Report 3. Unequal Probability Design.

Name of the project: Analysis of Average Winter Temperature in College Park, Maryland

The true parameter: the true average temperature (in ℉) in College Park, MD during winter months, μ = 39.177

The population size *N* = 720

The sample size *n* = 84

1. Take a variable that is related to your variable of interest (correlated) and assign p_i's proportional to it. You can take a categorical variable which is related to y. Note in this case we do not talk about correlation. For categorical variable p_i is equal to its relative frequency.

First, we will take the **dew** variable, as it is correlated with temperature.

We will assign p_i values as follows: The probability of a particular item being chosen is its observed dew value, over the total sum of all dew values.

Note: Some of the dew point values are negative. To deal with this, we will take the absolute value of those values, so each Pi will be | dew | / sum(| dews |).

Sum of all dew values in the data (after taking absolute value), yielding the result: 19582.9

Pi's will be each |dew| / 19582.9. We will call this p in our code.

Another Note: One of the entries in our data had the value 0.0 for dew, so that row was omitted from our population data (figured out that value was causing the true variances to be very high).

Instead of N = 720, it is now N = 719. The highlighted gray in the code displays the new code that was added.

Code:

```
import math

import pandas as pd

import scipy.stats as scipy



data = pd.read_csv('/Users/sophiahu/downloads/winterWeather.csv', usecols= ['temp', 'dew'])
```

```
n = 84

N = 720



#print(data['dew'].describe())

#print(data['temp'].corr(data['dew']))



print("Original # rows in data:", len(data))

#change negative dews to be absolute value to get rid of negative dews

for i in data.index:

    if data.at[i, "dew"] == 0:

        data.drop([i], axis=0, inplace=True)

print("New # rows in data:", len(data))

#adds probability column to each entry

data["p"] = abs(data["dew"])/abs(data["dew"]).sum() #absolute value of dew data for negatives

print(data)

print("Sum: ", data["p"].sum())

print("Dew Sum: ", data["dew"].sum())

#print(data['temp'].corr(data['dew']))
```

Output:

Original # rows in data: 720

New # rows in data: 719

	temp	dew	p
0	56.9	44.9	0.002293
1	37.9	32.8	0.001675

2 45.0 38.8 0.001981

3 40.6 26.4 0.001348

4 40.9 32.8 0.001675

.. ..

715 35.2 26.3 0.001343

716 39.9 31.4 0.001603

717 36.1 19.7 0.001006

718 43.1 19.2 0.000980

719 41.1 19.6 0.001001

[719 rows x 3 columns]

Sum: 1.0

Dew Sum: 19582.9

2. Take a sample of size n (that you define using SRS design) with replacement with p_i defined above.

Code:

```
#takes sample of n=84 with replacement with probabilities using dew

sample = data.sample(n=84,replace=True, weights=data["p"])

print(sample)
```

Output:

```
      temp   dew         p
557   33.5   26.1   0.001333
607   41.0   22.2   0.001134
264   62.1   52.7   0.002691
222   61.1   50.4   0.002574
328   50.7   48.2   0.002461
```

```
..    ...   ...       ...
407  34.6  24.4  0.001246
489  32.0  16.2  0.000827
649  32.3  18.7  0.000955
540  43.4  28.4  0.001450
325  34.5  17.4  0.000889

[84 rows x 3 columns]
```

3. Estimate your parameter of interest by Hansen-Hurwitz estimator. Estimate its variance and give a confidence interval of α level that you choose in SRS design.

