Analysis of Water Temperature in the Bay (1989-2019)

Group Members: Amar Anand, Betel Mekonnen, Garret Shen, Franciszek Pajak

December 8th, 2024

STAT 440

Natalia Tchetcherina

Tables of Content

Introduction

Our dataset, titled “Water Quality Data for the Refuge”, was sourced from the U.S. Fish and Wildlife Service, a bureau of the U.S. Department of the Interior. We sourced our data from Data.gov. The CSV file containing our data has samples showing water quality indicators, including turbidity, pH, dissolved oxygen, salinity, and temperature. This data was collected from multiple water bodies in the Back Bay National Wildlife Refuge in Virginia Beach, Virginia. These bodies of water included the Bay, D-Pool, C-Pool, B-Pool, and A-Pool. Volunteers conducted bi-weekly sampling over three decades, spanning from 1989 to 2019.

Our population was the measurements of water temperature from the bay between 1989 and 2019 in September through May. Using Python, we filtered out observations with missing data and removed outliers using the IQR method, we reduced the dataset from 2372 to a population size of N=514.

The variable of interest in our study is water temperature (measured in °C). The dates when the water temperature data was collected constitute the sampling units. The population mean (μ) of water temperature was calculated as 14.227°C. To estimate this parameter, we determined an optimal sample size of n=85, with an α-level of 0.05.

Throughout this dataset, we asked the question “What is the true mean of water temperature in the Bay from 1989 to 2019 during the months of September through May?”. The goal of our study is to estimate the true mean water temperature in the Bay during the months of September through May. Our aim is to understand ecological patterns and support climate research and marine conservation. We understand that water temperature is crucial for several ecological and environmental concerns so we knew it was important to look into water temperature. Water temperature significantly influences fish spawning cycles and the flora and fauna in the Bay, so we knew our findings would be important to maintaining the health of a marine ecosystem. Although we didn’t study how the average water temperature in the Bay has changed over time, our findings could also be important in monitoring climate change. Historical water temperature data provides valuable insights into long-term climate trends and informs policy-making for mitigation and adaptation strategies.

Parameter Estimation

Throughout this project, we used multiple estimators to find the mean of water temperature in the Bay. We ended up choosing 4 of our best estimators, which in this case are Systematic Sampling (5-in-30, Ordered Population), Ratio Estimator with Double Sampling (SRS), Regression Estimator (SRS) with Air Temperature as an Auxiliary Variable, and Stratified Random Sampling with Proportional Allocation. These four estimators were chosen because they had the lowest estimated variance for our parameter. Our estimators all used a sample size of 85 and a significance level of 0.05.

Top Four Estimators

Estimator 1: Systematic Sampling (5-in-30, Ordered)

- **Sampling Design**: Ordered population, systematic selection of primary units.
- **Formula**: $\hat{\mu} = \frac{1}{M} \cdot \frac{N}{n} \cdot \sum_{i=1}^{n} \sum_{j=1}^{M_i} y_{ij}$

$$v\hat{a}r(\hat{\mu}) = \frac{1}{M^2} \cdot N \cdot (N-n) \cdot \frac{\frac{1}{n-1} \cdot \sum_{i=1}^{n} (y_i - \bar{y})^2}{n}$$

n is the number of primary units in the sample

Confidence Interval: $\hat{\mu} \pm t_{n-1} \cdot \sqrt{v\hat{a}r(\hat{\mu})}$

Note: In the formulas above N is the number of primary units in the population and y_ij denotes the variable of interest in the jth secondary unit of the ith primary unit. y_i is the total for the ith primary unit. M is the total number of secondary units in the population.

- **Estimated Variance**: 0.022
- **Confidence Interval**: [13.723, 14.309]
- **Rationale**: This estimator produces the minimum estimated variance while retaining the true parameter within the confidence interval.

Assumptions

The assumptions behind this estimator are that the sample must be representative of the population. Also, there may be no periodic patterns that align with our sampling interval. This assumption was met as we ordered the population with respect to our variable of interest (water temperature). By ordering the population and repeating a randomly selected pattern, we capture a sample that is representative of the population since we collect data from the parts of the population with high water temperature and low water temperature.

Estimator 2: Ratio Estimator with Double Sampling (SRS)

For this estimator, we used double sampling with simple random sampling. In our first phase, we picked primary units constructed by stratification using air temperature (measured in Fahrenheit) as the auxiliary variable. We first checked the linear relationship between our variable of interest (water temperature) and our auxiliary variable (air temperature). The correlation coefficient between air temperature and water temperature was 0.901, indicating a strong linear relationship. With an R-squared value of 0.859 and adjusted R-squared value of 0.857, tell us that a large proportion of the variance in our water temperature variable can be explained by the auxiliary variable, air temperature.

After confirming our two variables have a good linear relationship we implemented a double sampling method with simple random sampling (SRS). In double sampling, the first step was to take a sample to collect auxiliary information, and the second step was to collect data for both the auxiliary variable and the variable of interest. For our analysis, we conducted the first phase with an initial sample size $n' = 2 * 85 = 170$, followed by a second-phase sample size $n=85$. Using the data collected in step one, we were able to estimate the auxiliary variable's population total. Using the data collected in step two, we computed the ratio estimator and its variance.

Assumptions

The ratio estimator relies on the assumption of a linear relationship between the auxiliary variable (air temperature, x) and the variable of interest (water temperature, y). We confirmed this relationship by performing a regression analysis between the two variables. The regression analysis showed that both p-values for the intercept and slope were <0.001. These results aren't ideal for utilizing ratio estimators because although the second p-value for slope indicates a linear relationship between the two variables, the first p-value indicates that the line of regression does not go through the origin, which is not ideal when using the ratio estimator.

Formulas

$$r = \frac{\sum_{i=1}^{n} y_i}{\sum_{i=1}^{n} x_i}$$

$$\widehat{\mu_x} = \frac{1}{n'} \sum_{i=1}^{n'} x_i$$

$$\widehat{\mu_r} = r\widehat{\mu_x}$$

$$\widehat{var}(\widehat{T_r}) = N(N - n')\frac{s^2}{n'} + N^2 \frac{n'-n}{n'n(n-1)} \sum_{i=1}^{n} (y_i - rx_i)^2$$

$$\widehat{var}(\widehat{\mu_r}) = \widehat{var}(\widehat{T_r}) / N^2$$

$\hat{\mu_r} = 14.144$

$v\hat{a}r(\hat{\mu_r}) = 0.219$

Standard Deviation: 0.46763677368785384

Estimator 3: Regression Estimator (SRS) with Air Temperature as Auxiliary Variable

For this estimator, we used simple random sampling with air temperature as the auxiliary variable to estimate water temperature. Simple random sampling involves selecting samples randomly without replacement, ensuring each element in the population has an equal probability of being chosen. While SRS is straightforward and unbiased, it can lack precision with smaller sample sizes. This is where regression estimation comes in. It improves upon SRS by incorporating auxiliary information- in our case, the auxiliary variable is air temperature. This method leverages the relationship between air and water temperatures to refine our estimates.

The population size for our study was N=514, and we selected a sample size of n=85 for both air and water temperature measurements. We chose air temperature as the auxiliary variable after confirming its high correlation with water temperature where our r value was 0.901, making it the best auxiliary variable compared to the rest of the variables in our dataframe.

Assumptions

We first conducted a random sampling without replacement for both variables and then performed a diagnostic analysis to verify the linear relationship between air and water temperature. Our R squared value is 0.770 with an adjusted R squared value of 0.767, demonstrating that a large proportion of the variance within water temperature in our model can be explained by the auxiliary variable. In our regression analysis the p-value for slope of <0.0001 is significantly below the significance level of 0.05 allowing us to reject the null hypothesis that there exists no linear relationship between the two variables, confirming that a linear relationship exists. Also the p-value for the intercept was <0.0001, allowing us to reject the null hypothesis that the intercept is close to 0. This indicates that the regression estimator could be a good choice since the intercept is non-zero and there's a linear relationship between the two variables.

Using the regression estimation method, we calculate the estimated water temperature mean as 14.48, with a variance of 0.078. After producing the 95% confidence interval [13.823, 14.936], we observed our true mean (14.227) falling well within the confidence interval, further confirming the accuracy of the regression estimator. By using auxiliary information, regression estimation significantly reduces variance compared to SRS alone. And as you can see the low variance and narrow confidence interval make it a highly reliable method.

Formula:

$$\hat{\mu}_L = a + b\mu_x \qquad a = \overline{y} - b\overline{x} \qquad b = \frac{\sum_{i=1}^{n}(x_i-\overline{x})(y_i-\overline{y})}{\sum_{i=1}^{n}(x_i-\overline{x})^2}$$

$$\widehat{var}(\widehat{\mu_L}) = \left(\sum_{i=1}^{N}(y_i - a - bx_i)^2\right) * \frac{(N-n)}{(N*n*(n-2))}$$

95% CI: $\widehat{\mu_L} \pm t_{0.025,n-2}\sqrt{\widehat{var}(\widehat{\mu_L})}$

OLS Regression Results

Dep. Variable:	WaterTemp	**R-squared:**	0.770
Model:	OLS	**Adj. R-squared:**	0.767
Method:	Least Squares	**F-statistic:**	277.7
Date:	Sun, 10 Nov 2024	**Prob (F-statistic):**	3.27e-28
Time:	22:07:14	**Log-Likelihood:**	-207.86
No. Observations:	85	**AIC:**	419.7
Df Residuals:	83	**BIC:**	424.6
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	-12.3411	1.657	-7.449	0.000	-15.636	-9.046
AirTemp	0.4710	0.028	16.665	0.000	0.415	0.527

Omnibus:	26.890	**Durbin-Watson:**	1.417
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	68.374
Skew:	1.037	**Prob(JB):**	1.42e-15
Kurtosis:	6.873	**Cond. No.**	317.

Regression estimator $\hat{\mu}_L$ = 14.38
Estimated Variance of Regression Estimator: 0.078
Confidence Interval: (13.823, 14.936)

Estimator 4: Stratified Random Sampling with Proportional Allocation

For this estimator, we divided our population into five strata, based on the auxiliary variable air temperature. To determine the optimal stratification, we evaluated three different sets of boundaries for air temperature and calculated their delta values, which measure the improvement in variance reduction achieved by stratification. The options were as follows:

- Option 1: 5 strata, <40, 40-50, 50-60, 60-70, >=70
- Option 2: 4 strata, <40, 40-55, 55-70, >=70
- Option 3: 3 strata, <40, 45-65, (>=70 and 40-45)

Deltas 1: 79405
Deltas 2: 78438
Deltas 3: 70536

After calculating deltas for each of the three strata, we picked the stratification method which resulted in the highest delta, this was option one in this case. The deltas help provide us with the stratification option that provides the lowest variability within strata. The principle of stratification helps us achieve this through its process of dividing the population into smaller homogeneous subgroups. Using stratification option 1, the stratum sizes (Nh) of the five strats were: 46, 120, 143, 109, 96. We then used proportional allocation to ensure a consistent sampling ratio across the strata. We then calculated the sample sizes nh as: 7, 20, 24, 18, 16.

Assumptions

When using stratified random sampling, we made sure that each stratum was different but within each stratum, the units were as similar as possible. The delta values guided this process, as the reflect the improvement in reducing variance achieved by stratification. By selecting the stratification with the highest delta, we minimize within-stratum variance, which is in line with the stratification principle.

Formula:

$$\Delta = (N-1)\sigma^2 - \sum_{h=1}^{L}(N_h - 1)\,\sigma_h^{\,2}$$

$$\sigma^2 = \frac{1}{N-1}\sum_{i=1}^{N}(y_i - \mu)^2$$

$$\sigma_h^{\,2} = \frac{1}{N_h-1}\sum_{i=1}^{N_h}(y_{hi} - \mu_h)^2$$

$$n_h = \frac{nN_h}{N}$$

$$\bar{y}_{st} = \frac{1}{N}\sum_{h=1}^{L} N_h\bar{y}_h$$

$$\bar{y}_h = \frac{1}{n_h}\sum_{i=1}^{n_h} y_{hi}$$

$$\widehat{var}(\bar{y}_{st}) = \sum_{h=1}^{L}\left(\frac{N_h}{N}\right)^2\left(\frac{N_h - n_h}{N_h}\right)\frac{s_h^2}{n_h}$$

$$d = \left(\sum_{h=1}^{L} a_h s_h^{\,2}\right)^2 / \left[\sum_{h=1}^{L}(a_h s_h^{\,2})^2/(n_h - 1)\right]$$

$$a_h = N_h(N_h - n_h)/n_h$$

$$s_h^2 = \frac{1}{n_h-1}\sum_{i=1}^{n_h}(y_{hi} - \bar{y}_h)^2$$

Confidence Interval = $\bar{y}_{st} \pm t_d\sqrt{\widehat{var}(\bar{y}_{st})}$

$\overline{y}_{st} = 14.543$
$v\hat{a}r(\overline{y}_{st}) = 0.075$

```
Degrees of Freedom:  71.78513640532482
95% CI: {13.996} {15.09}
```

Summary of Results

Name of Estimator	Mean	Var	SD	95% C.I.
Systematic Sampling (5-in-30, Ordered)	14.016	0.022	0.148	[13.723, 14.309]
Ratio Estimator w/ double sampling SRS (Air temp)	14.144	0.219	0.468	
Regression Estimator w/ SRS (Air Temp)	14.38	0.078	0.279	[13.823, 14.936]
Stratified Random Sampling w/ Proportional Allocation	14.543	0.075	0.274	[13.996, 15.09]

Conclusion

Best Estimator: Systematic Sampling (5-in-30, Ordered)

The best estimator, overall, is the Systematic Sampling (5-in-30, Ordered) estimator. It is the best estimator due to it having the lowest estimated variance of 0.022, when compared to the rest. The average temperature in the Back Bay ecosystem calculated by our estimator of 14.016 was relatively close to the true mean temperature of 14.227. In addition, you can also look at our estimator's confidence interval and notice that the true mean of 14.227 is contained within this estimator’s confidence interval. It is true that the other estimators also have small variances and encapsulate the true mean in their confidence intervals but the systematic sampling estimator has the smallest variance and the narrowest confidence interval.

Interpretation

Overall, we believe we arrived at a good estimator. Our best estimator used the auxiliary variable air temperature to stratify the data. We chose this variable as opposed to a variable like dissolved oxygen because it had a higher correlation coefficient with water temperature. When we chose to use other auxiliary variables such as dissolved oxygen in our other estimators, the resulting estimators had significantly higher estimated variances. Studying water temperature in an aquatic ecosystem is important since it affects environmental factors such as how much dissolved oxygen water can hold. For example, warmer water holds less dissolved oxygen than cold water. Overall, our results give us some valuable insight into the Back Bay and Chesapeake Bay ecosystems. It is well known that the Chesapeake bay is quite shallow when compared to other Bays in the United States and its average depth is around 21 feet. This should naturally result in a high variance of temperatures as shallow bodies of water can heat up quicker than deeper ones. The wide range of our variable of interest works well with this hypothesis that the bay should display relatively high variances in temperature. Understanding these seasonal changes in temperature is important when analyzing fish spawning cycles. This data can inform conservation strategies for species like Striped Bass and American Shad which spawn in the Chesapeake. Striped Bass begin spawning when water temperatures reach 60 degrees Fahrenheit and it’s estimated that between 70 and 90 percent of the Altlatic coast Striped Bass are nursed in the Chesapeake. Ultimately, our findings can be used to assess warming trends in aquatic ecosystems to develop strategies to counteract climate-induced changes in the Bay.

