

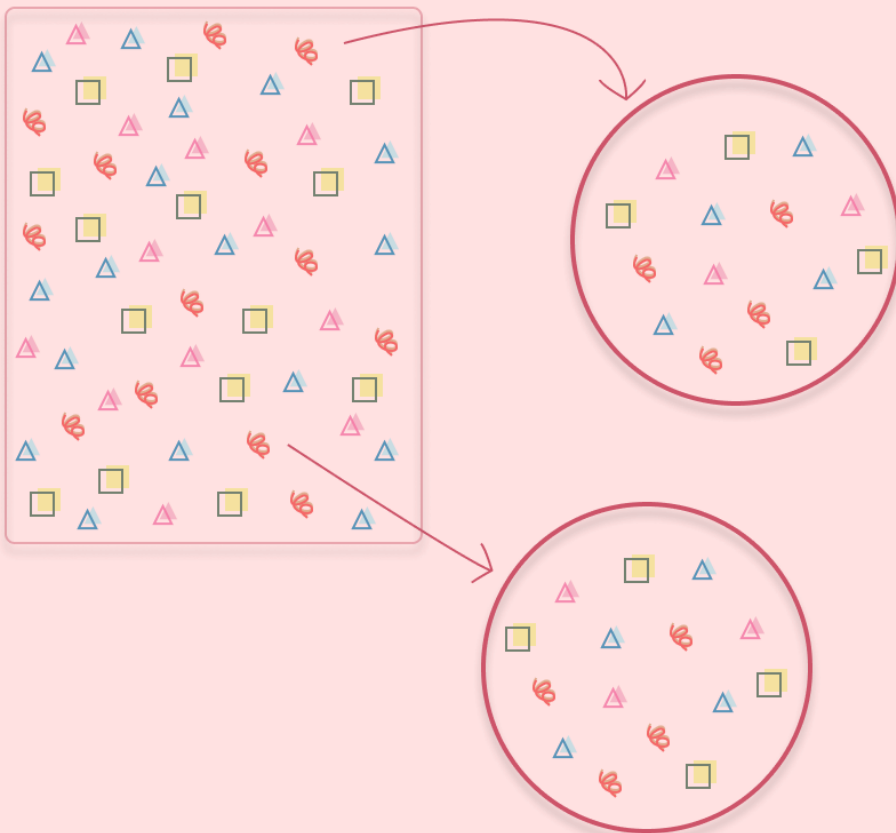
Get started

Open in app



Follow

610K Followers



Central Limit Theorem: a real-life application



Carolina Bento Oct 15, 2020 · 10 min read

The Central Limit Theorem (CLT) is one of the most popular theorems in statistics and it's very useful in real world problems. In this article we'll see why the Central Limit Theorem is so useful and how to apply it.

[Get started](#)[Open in app](#)

In a lot of situations where you use statistics, the ultimate goal is to identify the characteristics of a *population*.

Central Limit Theorem is an approximation you can use when the population you're studying is so big, it would take a long time to gather data about each individual that's part of it.

Population

Population is the group of individuals that you are studying. And even though they are referred to as individuals, the elements that make a population don't need to be people.

If you're a regional manager at a grocery chain and you're trying to be more efficient at re-stocking the seltzer water section every week in every store, so you sell as much seltzer as possible and avoid ending up with a lot of unsold inventory, all the *cases of seltzer sold* in that particular store represent the population.

If you're a poultry farmer and want to put in an order for chicken feed, you'll need to know how many pounds of grain your hens typically eat. So here, the *chickens* are your population.