Code:

```
#HH ESTIMATOR

#mu

sum = 0

for i, row in sample.iterrows():

   sum = sum + row["temp"]/row["p"]



tau_p_hat = sum/n

mu_p_hat = tau_p_hat/N

print("mu_p_hat: ", mu_p_hat)



#var

sum2 = 0

for i, row in sample.iterrows():

   sum2 = sum2 + (((row["temp"]/row["p"]) - tau_p_hat)**2)
```

```
var_hat_tau_p_hat = sum2/(n*(n-1))

var_hat_mu_p_hat = var_hat_tau_p_hat/(N**2)

print("var_hat_mu_p_hat: ", var_hat_mu_p_hat)







#find T critical value

t = scipy.t.ppf(q=1-.05/2,df=n-1)

CILower = mu_p_hat - t*math.sqrt(var_hat_mu_p_hat)

CIUpper = mu_p_hat + t*math.sqrt(var_hat_mu_p_hat)



print("CI for HH: ", CILower, CIUpper)
```

Output:

mu_p_hat: 38.711544996047216

var_hat_mu_p_hat: 1.3867968551637893

CI for HH: 36.36929943188548 41.05379056020895

Hansen-Hurwitz Estimator, estimate of true population mean:

$$\hat{\mu}_p = \left[\left(\frac{1}{n}\right)\sum_{i=1}^{n}\frac{y_i}{p_i}\right]/N = 38.7115$$

Estimated variance of this estimator:

$$\hat{var}(\hat{\mu}_p) = \frac{1}{N^2}\hat{var}(\hat{\tau}_p) = \frac{1}{N^2}\left[\left(\frac{1}{n(n-1)}\right)\sum_{i=1}^{n}\left(\frac{y_i}{p_i}-\hat{\tau}_p\right)^2\right] = 1.3868$$

95% Confidence Interval:

$\hat{\mu}_p \pm t_{n-1}\sqrt{\hat{var}(\hat{\mu}_p)}$ = (36.3693, 41.0538)

4. Calculate π_i's for selected sampling units.

$\pi_i = 1 - (1 - p_i)^n$

$\pi_{ij} = \pi_i + \pi_j - 1 + (1 - p_i - p_j)^n$

Code:

```
#pi_i's for all data

data["pi_i"] = 1-((1-data["p"])**n)



#pi_i's for selected sampling units

sample["pi_i"] = 1-((1-sample["p"])**n)

print(sample)



#removes duplicates in sample

uniqueSample = sample.drop_duplicates()

print("Unique sample count: ", len(uniqueSample))
```

Output:

```
     temp   dew         p      pi_i
557  33.5   26.1  0.001333  0.105982
607  41.0   22.2  0.001134  0.090882
264  62.1   52.7  0.002691  0.202568
222  61.1   50.4  0.002574  0.194641
328  50.7   48.2  0.002461  0.186986
```

```
..    ...   ...       ...       ...
407   34.6  24.4  0.001246  0.099431
489   32.0  16.2  0.000827  0.067157
649   32.3  18.7  0.000955  0.077115
540   43.4  28.4  0.001450  0.114771
325   34.5  17.4  0.000889  0.071950

[84 rows x 4 columns]
Unique sample count:  81
```

5. Estimate your parameter of interest by Horvitz-Thompson estimator. Estimate its variance and give a confidence interval of α level that you choose in SRS design.

Code:

```
#mu

sum3 = 0

for i, row in uniqueSample.iterrows():

   sum3 = sum3 + (row["temp"]/row["pi_i"])



mu_pi_hat = sum3/N

print("mu_pi_hat", mu_pi_hat)



#var

first_part = 0

second_part = 0



for i, row in uniqueSample.iterrows():
```

```
   first_part = first_part + ( (1/(row["pi_i"]**2) - (1/row["pi_i"]) ) * (row["temp"]**2) )



sum4 = 0

for i in range(0, len(uniqueSample)-1):

   for j in range(i+1, len(uniqueSample)):

       #row, col

       yi = uniqueSample.iloc[i, 0]

       yj = uniqueSample.iloc[j, 0]

       pi_i = uniqueSample.iloc[i, 3]

       pi_j = uniqueSample.iloc[j, 3]

       p_i = uniqueSample.iloc[i, 2]

       p_j = uniqueSample.iloc[j, 2]



       pi_ij = pi_i  + pi_j - 1 + ((1 - p_i - p_j)**n)



       second_part = second_part + ( ( (1/(pi_i*pi_j)) - (1/pi_ij) ) * yi * yj )



var_hat_mu_pi_hat = (first_part + 2*second_part) / (N**2)

print("var_hat_mu_pi_hat: ", var_hat_mu_pi_hat)

#find T critical value

v = len(uniqueSample)

t = scipy.t.ppf(q=1-.05/2,df=v-1)

CILower = mu_pi_hat - t*math.sqrt(var_hat_mu_pi_hat)

CIUpper = mu_pi_hat + t*math.sqrt(var_hat_mu_pi_hat)
```

```
print("CI for HT: ", CILower, CIUpper)
```

Output:

mu_pi_hat: 40.06004761376848

var_hat_mu_pi_hat: 2.0218090214970683

CI for HT: 37.2303698493423 42.88972537819466

Horvitz-Thompson Estimator, estimate of true population mean:

$$\hat{\mu}_{\pi} = \left[\sum_{i=1}^{v} \frac{y_i}{\pi_i}\right]/N = 40.0600$$

Estimated variance of this estimator:

$$\hat{var}(\hat{\mu}_{\pi}) = \frac{1}{N^2}\hat{var}(\hat{\tau}_{\pi}) = \frac{1}{N^2}\left[\sum_{i=1}^{v}\left(\frac{1}{\pi_i^2} - \frac{1}{\pi_i}\right)y_i^2 + 2\sum_{i=1}^{v}\sum_{i>j}\left(\frac{1}{\pi_i\pi_j} - \frac{1}{\pi_{ij}}\right)y_i y_j\right] = 2.0218$$

95% Confidence Interval:

$$\hat{\mu}_{\pi} \pm t_{v-1}\sqrt{\hat{var}(\hat{\mu}_{\pi})} = (37.2304, 42.8897)$$

6. Estimate your parameter of interest by Generalized Unequal Probability estimator ($\hat{\mu}_g$). Estimate its variance and give a confidence interval of α level that you choose in SRS design. Notice, since $\hat{\mu}_g$ is not unbiased estimator it is possible that your parameter would be off the confidence interval.

Code:

```
#GEN ESTIMATOR

#mu

sum5 = 0

sum6 = 0

for i, row in uniqueSample.iterrows():

   sum5 = sum5 + (row["temp"]/row["pi_i"])

   sum6 = sum6 + (1/row["pi_i"])
```

```
mu_g_hat = sum5/sum6

print("mu_g_hat: ", mu_g_hat)



#var

first_part = 0

second_part = 0



for i, row in uniqueSample.iterrows():

   first_part = first_part + ( ((1-row["pi_i"])/(row["pi_i"]**2)) * ((row["temp"] -
mu_g_hat)**2) )

 for i in range(0, len(uniqueSample)-1):

   for j in range(i+1, len(uniqueSample)):

       #row, col

       yi = uniqueSample.iloc[i, 0]

       yj = uniqueSample.iloc[j, 0]

       pi_i = uniqueSample.iloc[i, 3]

       pi_j = uniqueSample.iloc[j, 3]

       p_i = uniqueSample.iloc[i, 2]

       p_j = uniqueSample.iloc[j, 2]



       pi_ij = pi_i  + pi_j - 1 + ((1 - p_i - p_j)**n)



       second_part = second_part + ( ( (pi_ij - pi_i*pi_j)/(pi_i*pi_j) ) * ((yi-mu_g_hat) *
(yj-mu_g_hat)/pi_ij) )
```

```
var_hat_mu_g_hat = (first_part + second_part) / (N**2)

print("var_hat_mu_g_hat", var_hat_mu_g_hat)



#find T critical value

v = len(uniqueSample)

t = scipy.t.ppf(q=1-.05/2,df=v-1)

CILower = mu_g_hat - t*math.sqrt(var_hat_mu_g_hat)

CIUpper = mu_g_hat + t*math.sqrt(var_hat_mu_g_hat)

print("CI for Gen Est: ", CILower, CIUpper)
```

Output:

```
mu_g_hat:  40.95326093958118

var_hat_mu_g_hat 0.9678636062546507

CI for Gen Est:  38.99543536640442 42.91108651275793
```

Generalized Unequal Probability Estimator, estimate of true population mean:

$$\hat{\mu}_g = \left[\sum_{i=1}^{v} \frac{y_i}{\pi_i} \Big/ \sum_{i=1}^{v} \frac{1}{\pi_i} \right] = 40.9533$$

