Assignment #7

Thomas Lillis

1.
a. P(c = true) = 0.48
b. P(c = true | r = true) = 0.75
c. P(s = true | w = true) = 0.4
d. P(s = true | c = true, w = true) = 0.0

2.
a. P(c = true) = .5 (Given)

b. P(c = true | r = true) = P(r | c)P(c)/P(r)
P(r) = P(r | c)P(c)+P(r | ~c)P(~c)
P(c = true | r = true) = (P(r | c)P(c))/(P(P(r | c)P(c)+P(r | ~c)P(~c))
P(c = true | r = true) = (.8*.5)/(.8*.5+.2*.5) = 0.8

c. P(s = true | w = true) = P(w | s)P(s)/P(w)
P(s) = P(s | c)P(c)+P(s | ~c)P(~c)
P(w) = P(c)P(r | c)P(s | c)P(w | rs) + P(c)P(r | c)P(~s | c)P(w | r~s) + P(c)P(~r | c)P(s | c)P(w | ~rs) + P(c)P(~r | c)P(~s | c)P(w | ~r~s) + P(~c)P(r | ~c)P(s | ~c)P(w | rs) + P(~c)P(~r | ~c)P(s | ~c)P(w | ~rs) + P(~c)P(r | ~c)P(~s | ~c)P(w | r~s) + P(~c)P(~r | ~c)P(~s | ~c)P(w | ~r~s)

P(c)P(r | c)P(s | c)P(w | rs) = .5 * .8 * .1 * .99 = 0.0396
P(c)P(r | c)P(~s | c)P(w | r~s) = .5 * .8 * .9 * .9 = 0.324
P(c)P(~r | c)P(s | c)P(w | ~rs) = .5 * .2 * .1 * .9 = .009
P(c)P(~r | c)P(~s | c)P(w | ~r~s) = .5 * .2 * .9 * 0 = 0
P(~c)P(r | ~c)P(s | ~c)P(w | rs) = .5 * .2 * .5 * .99 = 0.0495
P(~c)P(~r | ~c)P(s | ~c)P(w | ~rs) = .5 * .8 * .5 *.9 = .18
(~c)P(r | ~c)P(~s | ~c)P(w | r~s) = .5 * .2 *.5 *.9 = .045
P(~c)P(~r | ~c)P(~s | ~c)P(w | ~r~s) = 0

P(w) = .6471
p(s) = P(s | c)P(c)+P(s | ~c)P(~c) = .1*.5 + .5*.5 = .3

P(w | s) = (0.0396+.009+0.0495+.18)/.3 = .2781/.3 = .927

P(s | w) = (.927 * .3)/.6471 = .4297

d. P(s = true | c = true, w = true) = P(swc)/P(wc) = (P(cs~rw) + P(csrw))/(P(csrw)+P(cs~rw)+P(c~srw) +P(c~s~rw))

P(cs~rw) = P(c)P(~r | c)P(s | c)P(w | ~rs) = .5 * .2 * .1 * .9 = .009
P(csrw) = P(c)P(r | c)P(s | c)P(w | rs) = .5 * .8 * .1 * .99 = 0.0396
P(c~srw) = P(c)P(r | c)P(~s | c)P(w | r~s) = .5 * .8 * .9 * .9 = 0.324
(c~s~rw) = P(c)P(~r | c)P(~s | c)P(w | ~r~s) = .5 * .2 * .9 * 0 = 0

P(s = true | c = true, w = true) = (.009 + 0.0396)/(0.0396+.009+0.324) = 0.1304347826

The error is within 0.5. This is because we are only using 100 random numbers which is 25 samples which is not very much.

3.
a. P(c = true) = 0.49
b. P(c = true | r = true) = 0.703703703704
c. P(s = true | w = true) = 0.4
d. P(s = true | c = true, w = true) = 0.0

4.
Rejection sampling did produce the same results for the last two because for the last two you need to calculate all of the probabilities which is what you are doing for prior sampling. For the first two the probabilities are different because you only need to find two probabilities so you can use more of the samples instead of some of the random numbers being used for the other nodes.

Code:
Prior.py:

```
import helpers
from random import random

print("Prior Sampling\n")

raw_samples = helpers.getSamples()
samples = []

raw_samples = []

for i in range(len(raw_samples)):
   if (i) % 4 == 0:
       sample = {
           "c": raw_samples[i],
           "s": raw_samples[i+1],
           "r": raw_samples[i+2],
           "w": raw_samples[i+3]
       }
       samples.append(sample)

test = []
num = 0
for sample in samples:
   num += 1
   test.append(helpers.priorCheck(sample))


#### P(c = true) ####
count = 0.0
for sample in test:
```

```
   if sample['c'] == True:
       count += 1

value = count/len(test)
print('P(c = true) = {0}'.format(value))


#### P(c = true | r = true) ####
value = 0.0
count = 0.0
count_total = 0.0
for sample in test:
   if sample['r'] == True:
       if sample['c'] == True:
           count += 1
       count_total += 1

value = count/count_total
print('P(c = true | r = true) = {0}'.format(value))


#### P(s = true | w = true) ####
value = 0.0
count = 0.0
count_total = 0.0
for sample in test:
   if sample['w'] == True:
       if sample['s'] == True:
           count += 1
       count_total += 1

value = count/count_total
print('P(s = true | w = true) = {0}'.format(value))


#### P(s = true | c = true, w = true) ####
value = 0.0
count = 0.0
count_total = 0.0
for sample in test:
   if sample['c'] == True and sample['w'] == True:
       if sample['s'] == True:
           count += 1
       count_total += 1

value = count/count_total
print('P(s = true | c = true, w = true) = {0}'.format(value))
```

