Simple random and systematic sampling

SAMPLING IN PYTHON



James Chapman
Curriculum Manager, DataCamp

datacamp

Simple random sampling





Simple random sampling of coffees





Simple random sampling with pandas

```
coffee_ratings.sample(n=5, random_state=19000113)
```

```
     total_cup_points         variety country_of_origin  aroma  flavor  \
437             83.25            None          Colombia   7.92    7.75
285             83.83  Yellow Bourbon            Brazil   7.92    7.50
784             82.08            None          Colombia   7.50    7.42
648             82.58         Caturra          Colombia   7.58    7.50
155             84.58         Caturra          Colombia   7.42    7.67

     aftertaste  body  balance
437        7.25  7.83     7.58
285        7.33  8.17     7.50
784        7.42  7.67     7.42
648        7.42  7.67     7.42
155        7.75  8.08     7.83
```

Systematic sampling





Systematic sampling - defining the interval

```
sample_size = 5
pop_size = len(coffee_ratings)
print(pop_size)
```

```
1338
```

```
interval = pop_size // sample_size
print(interval)
```

```
267
```

Systematic sampling - selecting the rows

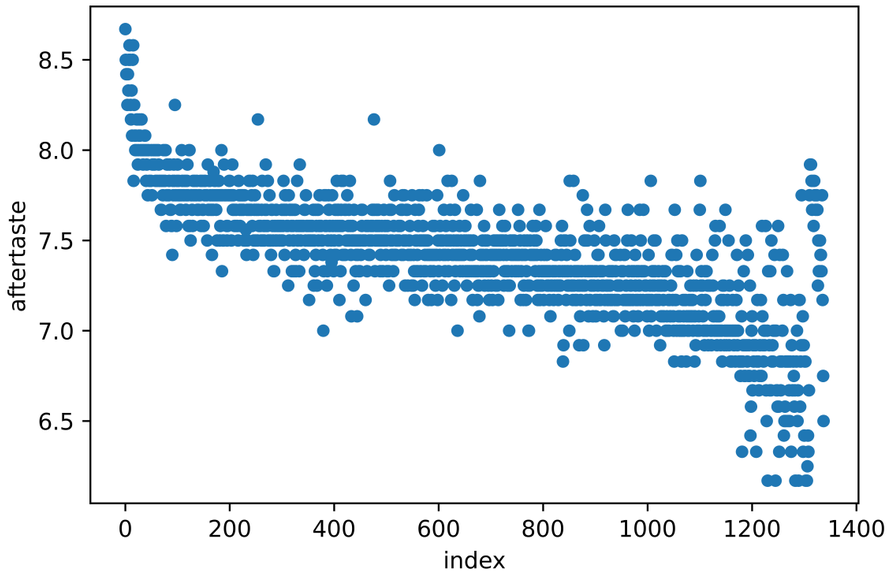
```
coffee_ratings.iloc[::interval]
```

```
      total_cup_points  variety country_of_origin  aroma  flavor  aftertaste  \
0                90.58     None          Ethiopia   8.67    8.83        8.67
267              83.92     None          Colombia   7.83    7.75        7.58
534              82.92  Bourbon       El Salvador   7.50    7.50        7.75
801              82.00   Typica            Taiwan   7.33    7.50        7.17
1068             80.50    Other            Taiwan   7.17    7.17        7.17

      body  balance
0     8.50     8.42
267   7.75     7.75
534   7.92     7.83
801   7.50     7.33
1068  7.17     7.25
```

The trouble with systematic sampling

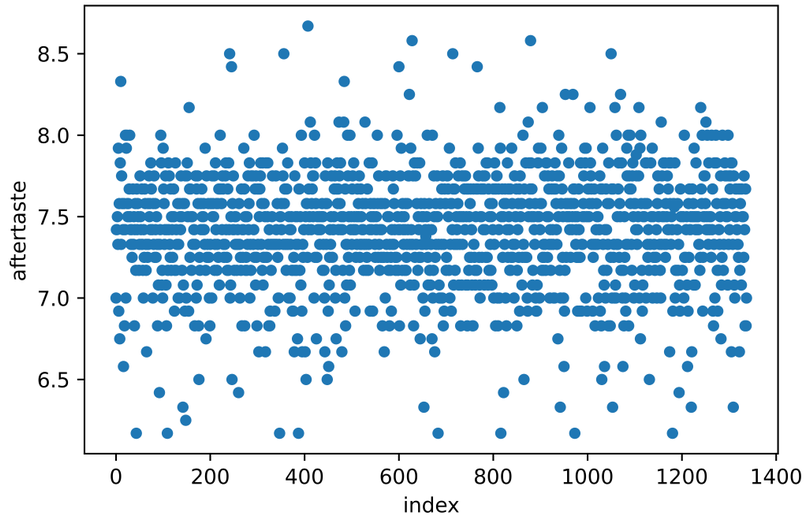
```
coffee_ratings_with_id = coffee_ratings.reset_index()
coffee_ratings_with_id.plot(x="index", y="aftertaste", kind="scatter")
plt.show()
```



