HW6

Geoffrey Woollard

My code lives in the repo https://github.com/geoffwoollard/prob_prog

Acknowledgments

- discussions with Jordan Lovrod, Ilias Karimalis, Gaurav Bhatt
- starter code for `smc.resample_particles` and `smc.SMC` from Masoud Mokhtari

In [1]:
```
from dill.source import getsource, getsourcelines
```

In [2]:
```
from smc import resample_particles, SMC

list_of_programs = [resample_particles, SMC]

for program in list_of_programs:
    for line_number, function_line in enumerate(getsourcelines(program)[0]):
        print(line_number, function_line,end='')
    print()
```

```
0 def resample_particles(particles, log_weights):
1     """
2     Eq. 4.24 in course textbook (https://arxiv.org/abs/1809.10756v2, pp. 122)
3     See Algorithm 15 in the course textbook, section 6.7 Sequantial Monte Carlo, p. 176
4     """
5     log_weights = tensor(log_weights)
6     n_particles = log_weights.size().numel()
7 
8     unnormalized_particle_weights = torch.exp(log_weights).detach().numpy()
9 
10     particle_idxs = np.random.choice(
11         a=range(n_particles),
12         size=n_particles,
13         p=unnormalized_particle_weights/unnormalized_particle_weights.sum(),
14         replace=True,
15         )
16     #print('particle_idxs',particle_idxs)
17 
18     new_particles = []
19     for idx in range(n_particles):
20         new_particles.append(particles[particle_idxs[idx]]) # TODO: copy?
21 
22     log_Z = np.log(np.sum(unnormalized_particle_weights)/n_particles)
23     return log_Z, new_particles

0 def SMC(n_particles, exp,do_log=False):
1 
2     particles = []
3     weights = []
4     logZs = []
5     output = lambda x: x
6 
7     for i in range(n_particles):
8 
9         res = evaluate(exp, env=None)('addr_start', output)
10         logW = 0.
11 
12 
13         particles.append(res)
14         weights.append(logW)
15 
16     #can't be done after the first step, under the address transform, so this should be fine:
17     done = False
18     smc_cnter = 0
19     while not done:
20         if do_log: print('In SMC step {}, Zs: '.format(smc_cnter), logZs)
21         for i in range(n_particles): #Even though this can be parallelized, we run it serially
22             res = run_until_observe_or_end(particles[i]) # particle i at next breakbpoint
23             if 'done' in res[2]: #this checks if the calculation is done
24                 particles[i] = res[0]
25                 if i == 0:
26                     done = True  #and enforces everything to be the same as the first particle
27                     address = ''
28                 else:
29                     if not done: # triggered when i=0 was not done and i>0 was done
30                         raise RuntimeError('Failed SMC, finished one calculation before the other')
31             else:
32                 #TODO: check particle addresses, and get weights and continuations
33                 particles[i] = res
34                 cont, args, sigma = res
35                 assert 'observe' == sigma['type']
36                 weights[i] = sigma['distribution'].log_prob(sigma['observed_constant'])
37 
38                  # check particle addresses
39                 if i == 0:
40                     break_point_address = sigma['address']
41                 else:
42                     if sigma['address'] != break_point_address:
43                         assert False, 'particles at different break points'
44 
45 
46 
47         if not done:
48             #resample and keep track of logZs
49             logZn, particles = resample_particles(particles, weights)
50             logZs.append(logZn)
51         smc_cnter += 1
52     logZ = sum(logZs)
53     return logZ, particles
```

1.daphne

In [2]:
```
from daphne import daphne
import os, json
import numpy as np
import torch
from torch import tensor
import pandas as pd
import matplotlib.pyplot as plt
```

In [3]:
```
def ast_helper(fname,directory):
    sugared_fname = '../prob_prog/hw/hw6/CS532-HW6/{}/{}'.format(directory,fname)
    desugared_ast_json_fname = '/Users/gw/repos/prob_prog/' + sugared_fname.replace('.daphne','.json')
    if os.path.isfile(desugared_ast_json_fname):
        with open(desugared_ast_json_fname) as f:
            ast = json.load(f)
    else:
        #note: the sugared path that goes into daphne desugar should be with respect to the daphne path!
        ast = daphne(['desugar-hoppl-cps', '-i', sugared_fname])

        with open(desugared_ast_json_fname, 'w') as f:
            json.dump(ast, f)
    return ast

i=1
fname = '{}.daphne'.format(i)
exp = ast_helper(fname,directory='programs')
%cat programs/1.daphne
```

```
(defn until-success [p n]
   (if (sample (flip p))
     n
     (until-success p (+ n 1))))

(let [p 0.01]
  (until-success p 0))
```

In [8]:
```
import smc, evaluator
import importlib
importlib.reload(smc)
```

Out[8]:
```
<module 'smc' from '/Users/gw/repos/prob_prog/hw/hw6/CS532-HW6/smc.py'>
```

In [11]:
```
output = lambda x: x
evaluator.evaluate(exp, env=None)('addr_start', output)
```

Out[11]:
```
(<function primitives.push_addr(alpha, value, k)>,
 ['addr_start', '0', <evaluator.Procedure at 0x135e73640>],
 {'type': 'proc'})
```

In [14]:
```
n_particles=3
logZ, particles = smc.SMC(n_particles, exp)
particles
```

Out[14]:
```
In SMC step 0, Zs:  []
[tensor(42), tensor(122), tensor(88)]
```

Note that there are no observed in this program, and thus Z is undefined

In [15]:
```
particle_counts = [1,10,100,1000,10000,100000]
fig, axes = plt.subplots(nrows=len(particle_counts),figsize=(30,20))
# fig.tight_layout()
plt.subplots_adjust(left=None, bottom=None, right=None, top=None, wspace=None, hspace=0.5) # https://stackoveri


for idx, n_particles in enumerate(particle_counts):
    logZ, particles = smc.SMC(n_particles, exp)
    samples_array = np.array([sample.item() for sample in particles])
    mean = samples_array.mean()
    var = samples_array.var()
    pd.Series(samples_array).plot.hist(ax=axes[idx], bins=50, title='Program {} | {} particles | mean {:1.3f} |
```

```
In SMC step 0, Zs:  []
In SMC step 0, Zs:  []
In SMC step 0, Zs:  []
In SMC step 0, Zs:  []
In SMC step 0, Zs:  []
In SMC step 0, Zs:  []
```

2.daphne

In [2]:
```
from daphne import daphne
import os, json
import numpy as np
import torch
from torch import tensor
import pandas as pd
import matplotlib.pyplot as plt
```

In [3]:
```
def ast_helper(fname,directory):
    sugared_fname = '../prob_prog/hw/hw6/CS532-HW6/{}/{}'.format(directory,fname)
    desugared_ast_json_fname = '/Users/gw/repos/prob_prog/' + sugared_fname.replace('.daphne','.json')
    if os.path.isfile(desugared_ast_json_fname):
        with open(desugared_ast_json_fname) as f:
            ast = json.load(f)
    else:
        #note: the sugared path that goes into daphne desugar should be with respect to the daphne path!
        ast = daphne(['desugar-hoppl-cps', '-i', sugared_fname])

        with open(desugared_ast_json_fname, 'w') as f:
            json.dump(ast, f)
    return ast

i=2
fname = '{}.daphne'.format(i)
exp = ast_helper(fname,directory='programs')
%cat programs/2.daphne
```

```
(defn marsaglia-normal [mean var]
   (let [d (uniform-continuous -1.0 1.0)
         x (sample d)
         y (sample d)
         s (+ (* x x ) (* y y ))]
    (if (< s 1)
        (+ mean (* (sqrt var)
                   (* x (sqrt (* -2 (/ ( log s) s))))))
        (marsaglia-normal mean var))))

(let [mu (marsaglia-normal 1 5)
      sigma (sqrt 2)
      lik (normal mu sigma)]
  (observe lik 8)
  (observe lik 9)
  mu)
```

In [4]:
```
import smc
import importlib
importlib.reload(smc)
```

Out[4]:
```
<module 'smc' from '/Users/gw/repos/prob_prog/hw/hw6/CS532-HW6/smc.py'>
```

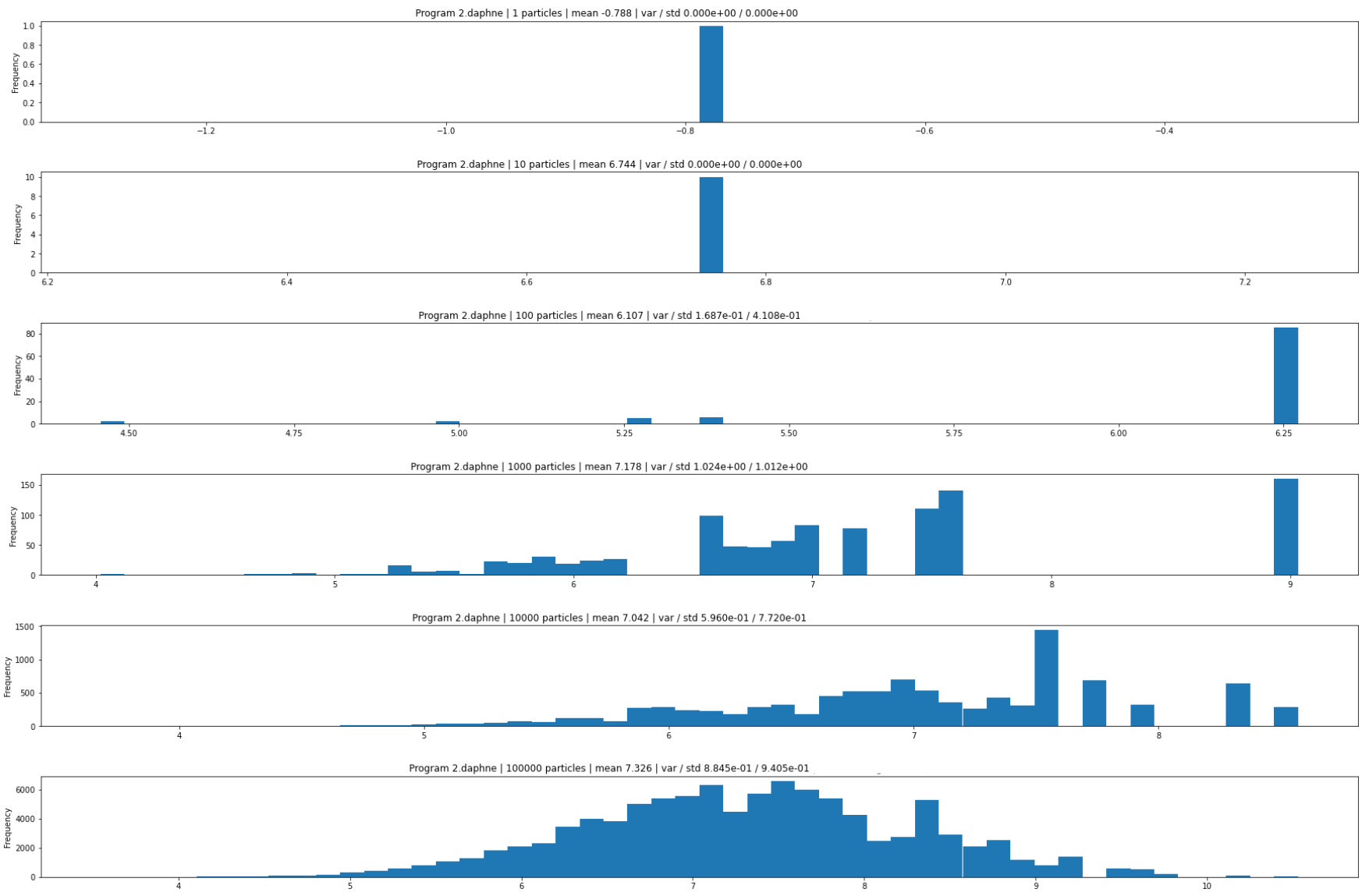
In [5]:
```
n_particles=3
logZ, particles = smc.SMC(n_particles, exp)
```

```
In SMC step 0, Zs:  []
In SMC step 1, Zs:  [-8.746667201168327]
In SMC step 2, Zs:  [-8.746667201168327, -11.342506857689752]
```

