array([ran.Exp(b) for _ in range(Nexp)])

    # generate experimental outcome of each experiment
    N  = np.array([ran.Poisson(mean) for mean in a])

    rootN = np.sqrt(N)

    # lower bounds of intervals
    aL = N - rootN

    # upper bounds of intervals
    aU = N + rootN

    return (a, N, aL, aU)
```

Analyze results of experiments

Relative frequency $p = k \,/\, n$ with rough measure of uncertainty $\sqrt{np(1-p)} \,/\, n$.

[8]:
```
def computeCoverage(a, aL, aU):

    # number of experiments
    n = len(a)

    # count number of true statements
    t = (aL <= a) * (a <= aU)

    # compute coverage fraction (i.e., fraction of true statements)
    k  = sum(t)
    p  = float(k)/n

    # since we have k true statements our of n, this is a binomial
    # problem with variance n*p*(1-p). Therefore, a rough estimate
    # of the uncertainty in p is
    dp = np.sqrt(n*p*(1-p))/n

    return (p, dp)
```

Perform experiments and compute coverage of results

[9]:
```
b = 5.0   # mean of exponential density

a, N, aL, aU = performExperiments(Nexp, b, ran)

results = computeCoverage(a, aL, aU)
print("Coverage: %8.3f +/- %-8.3f" % results)
```

```
Coverage:    0.603 +/- 0.005
```

[10]:
```
b = 10.0   # mean of exponential density

a, N, aL, aU = performExperiments(Nexp, b, ran)

results = computeCoverage(a, aL, aU)
print("Coverage: %8.3f +/- %-8.3f" % results)
```

```
Coverage:    0.640 +/- 0.005
```

2.4 Problem 4

$$
\begin{aligned}
p(D|s) &= \int_0^\infty p(D|s,b)\,\pi(b|s)\,db, \\
&= \frac{1}{N!M!}e^{-s}\int_0^\infty e^{-(1+\kappa)b}(s+b)^N\,(\kappa b)^M\,db \\
&= \frac{1}{N!M!}e^{-s}\int_0^\infty e^{-(1+\kappa)b}\left[\sum_{r=0}^{N}\binom{N}{r}s^{N-r}\,b^r\right](\kappa b)^M\,db \\
&= \frac{1}{M!}\sum_{r=0}^{N}e^{-s}\frac{s^{N-r}}{(N-r)!}\frac{1}{r!}\int_0^\infty e^{-(1+\kappa)b}\,b^{M+r}\,\kappa^M\,db \\
&= \sum_{r=0}^{N}\frac{\kappa^M}{(1+\kappa)^{M+r+1}\,M!}\,\text{Poisson}(N-r,s)\frac{1}{r!}\int_0^\infty e^{-(1+\kappa)b}[(1+\kappa)b]^{M+r}\,d(1+\kappa)b \\
&= \sum_{r=0}^{N}\frac{\kappa^M}{(1+\kappa)^{M+r+1}\,M!}\,\frac{\Gamma(M+r+1)}{r!}\,\text{Poisson}(N-r,s),
\end{aligned}
$$

$$
\begin{aligned}
&= \frac{1}{M}\sum_{r=0}^{N}\frac{\kappa^M}{(1+\kappa)^{M+r+1}}\,\frac{\Gamma(M+r+1)}{\Gamma(M)\Gamma(r+1)}\,\text{Poisson}(N-r,s), \\
\text{let } x &= \frac{1}{1+\kappa} \text{ and } 1-x=\frac{\kappa}{1+\kappa}, \text{ then} \\
&= \frac{x(1-x)}{M}\sum_{r=0}^{N}x^{r+1-1}\,(1-x)^{M-1}\,\frac{\Gamma(M+r+1)}{\Gamma(M)\Gamma(r+1)}\,\text{Poisson}(N-r,s), \\
&= \frac{x(1-x)}{M}\sum_{r=0}^{N}\text{beta}(x,r+1,M)\,\text{Poisson}(N-r,s).
\end{aligned}
$$

[ ]: