## Case Study 6 Homework: Exercises 1-4

Exercise 1

1/1 point (graded)

In Exercise 1, we will calculate the chance homophily for an arbitrary characteristic. Homophily is the proportion of edges in the network whose constituent nodes share that characteristic.

How much homophily do we expect by chance? If characteristics are distributed completely randomly, the probability that two nodes

$x$

and

$y$

share characteristic

$a$

is the probability both nodes have characteristic

$a$

, which is the marginal probability of

$a$

squared. The total probability that nodes

$x$

and

$y$

share their characteristic is therefore the sum of the square of the marginal probabilities of each characteristic in the network.

**Instructions**

- Create a function `marginal_prob` that takes a dictionary `chars` with personal IDs as keys and characteristics as values; it should return a dictionary with characteristics as keys and their marginal probability (frequency of occurence of a characteristic divided by the sum of frequencies of each characteristic) as values.
- Create a function `chance_homophily(chars)` that takes a dictionary `chars` defined as above and computes the chance homophily (homophily due to chance alone) for that characteristic.
- A sample of three peoples' favorite colors is given in `favorite_colors`. Use your function to compute the chance homophily in this group, and store it as `color_homophily`.
- Print `color_homophily`.

Here's the code to get you started:

```
       from collections import Counter
import numpy as np
```

```
def marginal_prob(chars):
    # Enter code here!

def chance_homophily(chars):
    # Enter code here!

favorite_colors = {
    "ankit":  "red",
    "xiaoyu": "blue",
    "mary":   "blue"
}

color_homophily = chance_homophily(favorite_colors)
print(color_homophily)
```

What is the chance homophily of the `favorite_colors` dictionary?

Answer = [0.555555555556]

Code = [

```
 from collections import Counter
import numpy as np

def marginal_prob(chars):
    # Enter code here!

def chance_homophily(chars):
    # Enter code here!

favorite_colors = {
    "ankit":  "red",
    "xiaoyu": "blue",
    "mary":   "blue"
}

color_homophily = chance_homophily(favorite_colors)
print(color_homophily).  ]
```

Exercise 2

0/1 point (graded)

In the remaining exercises, we will calculate actual homophily in these village and compare the obtained values to those obtained by chance. In Exercise 2, we subset the data into individual villages and store them.

**Instructions**

- Note that `individual_characteristics.dta` contains several characteristics for each individual in the dataset such as age, religion, and caste. Use the `pandas` library to read in and store these characteristics as a dataframe called `df`.
- Store separate datasets for individuals belonging to Villages 1 and 2 as `df1` and `df2`, respectively.
- Note that some attributes may be missing for some individuals.
- Use the `head` method to display the first few entries of `df1`.

Here is the code to get you started:

```
        import pandas as pd

df  = pd.read_csv("https://courses.edx.org/asset-
v1:HarvardX+PH526x+2T2019+type@asset+block@individual_c
haracteristics.csv", low_memory=False, index_col=0)
df1 = # Enter code here!
df2 = # Enter code here!
```

```
# Enter code here!
```

How many people had a `resp_gender` value of 1 in the first 5 entries of `df1`?

Answer = [3]

Code = [

```
import pandas as pd

df  = pd.read_csv("https://courses.edx.org/asset-
v1:HarvardX+PH526x+2T2019+type@asset+block@individual_characteristics.csv",
low_memory=False, index_col=0)
df1 = df[df["village"]==1]
df2 = df[df["village"]==2]

df1.head()
village     adjmatrix_key     pid    hhid  resp_id     resp_gend    resp_status
age   religion    caste ...   privategovt work_outside work_outside_freq
shgparticipate    shg_no savings     savings_no  electioncard rationcard
rationcard_colour
0     1     5     1002011002  1     1     Head of Household  38    HINDUISM
OBC   ...   PRIVATE BUSINESS  Yes   0.0   No    NaN   No    NaN   Yes   Yes
GREEN
1     1     6     1002021002  2     2     Spouse of Head of Household    27
HINDUISM    OBC   ...   NaN   NaN   NaN   No    NaN   No    NaN   Yes   Yes
GREEN
2     1     23    1006011006  1     1     Head of Household  29    HINDUISM
OBC   ...   OTHER LAND  No    NaN   No    NaN   No    NaN   Yes   Yes   GREEN
3     1     24    1006021006  2     2     Spouse of Head of Household    24
HINDUISM    OBC   ...   PRIVATE BUSINESS  No    NaN   Yes   1.0   Yes   1.0
Yes   No    NaN
4     1     27    1007011007  1     1     Head of Household  58    HINDUISM
OBC   ...   OTHER LAND  No    NaN   No    NaN   No    NaN   Yes   Yes   GREEN
5 rows × 48 columns
```

]

## Exercise 3

1/1 point (graded)

In Exercise 3, we define a few dictionaries that enable us to look up the sex, caste, and religion of members of each village by personal ID. For Villages 1 and 2, their personal IDs are stored as `pid`.

