Statistical Computing 2

숙제 1 수정
2019년 가을학기
응용통계학과 석사과정 최석준

4. Get gamma dist random sample

Feedback: You have to generate samples from Gamma(5, 1)

Parameter 만 마음대로 설정한 줄 알았더니 애초에 알고리즘이 틀렸던 것도 있어서 수정하였습니다. (수정사항은 주석으로 코드상에 기재)

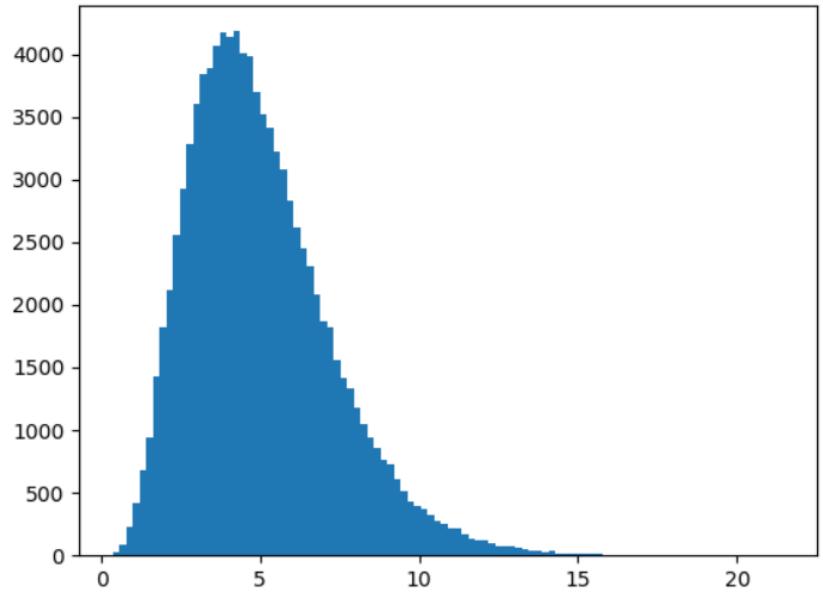
[Python]

```
#python 3 file created by Choi, Seokjun

#get gamma-distributed random samples
#using exponential samples achived by inverse-cdf method

from math import exp, log
from random import uniform, seed

import matplotlib.pyplot as plt



class ExponentialSampler:
    def __init__(self, param_scale):
        self.param_scale = param_scale

    def exponential_sampler(self):
        unif_sample = uniform(0,1)
        return (-self.param_scale*log(unif_sample))

    def get_exponential_sample(self, number_of_smpl):
        result = []
        for _ in range(0, number_of_smpl):
            result.append(self.exponential_sampler())
        return result

class GammaSampler(ExponentialSampler):
    def __init__(self, param_alpha, param_beta):
        if param_alpha%1 != 0:
            raise ValueError("alpha should be integer")
        self.param_alpha = param_alpha
        self.param_scale = param_beta
```

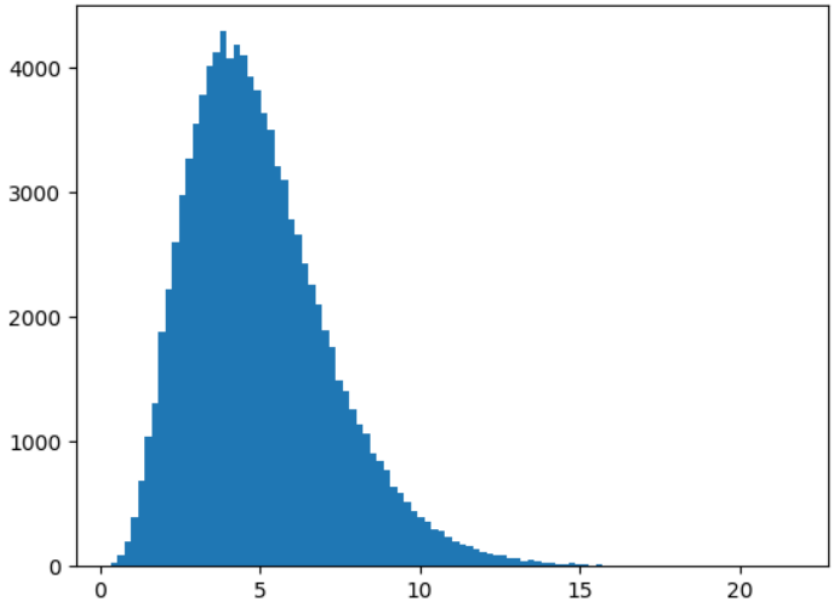
```
    def gamma_sampler(self):
        exp_samples = self.get_exponential_sample(self.param_alpha)
        sum = 0
        for smpl in exp_samples:
            sum += smpl #여기 더했어야함 (기존코드는... 곱해서 log씌웠다.. 그건 unif sample에 해야..)
        return sum

    def get_gamma_sample(self, number_of_smpl):
        result = []
        for _ in range(0, number_of_smpl):
            result.append(self.gamma_sampler())
        return result

class DirectGammaSampler:
    def __init__(self, param_alpha, param_beta):
        if param_alpha%1 != 0:
            raise ValueError("alpha should be integer")
        self.param_alpha = param_alpha
        self.param_beta = param_beta

    def sampler(self):
        product = 1
        for _ in range(self.param_alpha):
            product *= uniform(0,1)
        return -self.param_beta*log(product) #uniform(0,1) sample들을 가지고 바로

    def get_sample(self, number_of_smpl):
        result = []
        for _ in range(0, number_of_smpl):
            result.append(self.sampler())
        return result

if __name__ == "__main__":
    print('run as main')
    # EXPsampler = ExponentialSampler(0.5)
    # print(EXPsampler.get_exponential_sample(10))

    seed(2019-311-252)
    GAMMAsampler = GammaSampler(5,1)
    print(GAMMAsampler.get_gamma_sample(10))

    plt.hist(GAMMAsampler.get_gamma_sample(100000), bins=100)
    plt.show()

    GAMMAsampler2 = DirectGammaSampler(5,1)
    print(GAMMAsampler2.get_sample(10))
    plt.hist(GAMMAsampler2.get_sample(100000), bins=100)
    plt.show()
```

