## 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)
                                                                          (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)
                         <module 'smc' from '/Users/gw/repos/prob prog/hw/hw6/CS532-HW6/smc.py'>
Out[64]:
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
                         array([ 1.420, 1.600, 1.470, 1.480, 1.030, 1.540, 1.930, 1.930,
                                               1.730,
                                                                                        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://stackovel
                            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 <math>\{\}\ n \ title= \ n \
                                                 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
                                                                                                                         Program 3 daphne | 1 particles
mean [ 2.000 1.000 0.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.00
                                                                                                                                                                                                                                                                                                                        0
                                                                                                                                                                                                                                                                                                                        1
                                                                                                                                                                                                                                                                                                                         2
                                                                                                                         3
                                                                                                                                                                                                                                                                                                                        4
                                                                                                                                                                                                                                                                                                                        ____5
                                                                                                                                                                                                                                                                                                                        6
                                                                                                                                                                                                                                                                                                                        7
                                                                                                                                                                                                                                                                                                                        8
                                                                                                                         Program 3 daphre | 100 particles
mean [ 1.410 1.570 1.720 1.810 1.600 1.160 1.780 1.610 1.740 1.140
0.300 1.690 1.550 1.920 1.520 1.600 1.550]
var [ 0.802 0.645 0.242 0.154 0.000 0.694 0.232 0.538 0.252 0.980
0.510 0.214 0.559 0.074 0.730 0.560 0.247]
std [ 0.895 0.893 0.492 0.392 0.000 0.833 0.481 0.733 0.502 0.990
0.714 0.462 0.754 0.271 0.834 0.748 0.497]
Evidence: logz -44.510 / Z 4.675e-20
                                                                                                                                                                                                                                                                                                                        9
                                                                                                                                                                                                                                                                                                                        10
                                                                                                                                                                                                                                                                                                                        15
                                                                                                                         mean [ 1.355 1.636 1.729 1.590 1.625 1.481 1.557 1.635 1.610 1.074  
Var[ 0.817 0.54 1.682 1.666 1.732 1.671 1.646 0.947]  
Var[ 0.817 0.55 1.686 1.967 1.732 1.671 1.646 0.947]  
Var[ 0.817 0.55 1.686 0.999 0.91 0.232 0.955 0.935 0.622]  
Std [ 0.994 0.749 0.533 0.693 0.156 0.748 0.698 0.662 0.586 0.974  
0.535 0.623 0.625 0.482 0.534 0.550 0.789]  
Evidence: logZ -44.174 / Z 6.538e-20
                                                                                                                         Program 3 daphne | 10000 particles
mean [ 1.417 1.544 1.709 1.613 \( \)_02/0 1.442 1.675 1.668 1.619 1.063
\( 0.172 1.712 1.682 1.754 1.627 1.526 0.888 \)
var[ 0.763 0.657 0.313 0.455 0.020 0.613 0.406 0.421 0.422 0.958
\( 0.314 0.330 0.447 0.242 0.344 0.322 0.701 \)
std [ 0.873 0.810 0.559 0.674 0.141 0.783 0.637 0.649 0.650 0.979
\( 0.560 0.574 0.638 0.492 0.583 0.567 0.837 ) \)
Evidence: logZ -44.442 / Z 5.004e-20
                                                                                                                         Program 3 daphnej 100000 particles
mean [1.424 1.549 1.708 1.611 3,004 1.421 1.651 1.683 1.593 1.030
0.133 1.677 1.673 1.687 1.636 1.517 0.938]
var[0.759 0.651 0.316 0.459 0.016 0.602 0.423 0.403 0.432 0.954
0.248 0.371 0.430 0.302 0.345 0.328 0.690]
std[0.871 0.807 0.552 0.677 0.152 0.776 0.651 0.635 0.657 0.977
0.498 0.609 0.656 0.549 0.587 0.373 0.8311
Evidence: 0.022 -44.426/25 0.084e-20
```