Estimated variance of this estimator:

$$\widehat{var}(\hat{\mu}_g) = \frac{1}{N^2}\left[\sum_{i=1}^{v} \left(\frac{1-\pi_i}{\pi_i^2} \right) (y_i - \hat{\mu}_g)^2 + \sum_{i=1}^{v} \sum_{i \neq j} \left(\frac{\pi_{ij} - \pi_i \pi_j}{\pi_i \pi_j} \right) \left(\frac{(y_i - \hat{\mu}_g)(y_j - \hat{\mu}_g)}{\pi_{ij}} \right) \right] = 0.9679$$

95% Confidence Interval:

$$\hat{\mu}_g \pm t_{v-1} \sqrt{\widehat{var}(\hat{\mu}_g)} = (38.9954,\ 42.9111)$$

7. Choose the best estimator of your parameter based on width of CI (estimated variance).

The best estimator in terms of estimated variance (and width of confidence intervals) is the Generalized Unequal Probability Estimator. The estimator with the next most narrow confidence interval is the Hansen-Hurwitz Estimator. Lastly, the HT Estimator performed the worst with the largest variance and CI width.

8. Calculate variances of your estimators. Is your choice of best estimator above change? How does it change?

Calculations for true variance:

$$var(\hat{\mu}_p) = \frac{1}{N^2} var(\hat{\tau}_p) = \frac{1}{N^2}\left[(\frac{1}{n})\sum_{i=1}^{N} p_i\left(\frac{y_i}{p_i} - \tau\right)^2\right] = 1.7837$$

$$var(\hat{\mu}_\pi) = \frac{1}{N^2} var(\hat{\tau}_\pi) = \frac{1}{N^2}\left[\sum_{i=1}^{N}\left(\frac{1-\pi_i}{\pi_i}\right){y_i}^2 + \sum_{i=1}\sum_{j\neq i}\left(\frac{\pi_{ij}-\pi_i\pi_j}{\pi_i\pi_j}\right)y_i y_j\right] = 10.9298$$

$$var(\hat{\mu}_g) = \frac{1}{N^2}\left[\sum_{i=1}^{N}\left(\frac{1-\pi_i}{\pi_i}\right)(y_i - \mu)^2 + \sum_{i=1}^{N}\sum_{j\neq i}\left(\frac{\pi_{ij}-\pi_i\pi_j}{\pi_i\pi_j}\right)(y_i - \mu)(y_j - \mu)\right] = 0.1922$$

Based on the calculations for the true variances of the estimators, our choice of best estimator would not change because the true variance for the Generalized Unequal Probability Estimator is the smallest at 0.1922 as its estimated variance is also the smallest. Similarly to the calculations from the estimated variances, the next smallest true variance is the Hansen-Hurwitz Estimator at 1.7837. The largest true variance is the Horvitz-Thompson Estimator at 10.9298, just as it was for the calculations from the estimated variances.