In [7]:
```
particle_counts = [1,10,100,1000,10000,100000]
fig, axes = plt.subplots(nrows=len(particle_counts),figsize=(30,20))
# fig.tight_layout()
plt.subplots_adjust(left=None, bottom=None, right=None, top=None, wspace=None, hspace=0.5) # https://stackoverf


for idx, n_particles in enumerate(particle_counts):
    logZ, particles = smc.SMC(n_particles, exp)
    samples_array = np.array([sample.item() for sample in particles])
    mean = samples_array.mean()
    var = samples_array.var()
    pd.Series(samples_array).plot.hist(ax=axes[idx], bins=50, title='Program {} | {} particles | mean {:1.3f} |
```

```
In SMC step 0, Zs:  []
In SMC step 1, Zs:  [-20.574844409164438]
In SMC step 2, Zs:  [-20.574844409164438, -25.219081845127047]
In SMC step 0, Zs:  []
In SMC step 1, Zs:  [-3.8916753382007996]
In SMC step 2, Zs:  [-3.8916753382007996, -2.641245161237912]
In SMC step 0, Zs:  []
In SMC step 1, Zs:  [-5.708884842017928]
In SMC step 2, Zs:  [-5.708884842017928, -3.9738090087117763]
In SMC step 0, Zs:  []
In SMC step 1, Zs:  [-5.07073953887065]
In SMC step 2, Zs:  [-5.07073953887065, -2.9832770513027356]
In SMC step 0, Zs:  []
In SMC step 1, Zs:  [-5.4072599330000415]
In SMC step 2, Zs:  [-5.4072599330000415, -2.9642962996086535]
In SMC step 0, Zs:  []
In SMC step 1, Zs:  [-5.382312301483098]
In SMC step 2, Zs:  [-5.382312301483098, -2.790267592243308]
```



3.daphne

In [20]:
```
from daphne import daphne
import os, json
import numpy as np
import torch
from torch import tensor
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
def ast_helper(fname,directory):
    sugared_fname = '../prob_prog/hw/hw6/CS532-HW6/{}/{}'.format(directory,fname)
    desugared_ast_json_fname = '/Users/gw/repos/prob_prog/' + sugared_fname.replace('.daphne','.json')
    if os.path.isfile(desugared_ast_json_fname):
        with open(desugared_ast_json_fname) as f:
            ast = json.load(f)
    else:
        #note: the sugared path that goes into daphne desugar should be with respect to the daphne path!
        ast = daphne(['desugar-hoppl-cps', '-i', sugared_fname])

        with open(desugared_ast_json_fname, 'w') as f:
            json.dump(ast, f)
    return ast

i=3
fname = '{}.daphne'.format(i)
exp = ast_helper(fname,directory='programs')
%cat programs/3.daphne
```

```
(defn reduce [f x values]
               (if (empty? values)
                  x
                  (reduce f (f x (first values)) (rest values))))

(let [observations [0.9 0.8 0.7 0.0 -0.025 -5.0 -2.0 -0.1 0.0 0.13 0.45 6 0.2 0.3 -1 -1]
      init-dist (discrete [1.0 1.0 1.0])
      trans-dists {0 (discrete [0.1 0.5 0.4])
                   1 (discrete [0.2 0.2 0.6])
                   2 (discrete [0.15 0.15 0.7])}
      obs-dists {0 (normal -1 1)
                 1 (normal 1 1)
                 2 (normal 0 1)}]
      (reduce
        (fn [states obs]
          (let [state (sample (get trans-dists
                                   (peek states)))]
            (observe (get obs-dists state) obs)
            (conj states state)))
        [(sample init-dist)]
        observations))
```

In [64]:
```
import smc, evaluator
import importlib
importlib.reload(smc)
```

Out[64]:
```
<module 'smc' from '/Users/gw/repos/prob_prog/hw/hw6/CS532-HW6/smc.py'>
```

In [75]:
```
%%time
n_particles=100
logZ, particles = smc.SMC(n_particles, exp)
samples_array = torch.stack(particles).detach().numpy()
samples_array.mean(0)
```

```
CPU times: user 2.97 s, sys: 10.2 ms, total: 2.98 s
Wall time: 2.98 s
```

Out[75]:
```
array([ 1.420,  1.600,  1.470,  1.480,  1.030,  1.540,  1.930,  1.930,
        1.730,  0.940,  0.260,  1.750,  1.670,  1.930,  2.000,  2.000,
        0.940])
```