Systematic sampling is only safe if we don't see a pattern in this scatter plot

Making systematic sampling safe

```
shuffled = coffee_ratings.sample(frac=1)
shuffled = shuffled.reset_index(drop=True).reset_index()
shuffled.plot(x="index", y="aftertaste", kind="scatter")
plt.show()
```



Shuffling rows + systematic sampling is the same as simple random sampling

Let's practice!

SAMPLING IN PYTHON

Stratified and weighted random sampling

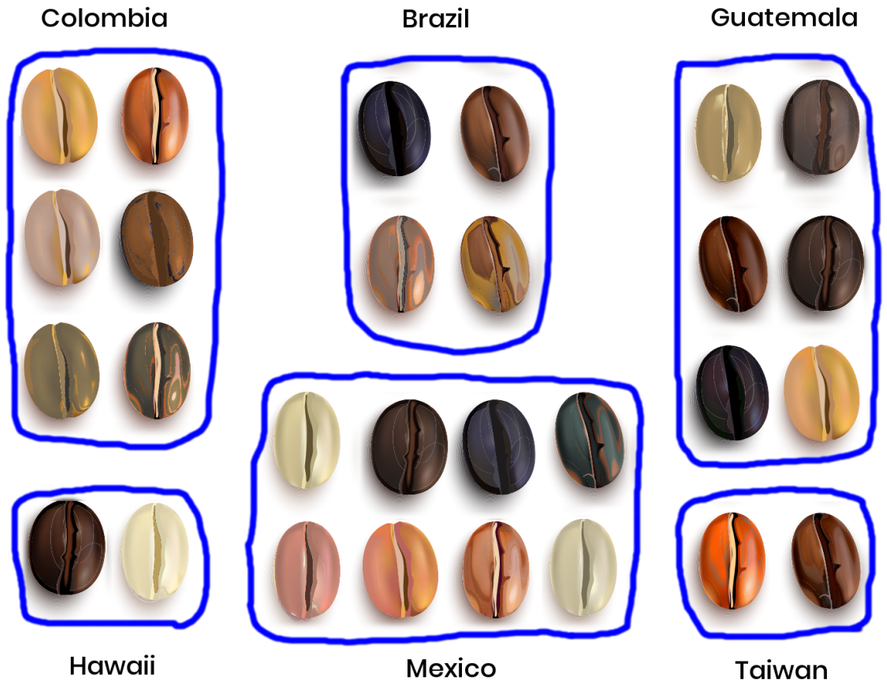
SAMPLING IN PYTHON



James Chapman
Curriculum Manager, DataCamp

datacamp

Coffees by country




```
top_counts = coffee_ratings['country_of_origin'].value_counts()
top_counts.head(6)
```

```
country_of_origin
Mexico                    236
Colombia                  183
Guatemala                 181
Brazil                    132
Taiwan                     75
United States (Hawaii)     73
dtype: int64
```

[1] The dataset lists Hawaii and Taiwan as countries for convenience, as they are notable coffee-growing regions.

Filtering for 6 countries

```
top_counted_countries = ["Mexico", "Colombia", "Guatemala",
  "Brazil", "Taiwan", "United States (Hawaii)"]

top_counted_subset = coffee_ratings['country_of_origin'].isin(top_counted_countries)

coffee_ratings_top = coffee_ratings[top_counted_subset]
```

Counts of a simple random sample

```
coffee_ratings_samp = coffee_ratings_top.sample(frac=0.1, random_state=2021)
```

```
coffee_ratings_samp['country_of_origin'].value_counts(normalize=True)
```

```
country_of_origin
Mexico                      0.250000
Guatemala                   0.204545
Colombia                    0.181818
Brazil                      0.181818
United States (Hawaii)      0.102273
Taiwan                      0.079545
dtype: float64
```

Comparing proportions

Population:

```
Mexico                          0.268182
Colombia                        0.207955
Guatemala                       0.205682
Brazil                          0.150000
Taiwan                          0.085227
United States (Hawaii)          0.082955
Name: country_of_origin, dtype: float64
```