Code:

```
import math

import pandas as pd

import scipy.stats as scipy



data = pd.read_csv('/Users/sophiahu/downloads/winterWeather.csv', usecols= ['temp', 'dew'])

n = 84

N = 720
```

```
#remove row where dew = 0
for i in data.index:
   if data.at[i, "dew"] == 0:
       data.drop([i], axis=0, inplace=True)

data["p"] = (data["dew"])/(data["dew"]).sum() #Not using absolute value dew values here
data["pi_i"] = 1-((1-data["p"])**n)

#true variances
#true variance for hh

tau = data["temp"].sum()
print("TAU", tau)

sum = 0
N = 719
for i, row in data.iterrows():
   p_i = row["p"]
   y_i = row["temp"]
   sum = sum + (p_i*((y_i/p_i-tau)**2))

true_var_tau_p_hat = sum/n
true_var_mu_p_hat = true_var_tau_p_hat/(N**2)
```

```
print("True var HH: ", true_var_mu_p_hat)



#true variance for ht

first_part = 0

second_part = 0

for i, row in data.iterrows():

   first_part = first_part + (((1-row["pi_i"])/row["pi_i"])*(row["temp"]**2))



#iterate over all pairs in uniqueSample

for i in range(0, len(data)-1):

   for j in range(i+1, len(data)):

       #row, col

       yi = data.iloc[i, 0]

       yj = data.iloc[j, 0]

       pi_i = data.iloc[i, 3]

       pi_j = data.iloc[j, 3]

       p_i = data.iloc[i, 2]

       p_j = data.iloc[j, 2]



       pi_ij = pi_i  + pi_j - 1 + ((1 - p_i - p_j)**n)

       second_part = second_part + ( ((pi_ij-pi_i*pi_j)/(pi_i*pi_j)) * yi * yj )



true_var_mu_HT = (first_part + second_part) / (N**2)

print("True var HT: ", true_var_mu_HT)
```

```
#true variance for gen prob

mu = data["temp"].sum()/N

first_part = 0

second_part = 0



for i, row in data.iterrows():

   first_part = first_part + ( ((1-row["pi_i"])/(row["pi_i"])) * ((row["temp"] - mu)**2) )



#iterate over all pairs in uniqueSample

for i in range(0, len(data)-1):

   for j in range(i+1, len(data)):

       #row, col

       yi = data.iloc[i, 0]

       yj = data.iloc[j, 0]

       pi_i = data.iloc[i, 3]

       pi_j = data.iloc[j, 3]

       p_i = data.iloc[i, 2]

       p_j = data.iloc[j, 2]



       pi_ij = pi_i  + pi_j - 1 + ((1 - p_i - p_j)**n)

       second_part = second_part +  (((pi_ij-pi_i*pi_j)/(pi_i*pi_j)) * (yi-mu) * (yj-mu) )



true_var_Gen = (first_part + second_part) / (N**2)

print("True var Gen Est: ", true_var_Gen)
```

Output:

TAU 28189.3

True var HH: 1.7837059286408554

True var HT: 10.929769902552328

True var Gen Est: 0.19220654353328126

9. Repeat steps 1-9 with another variable.

Next, *we will take the **humidity** variable. Here, we take a sample with replacement with selected* $p_i's$ *proportional to humidity of size n=84.*

Repeating 1)

Now, *we will take the humidity variable. Here, we take a sample with replacement with selected* $p_i's$ *proportional to humidity of size n=84.*

We will assign p_i values as follows: The probability of a particular item being chosen is its observed humidity value, over the total sum of all humidity values.

Sum of all humidity values in the data (after taking absolute value), yields the result: 46153.99

Pi's will be each humidity / 46153.99. We will call this p in our code.

Code:

```
import pandas as pd

import scipy.stats as scipy

import math



data = pd.read_csv('/Users/sophiahu/downloads/winterWeather.csv', usecols= ['temp',
'humidity'])

n = 84

N = 720

#adds probability column to each entry
```

```
data["p"] = data["humidity"]/data["humidity"].sum()



print(data)

print("Humidity Sum: ", data["humidity"].sum())
```

Output:

temp humidity p

0 56.9 65.00 0.001408

1 37.9 82.00 0.001777

2 45.0 80.20 0.001738

3 40.6 57.10 0.001237

4 40.9 73.00 0.001582

.. ..

715 35.2 71.37 0.001546

716 39.9 74.48 0.001614

717 36.1 51.58 0.001118

718 43.1 42.71 0.000925

719 41.1 42.87 0.000929

[720 rows x 3 columns]

Humidity Sum: 46153.98999999999

Repeating 2)

Code:

```
#takes sample of n=84 with replacement with probabilities using humidity

sample = data.sample(n=84,replace=True, weights=data["p"])

print(sample)
```

Output:

```
     temp  humidity         p
208  41.1     76.02  0.001647
154  47.7     88.63  0.001920
129  44.9     91.07  0.001973
388  47.7     72.90  0.001579
325  34.5     50.20  0.001088
..    ...       ...       ...
148  31.4     70.00  0.001517
188  32.8     46.86  0.001015
94   41.0     72.06  0.001561
302  16.9     45.70  0.000990
164  20.5     45.42  0.000984

[84 rows x 3 columns]
```

Repeating 3)

Code: Same code used as before.

