



Strengths:

Amazing EDA analysis!!

```
[ ] # Simple Random Sampling
def random_sampling(df, sample_size=200):
    # Shuffle the DataFrame:
    df_shuffled = df.sample(frac=1).reset_index(drop=True)
    # Take random samples:
    samples = [df_shuffled.sample(n=sample_size) for _ in range(3)]
    return samples
```

Good idea to shuffle the data in random sampling!

Random Sampling Consistency

The histograms above show the distribution of the 'Field_area' for the three random samples! While there are slight variations in each sample, they broadly exhibit a similar distribution pattern. Therefore, it suggests that the random sampling method is reasonably consistent across different executions.

First sample is missing 'Blueberry', 'Endive', 'Hot pepper', 'Jerusalem artichoke'.

Second sample is missing 'Celery', 'Endive', 'Pepper', 'Turnip'.

Third sample is missing: 'Endive', 'Hot pepper', 'Jerusalem artichoke', 'Kiwat tomato', 'Pepper', 'Turnip', 'Watermelon'.

These differences highlight the variability that may occur with random sampling especially with crops that have fewer occurrences in the dataset.

Great interpretation! Quite detailed 😊



Leandro Machado Project: Agricultural Sampling Frames.ipynb ☆
File Edit View Insert Runtime Tools Help [Last edited on April 22](#)

+ Code + Text

| | | |
|------------|-----|--------|
| Strawberry | 3 | NaN |
| Tomato | 79 | 69.69 |
| Turnip | 1 | NaN |
| Watermelon | 1 | NaN |
| Zucchini | 118 | 106.05 |

The table above compares the actual population counts of each crop with the estimated counts derived from the first random sample, scaled to infer the total population. The scaling factor 'P' has been used to extrapolate from the sample size to the population size.

Good Estimates: Crops like Carrot, Cucumber, and Green bean show estimates that are quite close or slightly higher than the actual values, indicating a good representation in the sample.

Underestimation: Crops like Mint, Tomato, and Zucchini show lower estimated counts compared to the actual, suggesting potential under-representation in the sample.

Overestimation: For smaller count crops like Endive, Helda bean, and Jerusalem artichoke, the estimates are higher than actual counts, which can be typical in samples where every occurrence has a disproportionately high impact due to the low base count.

Missing Values (NaN): Hot pepper, Kiwat tomato, Pepper, Strawberry, Turnip, and Watermelon have missing estimates, indicating they were not present in the sample. This highlights the limitation of random sampling, especially for less common items in a dataset.


Implications:

The variances in estimated counts reflect the random nature of the sampling and the distribution of crop counts within the sample. For crops with very low counts in the overall population, random sampling can either miss them entirely or provide a skewed representation. The discrepancies and missing values call for careful consideration in relying on small random samples for population estimation, especially for diverse populations with uneven distributions.

Again, beautiful interpretation for the interference. This is the most important thing in the analysis, interpretation, and you're great at it.

Improvements:

```
# Visualize your findings
```

 Description of The Field Area:

- Average: 1.28 units
- Minimum: 0.01 units
- Maximum: 20.0 units

Distribution:

- Variance: 3.609
- Skewness: 5.05

Field are is in Hectares as mentioned in the notebook introduction, not "units".

