In [1]:
```python
import torch
import numpy as np
import os, json
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

from evaluation_based_sampling import evaluate, evaluate_program
from daphne import daphne

from graph_based_sampling import sample_from_joint
```

Problem 3

In [2]:
```python
from load_helper import ast_helper, graph_helper
```

Importance sampling

- 10k samples in 1.59s implies 384k samples in 10 min
- 1k samples in 5.29s implies 113k samples in 10 min

In [3]:
```python
import parse
import importance_sampling
```

In [4]:
```python
fname = '3.daphne'
ast = ast_helper(fname)
ast
```

Out[4]:
```
[['let',
  ['data', ['vector', 1.1, 2.1, 2.0, 1.9, 0.0, -0.1, -0.05]],
  ['let',
   ['likes',
    ['vector',
     ['let',
      ['mu', ['sample', ['normal', 0.0, 10.0]]],
      ['let',
       ['sigma', ['sample', ['gamma', 1.0, 1.0]]],
       ['normal', 'mu', 'sigma']]],
     ['let',
      ['mu', ['sample', ['normal', 0.0, 10.0]]],
      ['let',
       ['sigma', ['sample', ['gamma', 1.0, 1.0]]],
       ['normal', 'mu', 'sigma']]],
     ['let',
      ['mu', ['sample', ['normal', 0.0, 10.0]]],
      ['let',
       ['sigma', ['sample', ['gamma', 1.0, 1.0]]],
       ['normal', 'mu', 'sigma']]]]],
   ['let',
    ['pi', ['sample', ['dirichlet', ['vector', 1.0, 1.0, 1.0]]]],
    ['let',
     ['z-prior', ['discrete', 'pi']],
     ['let',
      ['z',
       ['vector',
        ['let',
         ['y', ['get', 'data', 0]],
         ['let',
          ['z', ['sample', 'z-prior']],
          ['let', ['_', ['observe', ['get', 'likes', 'z'], 'y']], 'z']]],
        ['let',
         ['y', ['get', 'data', 1]],
         ['let',
          ['z', ['sample', 'z-prior']],
          ['let', ['_', ['observe', ['get', 'likes', 'z'], 'y']], 'z']]],
        ['let',
         ['y', ['get', 'data', 2]],
         ['let',
          ['z', ['sample', 'z-prior']],
          ['let', ['_', ['observe', ['get', 'likes', 'z'], 'y']], 'z']]],
        ['let',
         ['y', ['get', 'data', 3]],
         ['let',
          ['z', ['sample', 'z-prior']],
          ['let', ['_', ['observe', ['get', 'likes', 'z'], 'y']], 'z']]],
        ['let',
         ['y', ['get', 'data', 4]],
         ['let',
          ['z', ['sample', 'z-prior']],
          ['let', ['_', ['observe', ['get', 'likes', 'z'], 'y']], 'z']]],
        ['let',
         ['y', ['get', 'data', 5]],
         ['let',
          ['z', ['sample', 'z-prior']],
          ['let', ['_', ['observe', ['get', 'likes', 'z'], 'y']], 'z']]],
        ['let',
         ['y', ['get', 'data', 6]],
         ['let',
          ['z', ['sample', 'z-prior']],
          ['let', ['_', ['observe', ['get', 'likes', 'z'], 'y']], 'z']]]]],
      ['=', ['first', 'z'], ['second', 'z']]]]]]]]
```

In [36]:
```python
%%time
num_samples=113000
samples, sigmas = parse.take_samples(num_samples,ast=ast)
```