**Instructions**

- Define dictionaries with personal IDs as keys and a given covariate for that individual as values. Complete this for the sex, caste, and religion covariates, for Villages 1 and 2.
- For Village 1, store these dictionaries into variables named `sex1`, `caste1`, and `religion1`.
- For Village 2, store these dictionaries into variables named `sex2`, `caste2`, and `religion2`.

Here is some code to get you started:

```
sex1      = # Enter code here!
caste1    = # Enter code here!
religion1 = # Enter code here!

# Continue for df2 as well.
```

What is the `caste` value for personal ID 202802 in village 2?

Answer = [OBC]

Code = [

```
ex1 = df1.set_index("pid")["resp_gend"].to_dict()
caste1 = df1.set_index("pid")["caste"].to_dict()
religion1 = df1.set_index("pid")["religion"].to_dict()
```

```
sex2 = df2.set_index("pid")["resp_gend"].to_dict()
caste2 = df2.set_index("pid")["caste"].to_dict()
religion2 = df2.set_index("pid")["religion"].to_dict()

caste2[202802]
```

]

## Exercise 4

0/1 point (graded)

In Exercise 4, we will print the chance homophily of several characteristics of Villages 1 and 2.

**Instructions**

Use `chance_homophily` to compute the chance homophily for sex, caste, and religion In Villages 1 and 2. Consider whether the chance homophily for any attribute is very high for either village.

Which characteristic has the highest value of chance homophiliy?

Village 1, sex

Village 2, sex

Village 1, caste

Village 2, caste

Village 1, religion

Village 2, religion

correct

Code = [

```
print("Village 1 chance of same sex:", chance_homophily(sex1))
print("Village 1 chance of same caste:", chance_homophily(caste1))
print("Village 1 chance of same religion:", chance_homophily(religion1))

print("Village 2 chance of same sex:", chance_homophily(sex2))
print("Village 2 chance of same caste:", chance_homophily(caste2))
print("Village 2 chance of same religion:", chance_homophily(religion2))
Village 1 chance of same sex: 0.5027299861680701
Village 1 chance of same caste: 0.6741488509791551
Village 1 chance of same religion: 0.9804896988521925
Village 2 chance of same sex: 0.5005945303210464
Village 2 chance of same caste: 0.425368244800893
Village 2 chance of same religion: 1.0
```

]

Exercise 5

1/1 point (graded)

In Exercise 5, we will create a function that computes the observed homophily given a village and characteristic.

**Instructions**

Complete the function `homophily()`, which takes a network `G`, a dictionary of node characteristics `chars`, and node IDs `IDs`. For each node pair, determine whether a tie exists between them, as well as whether they share a characteristic. The total count of these is `num_ties` and `num_same_ties`, respectively, and their ratio is the homophily of `chars` in `G`. Complete the function by choosing where to increment `num_same_ties` and `num_ties`.

Complete this function:

```
       def homophily(G, chars, IDs):
    """
    Given a network G, a dict of characteristics chars
for node IDs,
    and dict of node IDs for each node in the network,
    find the homophily of the network.
    """
    num_same_ties = 0
    num_ties = 0
    for n1, n2 in G.edges():
        if IDs[n1] in chars and IDs[n2] in chars:
            if G.has_edge(n1, n2):
                # Should `num_ties` be incremented?
What about `num_same_ties`?
                if chars[IDs[n1]] == chars[IDs[n2]]:
```

```
                    # Should `num_ties` be incremented?
What about `num_same_ties`?
    return (num_same_ties / num_ties)
```

What should be done if the first conditional statement, `if G.has_edge(n1, n2)`, is `True`?

Increment `num_ties` and decrement `num_same_ties`

Do nothing to `num_ties` and increment `num_same_ties`

Decrement `num_ties` and do nothing to `num_same_ties`

Increment `num_ties` and do nothing to `num_same_ties`
Correct

Code = [

```
def homophily(G, chars, IDs):
    """
    Given a network G, a dict of characteristics chars for node IDs,
    and dict of node IDs for each node in the network,
    find the homophily of the network.
    """
    num_same_ties = 0
    num_ties = 0
    for n1, n2 in G.edges():
        if IDs[n1] in chars and IDs[n2] in chars:
            if G.has_edge(n1, n2):
                num_ties += 1
                if chars[IDs[n1]] == chars[IDs[n2]]:
                    num_same_ties += 1
    return (num_same_ties / num_ties)
```

]

Exercise 6

0/1 point (graded)

In Exercise 6, we will obtain the personal IDs for Villages 1 and 2. These will be used in the next exercise to calculate homophily for these villages.