References

U.S. Environmental Protection Agency. *Water Quality Data*. Retrieved from https://catalog.data.gov/dataset/water-quality-data-41c5e

U.S. Fish and Wildlife Service. *Big Branch Marsh National Wildlife Refuge Visitor Map*. Retrieved from https://www.fws.gov/sites/default/files/documents/BBNWR_Visitor%20Map%28Accessibility-Checked%29.pdf

Striped Bass Information

https://dnr.maryland.gov/fisheries/Pages/fish-facts.aspx?fishname=Striped+Bass#:~:text=Chesapeake%20Bay%20and%20its%20tributaries,%E2%80%8B

Appendix

Estimator 1 Code:

```
np.random.seed(42)
# Parameters
M = 514 #Total number of units in the population
needed_sample_size = 85 #Total sample size we need
n = 5 # Sample size of primary units
N = 30 # Number of Primary Units in the population

import random
import statistics
# Generate initial sample of 5 random indices between 0 and 29 inclusive
random_indices1 = random.sample(list(range(0, N)), n)

# Generate a larger sample by adding 30, 17 times to each initial sample
index
larger_sample1 = []
j = 0
while j <= 510:
    for index in random_indices1:
        if j + index >= 514 :
          break;
        else:
          larger_sample1.extend([j + index])
    j += 30

#The lists below contain the primary units I declare but don't initialize
them.
list_1__ = []
list_2__ = []
list_3__ = []
list_4__ = []
list_5__ = []
#Instantiate the primary units
j = 0
```

```python
while j <= 510:
    for k in range(5):
        if j + random_indices1[k] >= 514 :
          break;
        else:
          if k == 0:
            list_1__.extend([j + random_indices1[k]])
          elif k == 1:
            list_2__.extend([j + random_indices1[k]])
          elif k == 2:
            list_3__.extend([j + random_indices1[k]])
          elif k == 3:
            list_4__.extend([j + random_indices1[k]])
          else:
            list_5__.extend([j + random_indices1[k]])
    j += 30


list_of_all__ = df["Water Temp (?C)"].iloc[list(range(514))]
list_of_all__ = list_of_all__.sort_values()


list_1__new = list()
list_2__new = list()
list_3__new = list()
list_4__new = list()
list_5__new = list()
for i in list_1__:
  list_1__new.append(list_of_all__.iloc[i])
for i in list_2__:
  list_2__new.append(list_of_all__.iloc[i])
for i in list_3__:
  list_3__new.append(list_of_all__.iloc[i])
for i in list_4__:
  list_4__new.append(list_of_all__.iloc[i])
for i in list_5__:
  list_5__new.append(list_of_all__.iloc[i])

primary_unit_totals1 = [sum(list_1__new), sum(list_2__new),
sum(list_3__new), sum(list_4__new), sum(list_5__new)]
```

```
#Estimated parameter of interest is below
mu_hat_1 =
((N/n)*((sum(list_1__new))+(sum(list_2__new))+(sum(list_3__new))+(sum(list
_4__new))+(sum(list_5__new))))/M

display(Math(r'{\hat{\mu}} = ' + str(round(mu_hat_1, 3))))

#Estimated Variance of the parameter of Interest
s_u_squared1 = 0
for y_i in primary_unit_totals1:
  s_u_squared1 += (y_i - statistics.mean(primary_unit_totals1)) ** 2
s_u_squared1 = s_u_squared1 / (n - 1)

var_hat_mu_hat1 = (1/(M ** 2))*((N*(N - n)*s_u_squared1)/(n))

display(Math(r'{\hat{var}}({\hat{\mu}}) = ' + str(round(var_hat_mu_hat1,
3))))

#The alpha level chosen in report 2 was 0.95
#A 95% CI for my estimator is [13.815, 15.01]

# First get the t-distribution crictal value
alpha = 0.05
t_critical_95 = t.ppf(1 - alpha / 2, df = len(larger_sample1) - 1)

# Calculating the lower and upper bounds of the confidence interval
lower_95_CI = round(mu_hat_1 - (t_critical_95 * np.sqrt(var_hat_mu_hat1)),
3)
upper_95_CI = round(mu_hat_1 + (t_critical_95 * np.sqrt(var_hat_mu_hat1)),
3)

print("95% CI:", {lower_95_CI}, {upper_95_CI})
```