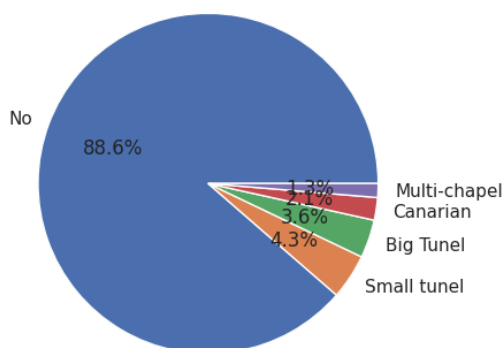


```
# 5. Who uses Greenhouses and who doesn't?
greenhouse_usage = df['Greenhouse'].value_counts()

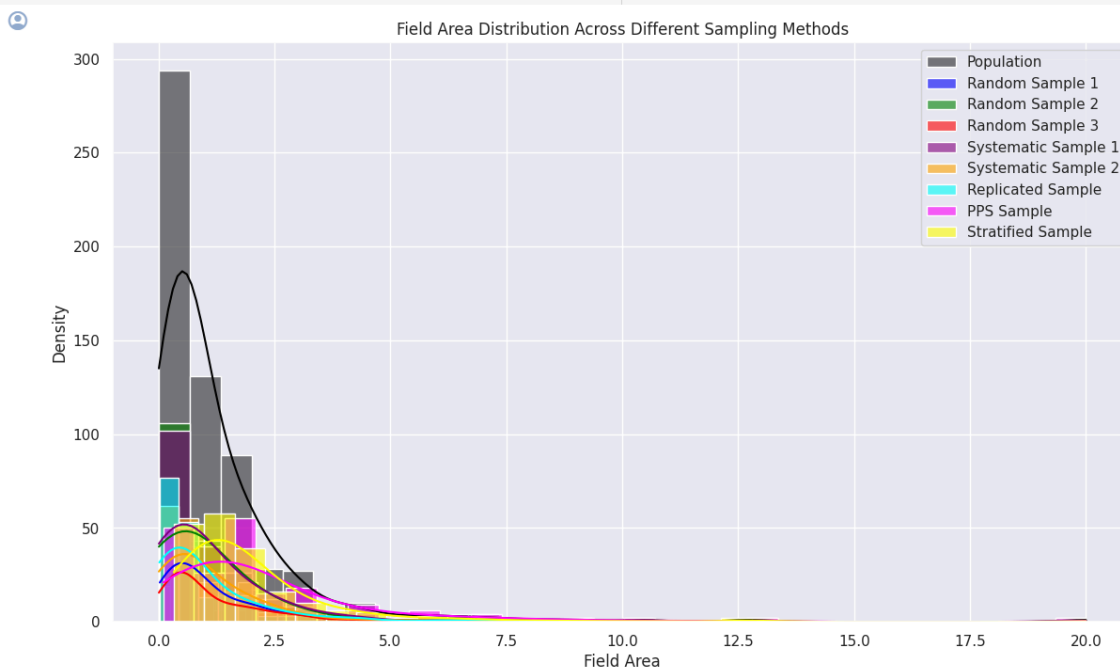
fig, ax = plt.subplots()
greenhouse_usage.plot(kind='pie', autopct='%1.1f%%', ax=ax)
ax.set_title('Greenhouse Usage')
ax.set_ylabel('')
```

Text(0, 0.5, '')

Greenhouse Usage



Pie charts are considered a bad representation in statistical analysis because you can't easily view categories or rank them in terms of proportion if you have more than 3. Bar charts are easier to view.



Beautiful chart to compare different methods, it's a little bit crowded, using one sample from each method would've been easier to read



Inference as a definition is straightforward. Create a population from each sample method. For clarity we'll provide the following example :

Since we took a random sample of 200 of the population, If we have for eg 100 farmers planting potatoes in the sample, it would probably be $100 * \frac{606}{200} = 303$ for the population, where 606 is the population size and 200 the sample size

```
# P = len(df)/len(df_random)
# Estimated_Population_Crop = df_random.Crop.value_counts() * P

-----
NameError                                Traceback (most recent call last)
<ipython-input-24-489cf34c3a5d> in <cell line: 1>()
----> 1 P = len(df)/len(df_random)
      2 Estimated_Population_Crop = df_random.Crop.value_counts() * P

NameError: name 'df_random' is not defined
```

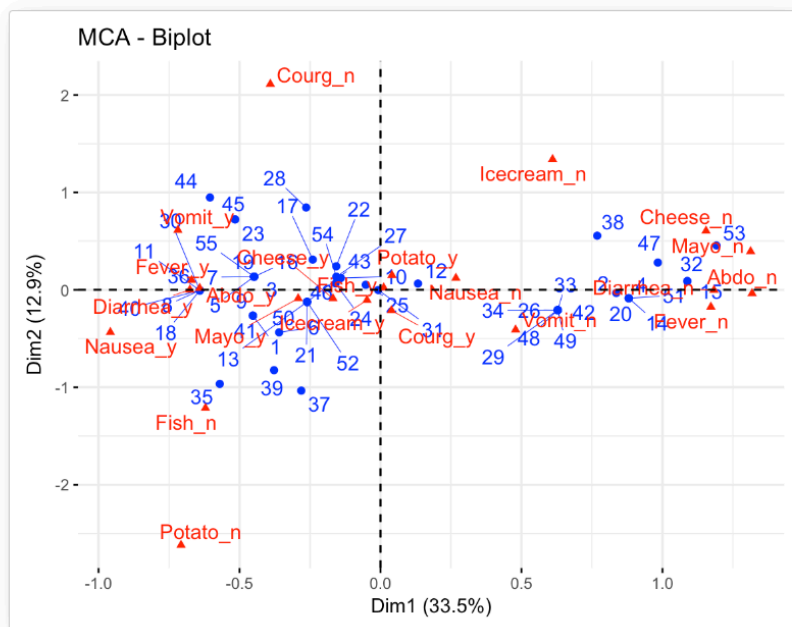
Old error in the notebook, best to clear output