10% simple random sample:

```
Mexico                          0.250000
Guatemala                       0.204545
Colombia                        0.181818
Brazil                          0.181818
United States (Hawaii)          0.102273
Taiwan                          0.079545
Name: country_of_origin, dtype: float64
```

Proportional stratified sampling

```
coffee_ratings_strat = coffee_ratings_top.groupby("country_of_origin")\
    .sample(frac=0.1, random_state=2021)
```

```
coffee_ratings_strat['country_of_origin'].value_counts(normalize=True)
```

```
Mexico                      0.272727
Guatemala                   0.204545
Colombia                    0.204545
Brazil                      0.147727
Taiwan                      0.090909
United States (Hawaii)      0.079545
Name: country_of_origin, dtype: float64
```

Equal counts stratified sampling

```
coffee_ratings_eq = coffee_ratings_top.groupby("country_of_origin")\
    .sample(n=15, random_state=2021)
```

```
coffee_ratings_eq['country_of_origin'].value_counts(normalize=True)
```

```
Taiwan                    0.166667
Brazil                    0.166667
United States (Hawaii)    0.166667
Guatemala                 0.166667
Mexico                    0.166667
Colombia                  0.166667
Name: country_of_origin, dtype: float64
```

Weighted random sampling

- Specify weights to adjust the relative probability of a row being sampled

```
import numpy as np
coffee_ratings_weight = coffee_ratings_top
condition = coffee_ratings_weight['country_of_origin'] == "Taiwan"
```

```
coffee_ratings_weight['weight'] = np.where(condition, 2, 1)
```

```
coffee_ratings_weight = coffee_ratings_weight.sample(frac=0.1, weights="weight")
```

Weighted random sampling results

10% weighted sample:

```
coffee_ratings_weight['country_of_origin'].value_counts(normalize=True)
```

```
Brazil                      0.261364
Mexico                      0.204545
Guatemala                   0.204545
Taiwan                      0.170455
Colombia                    0.090909
United States (Hawaii)      0.068182
Name: country_of_origin, dtype: float64
```

Let's practice!

SAMPLING IN PYTHON

datacamp

Cluster sampling

SAMPLING IN PYTHON



James Chapman
Curriculum Manager, DataCamp

datacamp

Stratified sampling vs. cluster sampling

Stratified sampling

- Split the population into subgroups
- Use simple random sampling on every subgroup

Cluster sampling

- Use simple random sampling to pick some subgroups
- Use simple random sampling on only those subgroups

Varieties of coffee



```
varieties_pop = list(coffee_ratings['variety'].unique())
```

```
[None, 'Other', 'Bourbon', 'Catimor',
'Ethiopian Yirgacheffe','Caturra',
'SL14', 'Sumatra', 'SL34', 'Hawaiian Kona',
'Yellow Bourbon', 'SL28', 'Gesha', 'Catuai',
'Pacamara', 'Typica', 'Sumatra Lintong',
'Mundo Novo', 'Java', 'Peaberry', 'Pacas',
'Mandheling', 'Ruiru 11', 'Arusha',
'Ethiopian Heirlooms', 'Moka Peaberry',
'Sulawesi', 'Blue Mountain', 'Marigojipe',
'Pache Comun']
```

Stage 1: sampling for subgroups



```
import random
varieties_samp = random.sample(varieties_pop, k=3)
```

```
['Hawaiian Kona', 'Bourbon', 'SL28']
```

Stage 2: sampling each group

```
variety_condition = coffee_ratings['variety'].isin(varieties_samp)
coffee_ratings_cluster = coffee_ratings[variety_condition]
```

```
coffee_ratings_cluster['variety'] = coffee_ratings_cluster['variety'].cat.remove_unused_categories()
```

```
coffee_ratings_cluster.groupby("variety")\
    .sample(n=5, random_state=2021)
```

Stage 2 output

		total_cup_points	variety	country_of_origin	...
variety					
Bourbon	575	82.83	Bourbon	Guatemala	
	560	82.83	Bourbon	Guatemala	
	524	83.00	Bourbon	Guatemala	
	1140	79.83	Bourbon	Guatemala	
	318	83.67	Bourbon	Brazil	
Hawaiian Kona	1291	73.67	Hawaiian Kona	United States (Hawaii)	
	1266	76.25	Hawaiian Kona	United States (Hawaii)	
	488	83.08	Hawaiian Kona	United States (Hawaii)	
	461	83.17	Hawaiian Kona	United States (Hawaii)	
	117	84.83	Hawaiian Kona	United States (Hawaii)	
SL28	137	84.67	SL28	Kenya	
	452	83.17	SL28	Kenya	
	224	84.17	SL28	Kenya	
	66	85.50	SL28	Kenya	
	559	82.83	SL28	Kenya	

Multistage sampling

- Cluster sampling is a type of multistage sampling
- Can have > 2 stages
- E.g., countrywide surveys may sample states, counties, cities, and neighborhoods

Let's practice!