Output:

mu_p_hat: 39.710831769433554

var_hat_mu_p_hat: 1.225854538647234

CI for HH: 37.508689005120786 41.91297453374632

Hansen-Hurwitz Estimator, estimate of true population mean:

$$\hat{\mu}_p = \left[\left(\frac{1}{n}\right) \sum_{i=1}^{n} \frac{y_i}{p_i} \right] / N = 39.7108$$

Estimated variance of this estimator:

$\widehat{var}(\widehat{\mu}_p) = \frac{1}{N^2}\widehat{var}(\widehat{\tau}_p) = \frac{1}{N^2}\left[\left(\frac{1}{n(n-1)}\right)\sum_{i=1}^{n}\left(\frac{y_i}{p_i} - \widehat{\tau}_p\right)^2\right] = 1.2259$

95% Confidence Interval:

$\widehat{\mu}_p \pm t_{n-1}\sqrt{\widehat{var}(\widehat{\mu}_p)}$ = (37.5087, 41.9130)

Repeating 4)

Code: Same code used as before.

Output:

	temp	humidity	p	pi_i
208	41.1	76.02	0.001647	0.129311
154	47.7	88.63	0.001920	0.149100
129	44.9	91.07	0.001973	0.152878
388	47.7	72.90	0.001579	0.124344
325	34.5	50.20	0.001088	0.087360
..	...	...	...	...
148	31.4	70.00	0.001517	0.119703
188	32.8	46.86	0.001015	0.081789
94	41.0	72.06	0.001561	0.123003
302	16.9	45.70	0.000990	0.079847
164	20.5	45.42	0.000984	0.079377

[84 rows x 4 columns]

Unique sample count: 79

Repeating 5)

Code: Same code used as before.

Output:

mu_pi_hat: 39.49008577773485

var_hat_mu_pi_hat: 2.273551567964612

CI for HT: 36.4882266713289 42.49194488414079

Horvitz-Thompson Estimator, estimate of true population mean:

$$\hat{\mu}_{\pi} = \left[\sum_{i=1}^{v} \frac{y_i}{\pi_i}\right]/N = 39.4901$$

Estimated variance of this estimator:

$$\widehat{var}(\hat{\mu}_{\pi}) = \frac{1}{N^2}\widehat{var}(\hat{\tau}_{\pi}) = \frac{1}{N^2}\left[\sum_{i=1}^{v}\left(\frac{1}{\pi_i^2} - \frac{1}{\pi_i}\right)y_i^2 + 2\sum_{i=1}^{v}\sum_{i>j}\left(\frac{1}{\pi_i\pi_j} - \frac{1}{\pi_{ij}}\right)y_i y_j\right] = 2.2736$$

95% Confidence Interval:

$$\hat{\mu}_{\pi} \pm t_{v-1}\sqrt{\widehat{var}(\hat{\mu}_{\pi})}$$ = (36.4882, 42.4919)

Repeating 6)

Code: Same code used as before.

Output:

```
mu_g_hat:  39.0617529629169

var_hat_mu_g_hat 1.4226893070136035

CI for Gen Est:  36.687139456530076 41.43636646930372
```

Generalized Unequal Probability Estimator, estimate of true population mean:

$$\hat{\mu}_{g} = \left[\sum_{i=1}^{v} \frac{y_i}{\pi_i} \Big/ \sum_{i=1}^{v} \frac{1}{\pi_i}\right] = 39.0618$$

Estimated variance of this estimator:

$$\widehat{var}(\hat{\mu}_{g}) = \frac{1}{N^2}\left[\sum_{i=1}^{v}\left(\frac{1-\pi_i}{\pi_i^2}\right)(y_i - \hat{\mu}_g)^2 + \sum_{i=1}^{v}\sum_{i\neq j}\left(\frac{\pi_{ij} - \pi_i\pi_j}{\pi_i\pi_j}\right)\left(\frac{(y_i - \hat{\mu}_g)(y_j - \hat{\mu}_g)}{\pi_{ij}}\right)\right] = 1.4227$$