Estimator 2 Code:

```
import random
import statistics
N = 514
n = 85
mu = 14.227 #True Parameter
```

```
# Auxilary Variable chosen: Air Temperature

np.random.seed(42)

#Double Sampling with SRS
first_phase_sample = random.sample(list(range(0, N)), n*2)
second_phase_sample = random.sample(first_phase_sample, n)

#sample = np.random.choice(population, size=n, replace=False)

# Performing random sampling (without replacement) for Water Temp
sample_water_temp = df["Water Temp (?C)"].iloc[second_phase_sample]


# Performing random sampling (without replacement) for Air Temp
sample_air_temp = df["Air Temp (?F)"].iloc[second_phase_sample]
print(sample_water_temp)
print(sample_air_temp)
data_thing = pd.DataFrame({
    "WaterTemp": sample_water_temp,
    "AirTemp": sample_air_temp
})
#Since both the p-values are <0.05 they are statistically significant and
therefore
#x and y have a linear relationship and the fitted line doesn't go through
the origin

#As a result of the fitted line not going through the origin I would
recommend to
#Use linear regression estimator instead of ratio estimator

full_list = list(range(0, N))
full_df = df['Air Temp (?F)'].iloc[full_list].copy()
full_df_ = list()
for i, value in enumerate(full_df):
  full_df_.append(value)
ful = df["Water Temp (?C)"].iloc[full_list].copy()
ful_ = list()
for i, value in enumerate(ful):
```

```
  ful_.append(value)
top = 0
for x in second_phase_sample:
  top = top + ful_[x]
bottom = 0
for x in second_phase_sample:
  bottom = bottom + full_df_[x]
r = top / bottom
tau_hat_x = 0
for x in first_phase_sample:
  tau_hat_x = tau_hat_x + full_df_[x]
tau_hat_x = tau_hat_x*(N/(n*2))

#Estimate parameter of interest by ratio estimator
mu_hat_r = (r*tau_hat_x)/N
display(Math(r'{\hat{\mu_{r}}} = ' + str(round(mu_hat_r, 3))))

#Estimating variance
data = list()
for x in second_phase_sample:
  data.append(ful_[x])
s_squared = statistics.variance(data)

var_hat_mu_hat_r = 0
for x in second_phase_sample:
  var_hat_mu_hat_r = var_hat_mu_hat_r + ((ful_[x] - r*full_df_[x]) ** 2)
var_hat_mu_hat_r = ((N**2)*((2*n - n)/((2*n)*n*(n - 1))))*var_hat_mu_hat_r
var_hat_mu_hat_r = var_hat_mu_hat_r + N*(N - 2*n)*s_squared/(2*n)
var_hat_mu_hat_r = (1/(N**2))*var_hat_mu_hat_r
display(Math(r'{\hat{var}}({\hat{\mu_{r}}}) = ' +
str(round(var_hat_mu_hat_r, 3))))
print("Standard Deviation: ", math.sqrt(var_hat_mu_hat_r))
```