SAMPLING IN PYTHON

datacamp

Comparing sampling methods

SAMPLING IN PYTHON



James Chapman
Curriculum Manager, DataCamp

datacamp

Review of sampling techniques - setup

```
top_counted_countries = ["Mexico", "Colombia", "Guatemala",
  "Brazil", "Taiwan", "United States (Hawaii)"]

subset_condition = coffee_ratings['country_of_origin'].isin(top_counted_countries)
coffee_ratings_top = coffee_ratings[subset_condition]

coffee_ratings_top.shape
```

```
(880, 8)
```

Review of simple random sampling

```
coffee_ratings_srs = coffee_ratings_top.sample(frac=1/3, random_state=2021)
```

```
coffee_ratings_srs.shape
```

```
(293, 8)
```

Review of stratified sampling

```
coffee_ratings_strat = coffee_ratings_top.groupby("country_of_origin")\
    .sample(frac=1/3, random_state=2021)
```

```
coffee_ratings_strat.shape
```

```
(293, 8)
```

Review of cluster sampling

```
import random
top_countries_samp = random.sample(top_counted_countries, k=2)
top_condition = coffee_ratings_top['country_of_origin'].isin(top_countries_samp)
coffee_ratings_cluster = coffee_ratings_top[top_condition]
coffee_ratings_cluster['country_of_origin'] = coffee_ratings_cluster['country_of_origin']\
    .cat.remove_unused_categories()

coffee_ratings_clust = coffee_ratings_cluster.groupby("country_of_origin")\
    .sample(n=len(coffee_ratings_top) // 6)
```

```
coffee_ratings_clust.shape
```

```
(292, 8)
```

Calculating mean cup points

Population

```
coffee_ratings_top['total_cup_points'].mean()
```

```
81.94700000000002
```

Simple random sample

```
coffee_ratings_srs['total_cup_points'].mean()
```

```
81.95982935153583
```

Stratified sample

```
coffee_ratings_strat['total_cup_points'].mean()
```

```
81.92566552901025
```

Cluster sample

```
coffee_ratings_clust['total_cup_points'].mean()
```

```
82.03246575342466
```

Mean cup points by country: simple random

Population:

```
coffee_ratings_top.groupby("country_of_origin")\
    ['total_cup_points'].mean()
```

```
country_of_origin
Brazil                   82.405909
Colombia                 83.106557
Guatemala                81.846575
Mexico                   80.890085
Taiwan                   82.001333
United States (Hawaii)   81.820411
Name: total_cup_points, dtype: float64
```

Simple random sample:

```
coffee_ratings_srs.groupby("country_of_origin")\
    ['total_cup_points'].mean()
```

```
country_of_origin
Brazil                   82.414878
Colombia                 82.925536
Guatemala                82.045385
Mexico                   81.100714
Taiwan                   81.744333
United States (Hawaii)   82.008000
Name: total_cup_points, dtype: float64
```

Mean cup points by country: stratified

Population:

```
coffee_ratings_top.groupby("country_of_origin")\
    ['total_cup_points'].mean()
```

```
country_of_origin
Brazil                   82.405909
Colombia                 83.106557
Guatemala                81.846575
Mexico                   80.890085
Taiwan                   82.001333
United States (Hawaii)   81.820411
Name: total_cup_points, dtype: float64
```

Stratified sample:

```
coffee_ratings_strat.groupby("country_of_origin")\
    ['total_cup_points'].mean()
```

```
country_of_origin
Brazil                   82.499773
Colombia                 83.288197
Guatemala                81.727667
Mexico                   80.994684
Taiwan                   81.846800
United States (Hawaii)   81.051667
Name: total_cup_points, dtype: float64
```

Mean cup points by country: cluster

Population:

```
coffee_ratings_top.groupby("country_of_origin")\
    ['total_cup_points'].mean()
```

```
country_of_origin
Brazil                    82.405909
Colombia                  83.106557
Guatemala                 81.846575
Mexico                    80.890085
Taiwan                    82.001333
United States (Hawaii)    81.820411
Name: total_cup_points, dtype: float64
```

Cluster sample:

```
coffee_ratings_clust.groupby("country_of_origin")\
    ['total_cup_points'].mean()
```

```
country_of_origin
Colombia    83.128904
Mexico      80.936027
Name: total_cup_points, dtype: float64
```

Let's practice!

SAMPLING IN PYTHON

datacamp