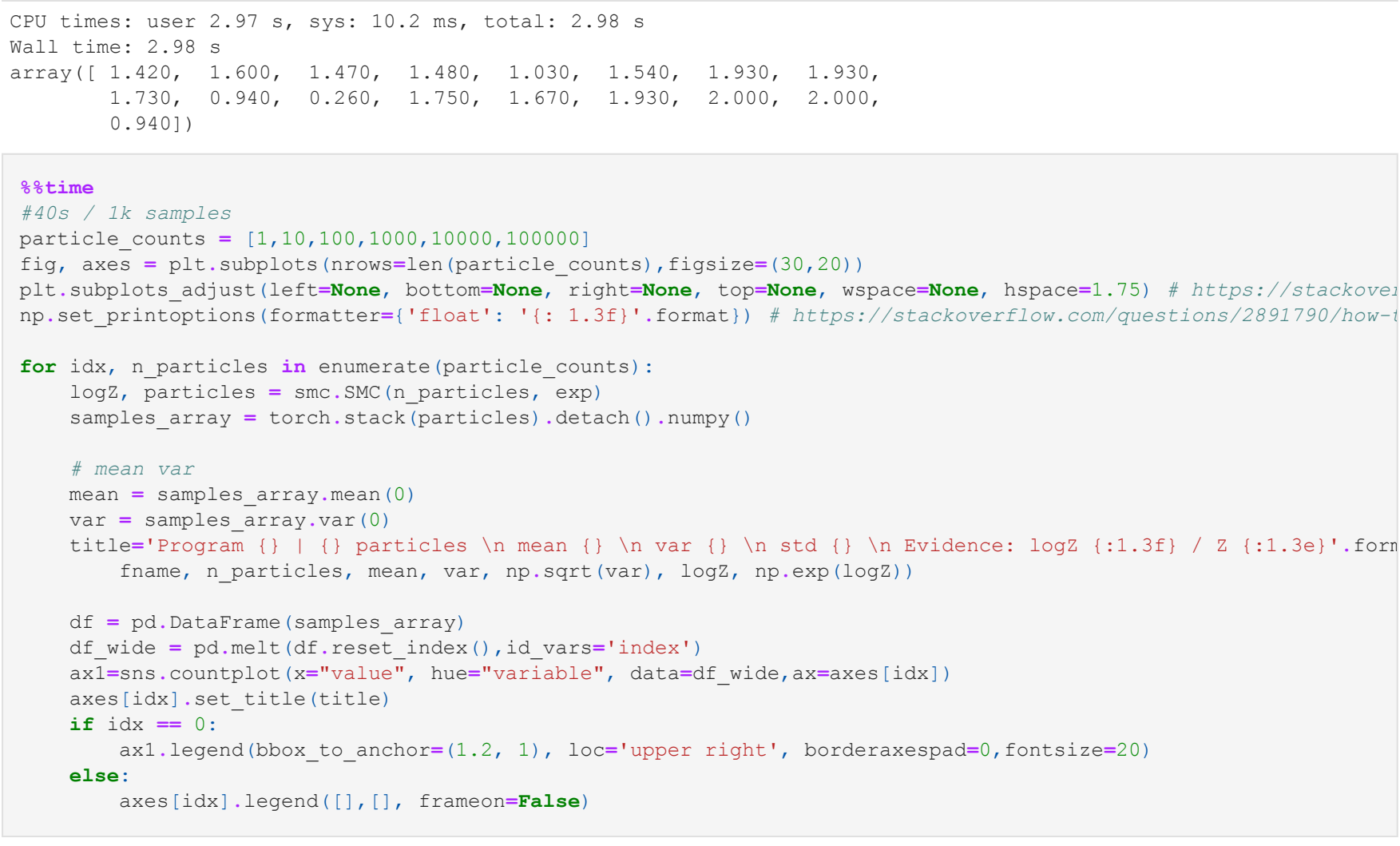
In [76]:
```
%%time
#40s / 1k samples
particle_counts = [1,10,100,1000,10000,100000]
fig, axes = plt.subplots(nrows=len(particle_counts),figsize=(30,20))
plt.subplots_adjust(left=None, bottom=None, right=None, top=None, wspace=None, hspace=1.75) # https://stackover
np.set_printoptions(formatter={'float': '{: 1.3f}'.format}) # https://stackoverflow.com/questions/2891790/how-t

for idx, n_particles in enumerate(particle_counts):
    logZ, particles = smc.SMC(n_particles, exp)
    samples_array = torch.stack(particles).detach().numpy()

    # mean var
    mean = samples_array.mean(0)
    var = samples_array.var(0)
    title='Program {} | {} particles \n mean {} \n var {} \n std {} \n Evidence: logZ {:1.3f} / Z {:1.3e}'.form
        fname, n_particles, mean, var, np.sqrt(var), logZ, np.exp(logZ))

    df = pd.DataFrame(samples_array)
    df_wide = pd.melt(df.reset_index(),id_vars='index')
    ax1=sns.countplot(x="value", hue="variable", data=df_wide,ax=axes[idx])
    axes[idx].set_title(title)
    if idx == 0:
        ax1.legend(bbox_to_anchor=(1.2, 1), loc='upper right', borderaxespad=0,fontsize=20)
    else:
        axes[idx].legend([],[], frameon=False)
```

```
CPU times: user 1h 4min 59s, sys: 18 s, total: 1h 5min 17s
Wall time: 1h 5min 26s
```