Rejection.py:

```
import helpers
from random import random

print("Rejection Sampling\n")

raw_samples = helpers.getSamples()
samples = []
"""
raw_samples = []
for x in range(100000):
   raw_samples.append(random())
"""

for i in range(len(raw_samples)):
   if (i) % 4 == 0:
       sample = {
           "c": raw_samples[i],
           "s": raw_samples[i+1],
           "r": raw_samples[i+2],
           "w": raw_samples[i+3]
       }
       samples.append(sample)

test = []
for sample in samples:
   test.append(helpers.priorCheck(sample))


#### P(c = true) ####
count = 0.0
for sample in raw_samples:
   if sample <= helpers.C_TRUE:
       count += 1

value = count/len(raw_samples)
print('P(c = true) = {0}'.format(value))


#### P(c = true | r = true) ####
value = 0.0
count = 0.0
count_total = 0.0
for i in range(0,len(raw_samples),2):
   c = raw_samples[i]
   r = raw_samples[i+1]

   if c <= helpers.C_TRUE:
```

```
        c = True
    else:
        c = False

    if r <= helpers.R_TRUE_C_TRUE and c == True:
        r = True
    elif r > helpers.R_TRUE_C_TRUE and c == True:
        r = False
    elif r <= helpers.R_TRUE_C_FALSE and c == False:
        r = True
    elif r > helpers.R_TRUE_C_FALSE and c == False:
        r = False

    if r == True:
        if c  == True:
            count += 1
        count_total += 1

value = count/count_total
print('P(c = true | r = true) = {0}'.format(value))


#### P(s = true | w = true) ####
# Same as prior
value = 0.0
count = 0.0
count_total = 0.0
for sample in test:
    if sample['w'] == True:
        if sample['s'] == True:
            count += 1
        count_total += 1

value = count/count_total
print('P(s = true | w = true) = {0}'.format(value))

#### P(s = true | c = true, w = true) ####
# same as prior
value = 0.0
count = 0.0
count_total = 0.0
for sample in test:
    if sample['c'] == True and sample['w'] == True:
        if sample['s'] == True:
            count += 1
        count_total += 1

value = count/count_total
print('P(s = true | c = true, w = true) = {0}'.format(value))
```

Helpers.py:

```
C_TRUE = 0.5

S_TRUE_C_TRUE = .1
S_TRUE_C_FALSE = .5

R_TRUE_C_TRUE = .8
R_TRUE_C_FALSE = .2

W_TRUE_S_TRUE_R_TRUE = 0.99
W_TRUE_S_TRUE_R_FALSE = 0.90
W_TRUE_S_FALSE_R_TRUE = 0.90
W_TRUE_S_FALSE_R_FALSE = 0.00

def priorCheck(sample):
   ret =  {}
   if sample['c'] < C_TRUE:
       ret['c'] = True
       s_test = S_TRUE_C_TRUE
       r_test = R_TRUE_C_TRUE
   else:
       ret['c'] = False
       s_test = S_TRUE_C_FALSE
       r_test = R_TRUE_C_FALSE

   if sample['s'] < s_test:
       ret['s'] = True
   else:
       ret['s'] = False

   if sample['r'] < r_test:
       ret['r'] = True
   else:
       ret['r'] = False

   if ret['r'] == True and ret['s'] == True:
       w_test = W_TRUE_S_TRUE_R_TRUE
   if ret['r'] == True and ret['s'] == False:
       w_test = W_TRUE_S_TRUE_R_FALSE
   if ret['r'] == False and ret['s'] == True:
       w_test = W_TRUE_S_FALSE_R_TRUE
   if ret['r'] == False and ret['s'] == False:
       w_test = W_TRUE_S_FALSE_R_FALSE

   if sample['w'] < w_test:
       ret['w'] = True
   else:
```

```
        ret['w'] = False
    return ret

def getSamples():
    samples_file = open('random_samples')
    samples_strings = samples_file.read().split(',')
    samples = []

    for i in range(len(samples_strings)):
        samples.append(float(samples_strings[i]))

    print "SAMPLES: \n"
    print samples
    print ""
    return samples
```