Estimator 3 Code:

```
# The value of the parameter

N = 514
n = 85
# Auxilary Variable chosen: Air Temperature
```

```
np.random.seed(42)

population = np.arange(0, N)

sample = np.random.choice(population, size=n, replace=False)

# Performing random sampling (without replacement) for Water Temp
sample_water_temp = df["Water Temp (?C)"].iloc[sample]


# Performing random sampling (without replacement) for Air Temp
sample_air_temp = df["Air Temp (?F)"].iloc[sample]
```

```
T_x = sum(df["Air Temp (?F)"])
mu_x = T_x / N

x_bar = np.mean(sample_air_temp)
y_bar = np.mean(sample_water_temp)

# ASSUME LENGTH X = LENGTH Y
b = sum((sample_air_temp - x_bar) * (sample_water_temp - y_bar))
b /= sum((sample_air_temp - x_bar) ** 2)
a = y_bar - (b * x_bar)

mu_hat_l = a + (b * mu_x)

print("Regression estimator =", round(mu_hat_l, 3))
# Estimation of Variance of Regression Esitmator: var_hat_mu_hat_l = ((N -
n) / (N * n * (n - 2))) * np.sum((y - a - b * x) ** 2)

var_hat_mu_hat_l = ((N - n) / (N * n * (n - 2))) *
np.sum((sample_water_temp - a - b * sample_air_temp) ** 2)

print("Estimation of Variance of Regression Esitmator =",
round(var_hat_mu_hat_l, 3))
# Confidence Inteval = (mu_hat_r - t_value(sqrt(var_hat_mu_hat_l),
mu_hat_r + t_value(sqrt(var_hat_mu_hat_l))
alpha = 0.05
```

```
t_critical_value = t.ppf(1 - alpha / 2, df= n - 2)

lower_bound = mu_hat_l - t_critical_value * (math.sqrt(var_hat_mu_hat_l))

upper_bound = mu_hat_l + t_critical_value * (math.sqrt(var_hat_mu_hat_l))

print("Confidence Inteval (for alpha = 0.05) =", (round(lower_bound, 3),
round(upper_bound, 3)))
```