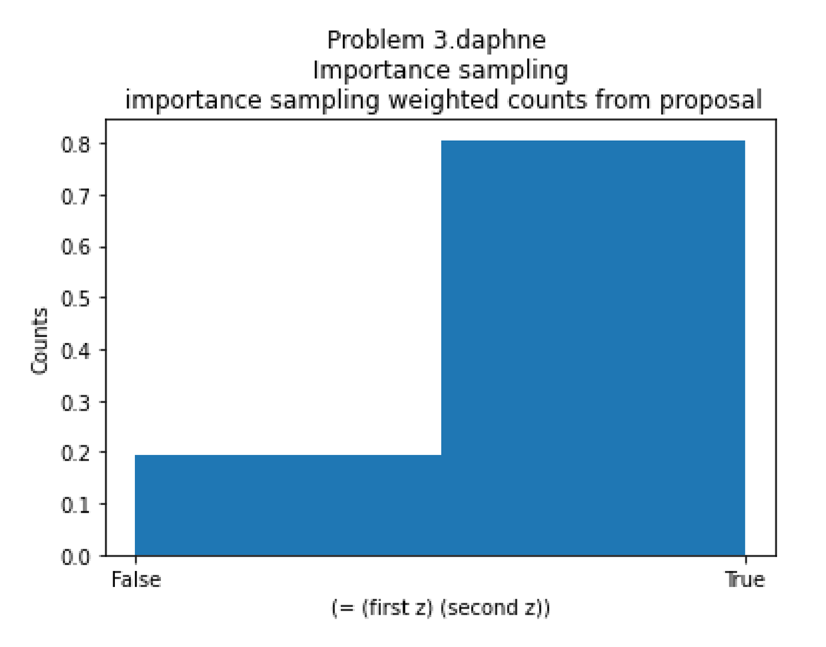
```
CPU times: user 10min 7s, sys: 1.91 s, total: 10min 9s
Wall time: 10min 11s
```

In [37]:
```python
samples = np.array([sample.item() for sample in samples])
```

In [38]:
```python
posterior_mean, probs = importance_sampling.weighted_average(samples,sigmas)
```

In [39]:
```python
_ = plt.hist(samples.astype(int), weights=probs, bins=2)
plt.xlabel('(= (first z) (second z))')
plt.title('Problem {} \n Importance sampling \n importance sampling weighted counts from proposal'.format(fname
plt.xticks([0,1],["False","True"])
plt.ylabel('Counts')
```

Out[39]:
```
Text(0, 0.5, 'Counts')
```



In [40]:
```python
method = "Importance sampling"

""" {}: The posterior probability that the first and second datapoint are in the same cluster,\
i.e. the posterior probability that z[1] == z[2]: {:0.3f}\
""".format(method,posterior_mean)
```

Out[40]:
```
' Importance sampling: The posterior probability that the first and second datapoint are in the same cluster,i.
e. the posterior probability that z[1] == z[2]: 0.806'
```

MH Gibbs

0.27 s / sample implies ~2.2k samples in 10 min 0.1k in 30.3s implies 1.98k in 10 min

My code is slow, perhaps because of some internal logging and so the burn in needs to be quite large compared with the total run. But the relatively long burn in is justified by the joint increasing.

In [10]:
```python
import graph_based_sampling
import mh_gibbs
from hmc import compute_log_joint_prob
```