1. The interpretation of the clustering can be more specific, for example you take each sub cluster and try to identify common characteristics, or you can use MCA to showcase different categories, Example of MCA using R ([Source](#)):

Biplot

The function `fviz_mca_biplot()` [factoextra package] is used to draw the biplot of individuals and variable categories:

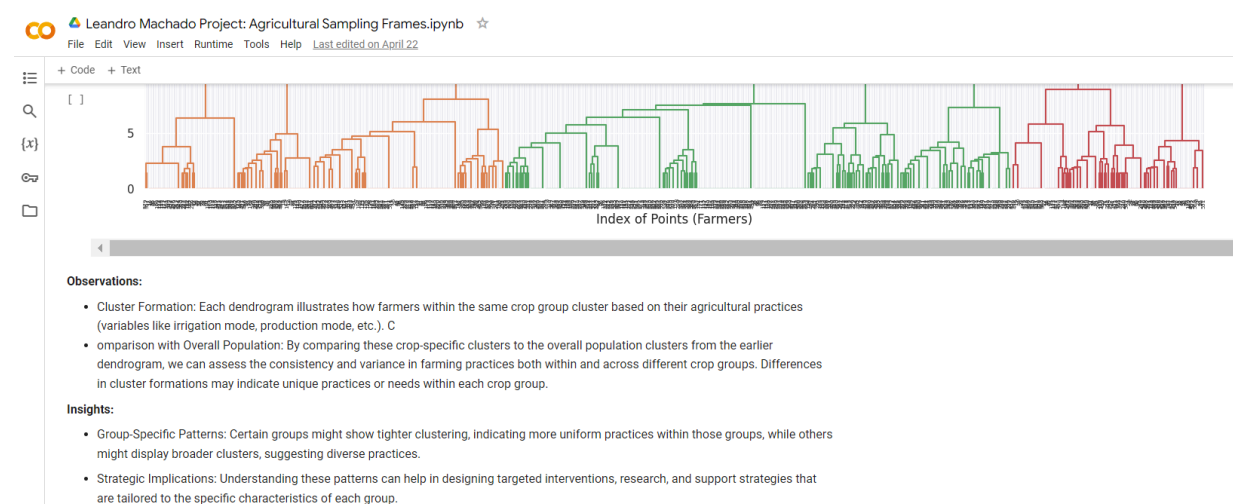
```
fviz_mca_biplot(res.mca,
                 repel = TRUE, # Avoid text overlapping (slow if many point)
                 ggtheme = theme_minimal())
```



The interpretation of the MCA analysis can be more specific, for example you take each sub cluster and try to identify common characteristics, or you can showcase



different categories, Example of MCA using R (offers better charts than Python for this specific use case) ([Source](#))



Both dendrograms look good, but they still need interpretation. Without insights about the different groups formed they are not useful.

Sampling and Inference:

- Your analysis showed great focus and attention to detail.
- We can tell you spent a lot of time on comparing them, however for the inference section, we saw you had the code to compare different methods, but the entire section only outputs one table that uses Simple Random Sampling



co

Leandro Machado Project: Agricultural Sampling Frames.ipynb

☆

File Edit View Insert Runtime Tools Help Last edited on April 22

+ Code + Text

the sampling process using the same mini sample sizes you used.

Scaling factor calculation for Random, Systematic, and Replicated samples:
P = len(df) / 200

Estimate population from samples (Random, Systematic, Replicated):
estimated_population_random = [sample['Crop'].value_counts() * P for sample in random_samples_df]
estimated_population_systematic = [sample['Crop'].value_counts() * P for sample in systematic_samples_df]
estimated_population_replicated = replicated_sample_df['Crop'].value_counts() * P

Comparing estimated populations with the actual population:
actual_population_crop = df['Crop'].value_counts()

Display estimated vs. actual for Random Sample 1 as an example:
comparison_random_sample_1 = pd.DataFrame({
 'Actual': actual_population_crop,
 'Estimated (Random Sample 1)': estimated_population_random[0]
})

comparison_random_sample_1

Actual Estimated (Random Sample 1)

Crop

| | | |
|-----------------------|----|-------|
| Absinthe | 2 | 3.03 |
| Artichoke | 3 | 3.03 |
| Blueberry | 2 | 3.03 |
| Carrot | 10 | 12.12 |
| Cauliflower | 7 | 3.03 |
| Celery | 5 | 6.06 |
| Coriander and parsley | 69 | 87.87 |
| Cucumber | 49 | 57.57 |
| Eggplant | 10 | 12.12 |
| Fennel | 1 | 3.03 |