Estimator 4 Code:

```
# True Parameter

import random

import statistics



mu = 14.227



# Population Size

N = 514



# Total Sample Size

n = 85

# Random sampling

np.random.seed(419)



# Get the data

full_list = list(range(0, N))

full_df = df['Air Temp (?F)'].iloc[full_list].copy()
```

```
ful = df["Water Temp (?C)"].iloc[full_list].copy()

# Stratify the sample using Air Temperature


list_1 = list()

list_2 = list()

list_3 = list()

list_4 = list()

list_5 = list()


for i, value in enumerate(full_df):

    original_index = df.index[full_list[i]]  # Get the original index

    #print(f"Original Index: {original_index}, Value: {value}")

    if value < 40 :

    #Append list_1

      list_1.append(original_index)

    elif value >= 40 and value < 50 :

    #Append list_2

      list_2.append(original_index)

    elif value >= 50 and value < 60 :

    #Append list_3

      list_3.append(original_index)
```

```
    elif value >= 60 and value < 70 :

    #Append list_4

      list_4.append(original_index)

    else:

    #Append list_5

      list_5.append(original_index)


#We will acess stuff in the lists this way

#value = df.loc[original_index, 'Water Temp (?C)']

list_of_lists_1 = list()


to_add = list()

for x in list_1:

  value = df.loc[x, 'Water Temp (?C)']

  to_add.append(value)

list_of_lists_1.append(to_add)

to_add = list()

for x in list_2:

  value = df.loc[x, 'Water Temp (?C)']

  to_add.append(value)

list_of_lists_1.append(to_add)

to_add = list()

for x in list_3:
```

```
  value = df.loc[x, 'Water Temp (?C)']

  to_add.append(value)

list_of_lists_1.append(to_add)

to_add = list()

for x in list_4:

  value = df.loc[x, 'Water Temp (?C)']

  to_add.append(value)

list_of_lists_1.append(to_add)

to_add = list()

for x in list_5:

  value = df.loc[x, 'Water Temp (?C)']

  to_add.append(value)

list_of_lists_1.append(to_add)

print("First way of stratifying the population: ")

print("I used 5 groups. The Nh's are below")

counter = 1

cou = 0

for g in list_of_lists_1:

  print("N_", counter, " : ", len(g))

  counter  = counter + 1

#Calculate Δ for the first method of stratification



#Obtaining full population
```

```
fulll = list()
to__add = list()
for i, value in enumerate(full_df):
  original_index = df.index[full_list[i]]  # Get the original index
  to__add.append(value)
fulll.append(to__add)

full_var = 0
for lis in fulll:
  delta_toadd = 0
  for r in lis:
    delta_toadd = delta_toadd + ((r - np.mean(lis))**2)
  full_var = full_var + (delta_toadd)

delta_1 = full_var
for lis in list_of_lists_1:
  delta_toadd = 0
  for r in lis:
    delta_toadd = delta_toadd + ((r - np.mean(lis))**2)
  delta_1 = delta_1 - (delta_toadd)
#Stratified Random Sample with Proportional allocation
print("Stratified Random Sample with Proportional allocation")
```

```
print("The nh's are below")

list_of_nhs = list()

counter = 1

for g in list_of_lists_1:

  top = int(round((n * len(g)) / N))

  list_of_nhs.append(top)

  print("n_",counter, " : ", top, "rounded from: ", ((n * len(g)) / N))

  counter  = counter + 1

print("Since n_1 was rounded the most and the sum of the n_h's is 86 and
is not equal to n. I manually adjust n_1 to be 7")

list_of_nhs[0] = 7



counter = 1

for g in list_of_nhs:

  print("n_",counter, " : ", g)

  counter  = counter + 1

#Generate the indices of the random sample

random_indices = list()

for g in range(0, 5):

  gg = random.sample(list(range(0, len(list_of_lists_1[g]))),
list_of_nhs[g])

  random_indices.append(gg)

y_bar_h = list()

for g in range(0, 5):
```

```
  y_bar = 0

  for x in random_indices[g]:

    y_bar = y_bar + list_of_lists_1[g][x]

  y_bar = y_bar / list_of_nhs[g]

  y_bar_h.append(y_bar)

s_squared_h = list()

for g in range(0, 5):

  to_addd = 0

  for x in random_indices[g]:

    to_addd = to_addd + ((list_of_lists_1[g][x] - y_bar_h[g]) ** 2)

  to_addd = to_addd / (list_of_nhs[g] - 1)

  s_squared_h.append(to_addd)

#Estimate the parameter of interest

tau_hat_st = 0

for g in range(0, 5):

  tau_hat_st = tau_hat_st + (len(list_of_lists_1[g]))*(y_bar_h[g])

mu_hat_st = tau_hat_st / N

display(Math(r'{\hat{\mu_{st}}} = ' + str(round(mu_hat_st, 3))))

#Estimated variance of the parameter of interest

var_hat_tau_hat_st = 0

for g in range(0, 5):

  cap = (len(list_of_lists_1[g]))*(len(list_of_lists_1[g]) -
list_of_nhs[g])*((s_squared_h[g])/(list_of_nhs[g]))

  var_hat_tau_hat_st = var_hat_tau_hat_st + cap
```

```
var_hat_mu_hat_st = var_hat_tau_hat_st / (N ** 2)

display(Math(r'{\hat{var}}({\hat{\mu_{st}}}) = ' +
str(round(var_hat_mu_hat_st, 3))))



#We will use Satterthwaite formula for adjusted degrees of freedom

number_1 = 0

number_2 = 0

for g in range(0, 5):

  number_1 = number_1 + (len(list_of_lists_1[g]))*(len(list_of_lists_1[g])
- list_of_nhs[g])*((s_squared_h[g])/(list_of_nhs[g]))

  number_2 = number_2 +
((((len(list_of_lists_1[g]))*(len(list_of_lists_1[g]) -
list_of_nhs[g])*((s_squared_h[g])/(list_of_nhs[g]))) ** 2) /
(list_of_nhs[g] - 1))

degrees_of_freedom = (number_1 ** 2) / number_2

print("Degrees of Freedom: ", degrees_of_freedom)

#The alpha level chosen in report 2 was 0.95

#A 95% CI for my estimator is below



# First get the t-distribution crictal value

alpha = 0.05

t_critical_95 = t.ppf(1 - alpha / 2, df = degrees_of_freedom)



# Calculating the lower and upper bounds of the confidence interval

lower_95_CI = round(mu_hat_st - (t_critical_95 *
np.sqrt(var_hat_mu_hat_st)), 3)
```

```
upper_95_CI = round(mu_hat_st + (t_critical_95 *
np.sqrt(var_hat_mu_hat_st)), 3)


print("95% CI:", {lower_95_CI}, {upper_95_CI})
```