In [11]:
```python
fname = '3.daphne'
graph = graph_helper(fname)
graph
```

Out[11]:
```
[{},
 {'V': ['sample5',
   'sample0',
   'observe18',
   'observe12',
   'sample17',
   'sample7',
   'sample9',
   'sample15',
   'sample19',
   'sample6',
   'observe14',
   'observe10',
   'sample13',
   'sample4',
   'sample2',
   'sample1',
   'observe8',
   'sample3',
   'sample11',
   'observe16',
   'observe20'],
  'A': {'sample5': ['observe18',
    'observe12',
    'observe14',
    'observe10',
    'observe8',
    'observe16',
    'observe20'],
   'sample0': ['observe18',
    'observe12',
    'observe14',
    'observe10',
    'observe8',
    'observe16',
    'observe20'],
   'sample17': ['observe18'],
   'sample7': ['observe8'],
   'sample9': ['observe10'],
   'sample15': ['observe16'],
   'sample19': ['observe20'],
   'sample6': ['sample17',
    'sample7',
    'sample9',
    'sample15',
    'sample19',
    'sample13',
    'sample11'],
   'sample13': ['observe14'],
   'sample4': ['observe18',
    'observe12',
    'observe14',
    'observe10',
    'observe8',
    'observe16',
    'observe20'],
   'sample2': ['observe18',
    'observe12',
    'observe14',
    'observe10',
    'observe8',
    'observe16',
    'observe20'],
   'sample1': ['observe18',
    'observe12',
    'observe14',
    'observe10',
    'observe8',
    'observe16',
    'observe20'],
   'sample3': ['observe18',
    'observe12',
    'observe14',
    'observe10',
    'observe8',
    'observe16',
    'observe20'],
   'sample11': ['observe12']},
  'P': {'sample5': ['sample*', ['gamma', 1.0, 1.0]],
   'sample0': ['sample*', ['normal', 0.0, 10.0]],
   'observe18': ['observe*',
    ['get',
     ['vector',
      ['normal', 'sample0', 'sample1'],
      ['normal', 'sample2', 'sample3'],
      ['normal', 'sample4', 'sample5']],
     'sample17'],
    -0.1],
   'observe12': ['observe*',
    ['get',
     ['vector',
      ['normal', 'sample0', 'sample1'],
      ['normal', 'sample2', 'sample3'],
      ['normal', 'sample4', 'sample5']],
     'sample11'],
    2.0],
   'sample17': ['sample*', ['discrete', 'sample6']],
   'sample7': ['sample*', ['discrete', 'sample6']],
   'sample9': ['sample*', ['discrete', 'sample6']],
   'sample15': ['sample*', ['discrete', 'sample6']],
   'sample19': ['sample*', ['discrete', 'sample6']],
   'sample6': ['sample*', ['dirichlet', ['vector', 1.0, 1.0, 1.0]]],
   'observe14': ['observe*',
    ['get',
     ['vector',
      ['normal', 'sample0', 'sample1'],
      ['normal', 'sample2', 'sample3'],
      ['normal', 'sample4', 'sample5']],
     'sample13'],
    1.9],
   'observe10': ['observe*',
    ['get',
     ['vector',
      ['normal', 'sample0', 'sample1'],
      ['normal', 'sample2', 'sample3'],
      ['normal', 'sample4', 'sample5']],
     'sample9'],
    2.1],
   'sample13': ['sample*', ['discrete', 'sample6']],
   'sample4': ['sample*', ['normal', 0.0, 10.0]],
   'sample2': ['sample*', ['normal', 0.0, 10.0]],
   'sample1': ['sample*', ['gamma', 1.0, 1.0]],
   'observe8': ['observe*',
    ['get',
     ['vector',
      ['normal', 'sample0', 'sample1'],
      ['normal', 'sample2', 'sample3'],
      ['normal', 'sample4', 'sample5']],
     'sample7'],
    1.1],
   'sample3': ['sample*', ['gamma', 1.0, 1.0]],
   'sample11': ['sample*', ['discrete', 'sample6']],
   'observe16': ['observe*',
    ['get',
     ['vector',
      ['normal', 'sample0', 'sample1'],
      ['normal', 'sample2', 'sample3'],
      ['normal', 'sample4', 'sample5']],
     'sample15'],
    0.0],
   'observe20': ['observe*',
    ['get',
     ['vector',
      ['normal', 'sample0', 'sample1'],
      ['normal', 'sample2', 'sample3'],
      ['normal', 'sample4', 'sample5']],
     'sample19'],
    -0.05]},
  'Y': {'observe8': 1.1,
   'observe10': 2.1,
   'observe12': 2.0,
   'observe14': 1.9,
   'observe16': 0.0,
   'observe18': -0.1,
   'observe20': -0.05}},
 ['=', 'sample7', 'sample9']]
```

In [ ]:
```python
num_steps=1980
return_list, samples_whole_graph = mh_gibbs.mh_gibbs_wrapper(graph,num_steps)
```

In [44]:
```python
samples = np.array([sample.item() for sample in return_list])
```