Studying the population is hard

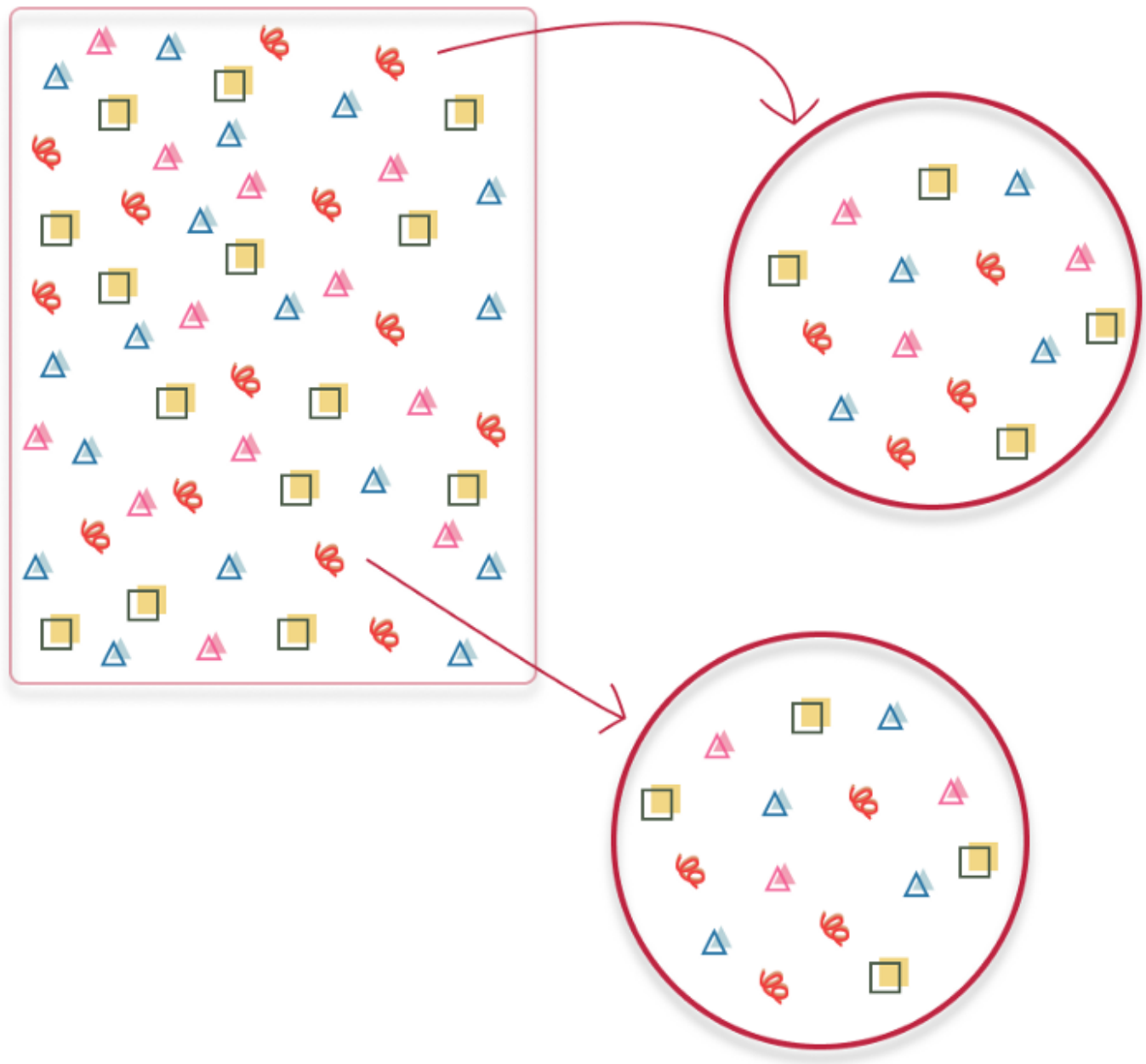
Depending on the problem you're solving, it will be extremely hard to gather data for the entire population.

If a company like Coca-Cola wants to know if their US customers will like the new product they are developing, they can't send an army of researchers to talk to every single person in the US. Well, they probably could, but it would be very expensive and would take a long time to collect all the data 😊

That's why companies do user studies with several groups of people that represent of their product's audience, their population, so they can gather data and determine if it's worth moving forward with product development. All of this, without talking to the entire population.

[Get started](#)[Open in app](#)

population.



Taking two samples from the population.

A good sample must be:

- Representative of the population,

[Get started](#)[Open in app](#)

- Picked at random, so you're not biased towards certain characteristics in the population.

Representative samples

A representative sample must showcase all the different characteristics of the population.

If you want to know who is more likely to win the Super Bowl and decide to poll the US population, i.e., take a sample from the US population, you need to make sure to talk to people from:

- All the different states about who they think is going to win,
- Different age groups and different genders,

And only include in your study the people that have interest in sports or in the event itself otherwise, they will not be part of the population that is interested in what you're studying.

The case(s) of seltzer

You're the regional manager at a grocery chain, in charge of 350 stores in the region, and the next project you're going to take on is to optimize the weekly re-stocking of seltzer water.