95% Confidence Interval:

$\hat{\mu}_g \pm t_{v-1}\sqrt{\widehat{var}(\hat{\mu}_g)} = (36.6871, 41.4364)$

Repeating 7)

For humidity, the best estimator in terms of estimated variance (and width of confidence intervals) is the Hansen-Hurwitz Estimator. Similarly to dew, the Generalized Unequal Probability Estimator is close, but not nearly as precise as the Hansen-Hurwitz Estimator. Lastly, the HT Estimator performed the worst with the largest estimated variance and CI width.

Repeating 8)

Calculations for true variance:

$$var(\hat{\mu}_p) = \frac{1}{N^2} var(\hat{\tau}_p) = \frac{1}{N^2}\left[(\frac{1}{n})\sum_{i=1}^{N} p_i\left(\frac{y_i}{p_i} - \tau\right)^2\right] = 1.3055$$

$$var(\hat{\mu}_\pi) = \frac{1}{N^2} var(\hat{\tau}_\pi) = \frac{1}{N^2}\left[\sum_{i=1}^{N}\left(\frac{1-\pi_i}{\pi_i}\right)y_i^{\,2} + \sum_{i=1}\sum_{j\neq i}^{N}\left(\frac{\pi_{ij}-\pi_i\pi_j}{\pi_i\pi_j}\right)y_i y_j\right] = 10.3622$$

$$var(\hat{\mu}_g) = \frac{1}{N^2}\left[\sum_{i=1}^{N}\left(\frac{1-\pi_i}{\pi_i}\right)(y_i - \mu)^2 + \sum_{i=1}\sum_{j\neq i}^{N}\left(\frac{\pi_{ij}-\pi_i\pi_j}{\pi_i\pi_j}\right)(y_i - \mu)(y_j - \mu)\right] = 1.1476$$

Based on the calculations for the true variances of the estimators using humidity, our choice of best estimator would change because the true variance for the Generalized Unequal Probability Estimator is the smallest of 1.1476. The next smallest true variance is the Hansen-Hurwitz Estimator. The largest true variance is the Horvitz-Thompson Estimator. The Horvitz-Thompson Estimator also had the largest estimated variance for humidity.

Code: Same code used as before.

Output:

True var HH: 1.3054954375129453

True var HT: 10.362249584653815

True var Gen Est: 1.1475798929555543

10. Which variable out of these two you prefer for unequal probability design and why? Make your selection based on estimation only. (forget step 8).

-**Dew HH** CI: (36.3693, 41.0538), Estimated Variance: 1.3868

-**Dew HT** CI: (37.2304, 42.8897), Estimated Variance: 2.0218

-**Dew Generalized Unequal Probability Estimator** CI: (38.9954, 42.9111), Estimated Variance: 0.9679

-Humidity HH CI: (37.5087, 41.9130), Estimated Variance: 1.2259

-Humidity HT CI: (36.4882, 42.4919), Estimated Variance: 2.2736

-Humidity Generalized Unequal Probability Estimator CI: (36.6871, 41.4364), Estimated Variance: 1.4227

For the most part, the estimated variances using humidity and dew are not too drastically different. However, the estimators using dew seem to have slightly smaller estimated variances than those using humidity as shown above. Therefore, we would prefer using the dew variable for unequal probability design since it leads to slightly more precise estimates of true average temperature with narrower confidence intervals.

11. Which estimator out of these six is best for your data? Make your selection based on estimation only (forget step 8). The best estimator should have a small variance and your parameter should be in CI.

The Generalized Unequal Probability Estimator's estimated variance using dew has the smallest estimated variance of 0.9679. The parameter (true mean temperature) is also contained within this confidence interval. The true mean temperature is 39.177 which is within the confidence interval (38.9954, 42.9111). Therefore, the Generalized Unequal Probability Estimator using dew is best for our data. This makes sense because dew is better correlated with temperature, compared to humidity.

12. Show all formulas that you use for each step and code.

Formulas, code, and output attached for each problem above.