Result:

[Console]
Run as main
[10.216513367270869, 3.3339340391082444, 3.615710979890341, 2.5920351867592313, 3.6536078329650676, 7.6291480731782295, 8.57887847429781, 5.245287216245572, 7.028910570926002, 4.933404613257407]



[Console]
[4.891628333933689, 5.546389126524366, 2.9159249095183797, 3.326348953116164, 6.212441935705352, 2.382918584596532, 6.267177550921478, 6.970221565680502, 4.239179118984349, 5.0619177405465114]



5. Get Poisson parameter's samples using rejection sampling

Feedback: Your ratio cannot protect underflow.

Log 변환하여 처리하도록 수정하였습니다. 이제 sampler 에서의 rejection rule 도 아예 log 씌운 값을 이용합니다.

```
[Python]
#python 3 file created by Choi, Seokjun

#using rejection sampling method,
#sampling poisson's parameter value with lognormal density

from math import log, exp, factorial
from statistics import mean
import random

import matplotlib.pyplot as plt


class Lognormal_Poisson_RejectionSampler:
    def __init__(self, data):
        self.data = data
        self.data_mean = mean(data)

    def pois_pmf(self, x, param_lambda):
        if not isinstance(x, int):
            raise ValueError("x should be integer.")
        return ((param_lambda**x)*exp(-param_lambda)/factorial(x))

    def log_thres_p_calculator(self, lognorm_sample):
        thres_p_upper = (self.pois_pmf(x, lognorm_sample) for x in self.data)
        thres_p_lower = (self.pois_pmf(x, self.data_mean) for x in self.data)
        log_thres_p = 0
        for up, low in zip(thres_p_upper, thres_p_lower):
            log_thres_p = log_thres_p + log(up) – log(low)
            #~수업 note~
            #이거 그냥계산하면 잘못하면 underflow 남
            #log씌워서 sum으로 계산하고 다시 변환해오자 (나중에 해볼것)
            #아니면 통째로 알고리즘을 다 log버전으로 돌리자(sampler를 수정. Uniform sample에 log씌우고 rejection
rule)
            #(수정함: rejection rule까지 log버전으로 고침.)
        return log_thres_p

    def sampler(self):
        #get one sample
        while(1):
            unif_sample = random.uniform(0,1)
            lognorm_sample = exp(random.normalvariate(log(4), 0.5))
```

```
                log_thres_p = self.log_thres_p_calculator(lognorm_sample)

                # print('p: ', thres_p ," and now uniform r.s : ", unif_sample)
                if log(unif_sample) < log_thres_p: # log값 이용
                    # print('accepted: ', lognorm_sample)
                    yield lognorm_sample
                else:
                    # print('rejected')
                    pass

    def get_sample(self, number_of_smpl):
        result = []
        for _ in range(0, number_of_smpl):
            result.append(next(self.sampler()))
        return result

if __name__ == "__main__":
    print('run as main')
    random.seed(2019-311-252)
    given_data = (8,3,4,3,1,7,2,6,2,7)
    LPsampler = Lognormal_Poisson_RejectionSampler(given_data)
    print(LPsampler.get_sample(10))
    plt.hist(LPsampler.get_sample(100000), bins=100)
    plt.show()
```

Result:

```
[Console]
run as main
[3.9497928775588886, 4.917777219059587, 3.854358211869205, 3.9719816335092757, 4.482831620647739,
3.9060658551095644, 4.584039558823791, 4.802052259589951, 3.691898778304127, 3.6294331768172485]
```