**Instructions**

In this dataset, each individual has a personal ID, or PID, stored in `key_vilno_1.csv` and `key_vilno_2.csv` for villages 1 and 2,
respectively. `data_filepath1` and `data_filepath2` contain the URLs to the datasets used in this exercise.
Use `pd.read_csv` to read in and
store `key_vilno_1.csv` and `key_vilno_2.csv` as `pid1` and `pid2` respectively.

The code to get you started can be found here:

```
      data_filepath1 = "https://courses.edx.org/asset-v1:HarvardX+PH526x+2T2019+type@asset+block@key_vilno_1.csv"
data_filepath2 = "https://courses.edx.org/asset-v1:HarvardX+PH526x+2T2019+type@asset+block@key_vilno_2.csv"

# Enter code here!
```

What is the personal ID of the person at index 100 in village 1?

Answer = [102205]

Code = [

```
    pid1 = pd.read_csv(data_filepath1, index_col=0)
pid2 = pd.read_csv(data_filepath2, index_col=0)

pid1.iloc[100]
0    102205
Name: 100, dtype: int64
```

]

## Exercise 7

0.33/1 point (graded)

In Exercise 7, we will compute the homophily of several network characteristics for Villages 1 and 2 and compare them to homophily due to chance alone. The networks for these villages have been stored as networkx graph objects G1 and G2.

### Instructions

- Use your `homophily()` function to compute the observed homophily for sex, caste, and religion in Villages 1 and 2. Print all six values.
- Use `chance_homophily()` to compare the observed homophily values to the chance homophily values. Are observed values higher or lower than those expected by chance?

Here's the code to get you started:

```
        import networkx as nx
A1 = np.array(pd.read_csv("https://courses.edx.org/asset-v1:HarvardX+PH526x+2T2019+type@asset+block@adj_allVillageRelationships_vilno1.csv", index_col=0))
```

```
A2 = np.array(pd.read_csv("https://courses.edx.org/
asset-
v1:HarvardX+PH526x+2T2019+type@asset+block@adj_allVilla
geRelationships_vilno2.csv", index_col=0))
G1 = nx.to_networkx_graph(A1)
G2 = nx.to_networkx_graph(A2)

pid1 = pd.read_csv(data_filepath1, dtype=int)
['0'].to_dict()
pid2 = pd.read_csv(data_filepath2, dtype=int)
['0'].to_dict()

# Enter your code here!
```

For which characteristics is the observed homophily higher than the chance homophily?

Select ALL that apply.

Village 1 sex

correct

Village 2 sex

correct

Village 1 caste

correct

Village 2 caste

correct

Village 1 religion

correct

## Village 2 religion

## Code = [

```
import networkx as nx
A1 = np.array(pd.read_csv("https://courses.edx.org/asset-
v1:HarvardX+PH526x+2T2019+type@asset+block@adj_allVillageRelationships_vilno1.csv"
, index_col=0))
A2 = np.array(pd.read_csv("https://courses.edx.org/asset-
v1:HarvardX+PH526x+2T2019+type@asset+block@adj_allVillageRelationships_vilno2.csv"
, index_col=0))
G1 = nx.to_networkx_graph(A1)
G2 = nx.to_networkx_graph(A2)

pid1 = pd.read_csv(data_filepath1, dtype=int)['0'].to_dict()
pid2 = pd.read_csv(data_filepath2, dtype=int)['0'].to_dict()

print("Village 1 observed proportion of same sex:", homophily(G1, sex1, pid1))
print("Village 1 observed proportion of same caste:", homophily(G1, caste1, pid1))
print("Village 1 observed proportion of same religion:", homophily(G1, religion1,
pid1))

print("Village 2 observed proportion of same sex:", homophily(G2, sex2, pid2))
print("Village 2 observed proportion of same caste:", homophily(G2, caste2, pid2))
print("Village 2 observed proportion of same religion:", homophily(G2, religion2,
pid2))

print("Village 1 chance of same sex:", chance_homophily(sex1))
print("Village 1 chance of same caste:", chance_homophily(caste1))
print("Village 1 chance of same religion:", chance_homophily(religion1))

print("Village 2 chance of same sex:", chance_homophily(sex2))
print("Village 2 chance of same caste:", chance_homophily(caste2))
print("Village 2 chance of same religion:", chance_homophily(religion2))
Village 1 observed proportion of same sex: 0.5908629441624366
Village 1 observed proportion of same caste: 0.7959390862944162
Village 1 observed proportion of same religion: 0.9908629441624366
Village 2 observed proportion of same sex: 0.5658073270013568
Village 2 observed proportion of same caste: 0.8276797829036635
Village 2 observed proportion of same religion: 1.0
Village 1 chance of same sex: 0.5027299861680701
Village 1 chance of same caste: 0.6741488509791551
Village 1 chance of same religion: 0.9804896988521925
Village 2 chance of same sex: 0.5005945303210464
Village 2 chance of same caste: 0.425368244800893
Village 2 chance of same religion: 1.0
```

]