In [54]:
```python
burn_in = 1500#int(0.01*num_steps)
pd.Series(samples[burn_in:]).astype(float).plot.hist()
plt.xlabel('(= (first z) (second z))')
plt.title('MH Gibbs | {}'.format(fname))
```

Out[54]:
```
Text(0.5, 1.0, 'MH Gibbs | 3.daphne')
```

In [55]:
```python
posterior_mean = samples[burn_in:].mean(0)
```

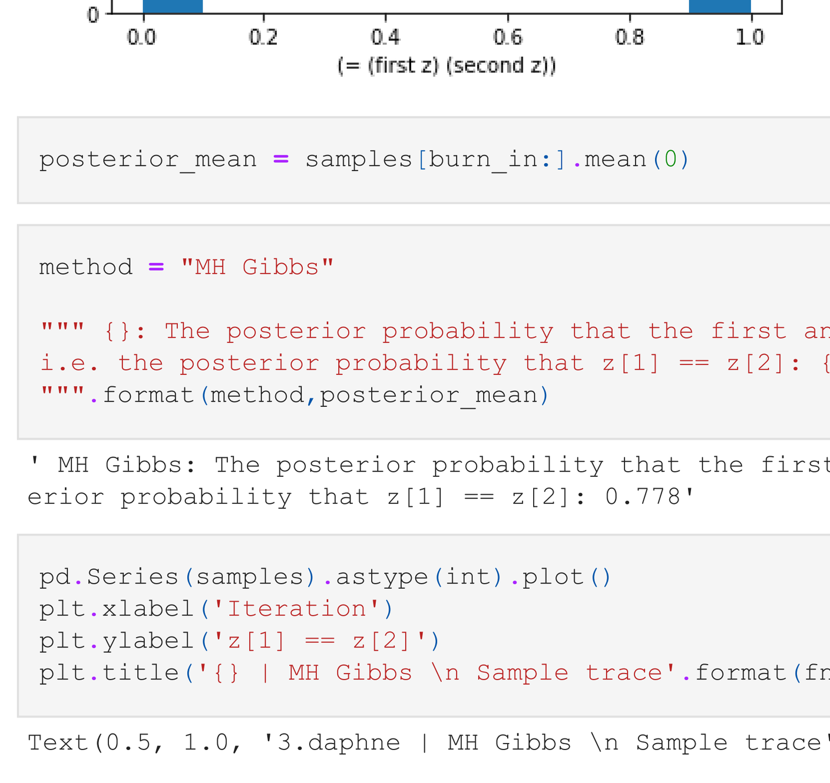
In [56]:
```python
method = "MH Gibbs"

""" {}: The posterior probability that the first and second datapoint are in the same cluster,\
i.e. the posterior probability that z[1] == z[2]: {:0.3f}\
""".format(method,posterior_mean)
```

Out[56]:
```
' MH Gibbs: The posterior probability that the first and second datapoint are in the same cluster,i.e. the post
erior probability that z[1] == z[2]: 0.778'
```

In [17]:
```python
pd.Series(samples).astype(int).plot()
plt.xlabel('Iteration')
plt.ylabel('z[1] == z[2]')
plt.title('{} | MH Gibbs \n Sample trace'.format(fname))
```

Out[17]:
```
Text(0.5, 1.0, '3.daphne | MH Gibbs \n Sample trace')
```



In [18]:
```python
G = graph[1]
Y = G['Y']
Y = {key:evaluate([value])[0] for key,value in Y.items()}
P = G['P']
```

In [19]:
```python
size = len(samples_whole_graph)
jll = np.zeros(size)
for idx in range(size):
    jll[idx] = compute_log_joint_prob(samples_whole_graph[idx],Y,P)
```

In [20]:
```python
pd.Series(jll).plot()
plt.xlabel('Iteration (t)')
plt.ylabel(r'$-\log p(X=x_t,Y=y_t)$')
plt.title('{} | MH Gibbs \n Joint density'.format(fname))
```

Out[20]:
```
Text(0.5, 1.0, '3.daphne | MH Gibbs \n Joint density')
```