4.daphne

In [11]:
```
from daphne import daphne
import os, json
import numpy as np
import torch
from torch import tensor
import pandas as pd
import matplotlib.pyplot as plt
```

In [34]:
```
def ast_helper(fname,directory):
    sugared_fname = '../prob_prog/hw/hw6/CS532-HW6/{}/{}'.format(directory,fname)
    desugared_ast_json_fname = '/Users/gw/repos/prob_prog/' + sugared_fname.replace('.daphne','.json')
    if os.path.isfile(desugared_ast_json_fname):
        with open(desugared_ast_json_fname) as f:
            ast = json.load(f)
    else:
        #note: the sugared path that goes into daphne desugar should be with respect to the daphne path!
        ast = daphne(['desugar-hoppl-cps', '-i', sugared_fname])

        with open(desugared_ast_json_fname, 'w') as f:
            json.dump(ast, f)
    return ast

i=4
fname = '{}.daphne'.format(i)
exp = ast_helper(fname,directory='programs')
%cat programs/4.daphne
```

```
(let [mu (sample (normal 1 (sqrt 5)))
          sigma (sqrt 2)
          lik (normal mu sigma)]
      (observe lik 8)
      (observe lik 9)
      mu)
```

In [35]:
```
import smc
import importlib
importlib.reload(smc)
```

Out[35]:
```
<module 'smc' from '/Users/gw/repos/prob_prog/hw/hw6/CS532-HW6/smc.py'>
```

In [45]:
```
particle_counts = [1,10,100,1000,10000,100000]
fig, axes = plt.subplots(nrows=len(particle_counts),figsize=(30,20))
# fig.tight_layout()
plt.subplots_adjust(left=None, bottom=None, right=None, top=None, wspace=None, hspace=0.5) # https://stackoveri


for idx, n_particles in enumerate(particle_counts):
    logZ, particles = smc.SMC(n_particles, exp)
    samples_array = np.array([sample.item() for sample in particles])
    mean = samples_array.mean()
    var = samples_array.var()
    pd.Series(samples_array).plot.hist(ax=axes[idx], bins=50, title='Program {} | {} particles | mean {:1.3f} |
```

```
In SMC step 0, Zs:  []
In SMC step 1, Zs:  [-5.426987590543291]
In SMC step 2, Zs:  [-5.426987590543291, -7.7169570978697495]
In SMC step 0, Zs:  []
In SMC step 1, Zs:  [-4.409021968208568]
In SMC step 2, Zs:  [-4.409021968208568, -3.281747838494585]
In SMC step 0, Zs:  []
In SMC step 1, Zs:  [-5.44325713520603]
In SMC step 2, Zs:  [-5.44325713520603, -2.7813596798573204]
In SMC step 0, Zs:  []
In SMC step 1, Zs:  [-5.348063793569588]
In SMC step 2, Zs:  [-5.348063793569588, -3.34402504427412]
In SMC step 0, Zs:  []
In SMC step 1, Zs:  [-5.384503472706044]
In SMC step 2, Zs:  [-5.384503472706044, -2.8596724029796277]
In SMC step 0, Zs:  []
In SMC step 1, Zs:  [-5.406247981671558]
In SMC step 2, Zs:  [-5.406247981671558, -2.857562885985456]
```