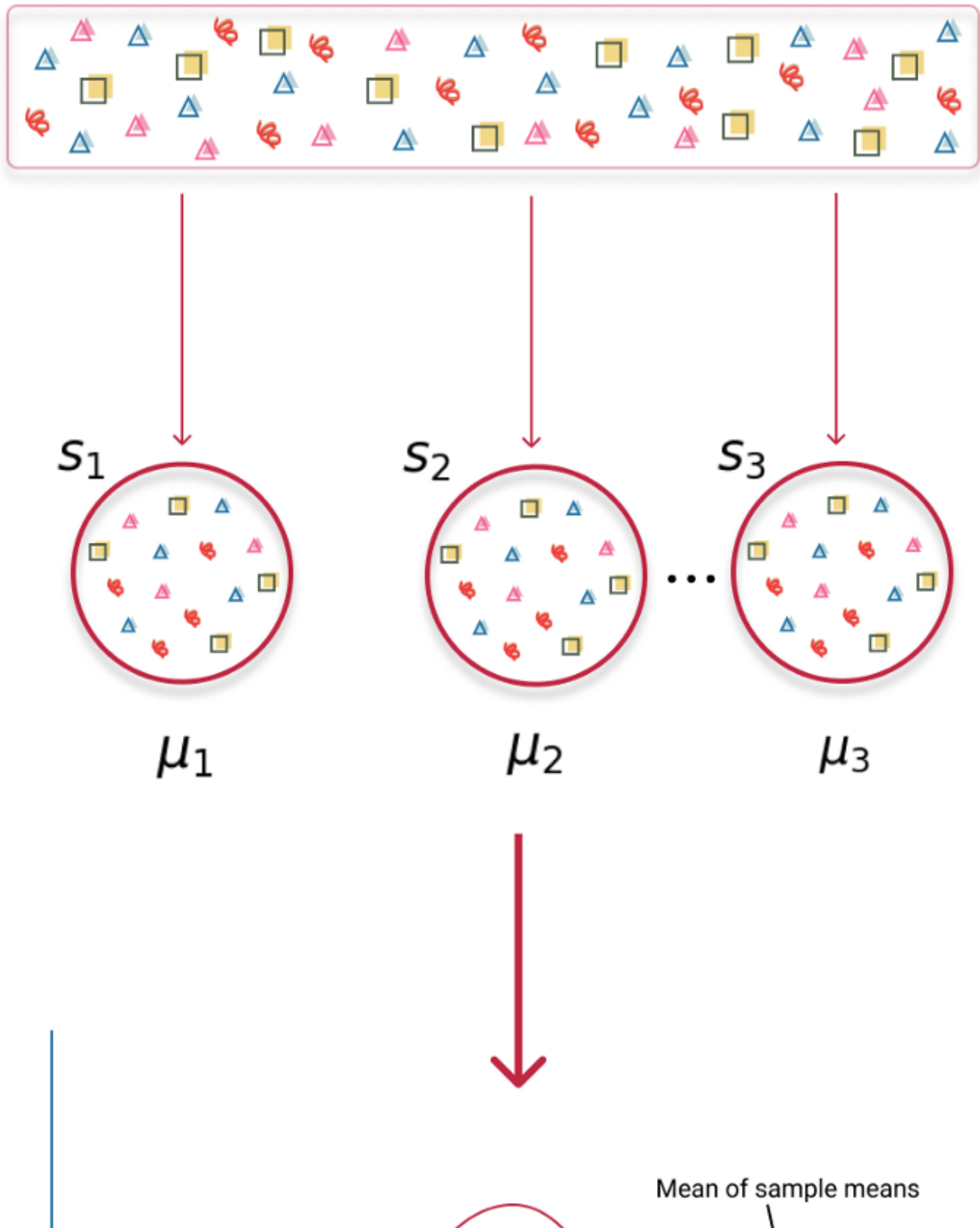
You want to know many cases of seltzer water to order weekly, for each store, so you minimize the amount of inventory that ends up sitting idle in store shelves.

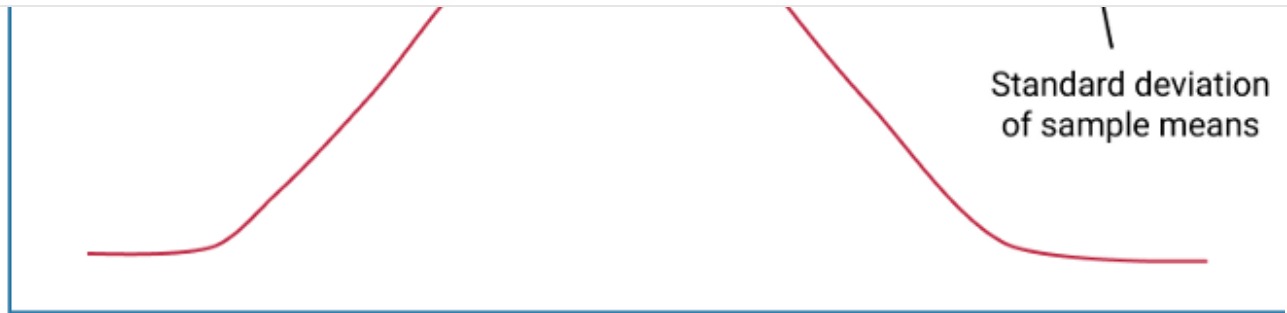
You know there has to be a better way to get to a plausible answer that doesn't involve visiting every single store in your region and get their sales numbers every single week.

Since you've taken a few statistics classes, the Central Limit Theorem comes to mind. You know that, applied to real-world problems, the Central Limit Theorem helps you balance the time and cost of collecting all the data you need to draw conclusions about the population.

[Get started](#)[Open in app](#)

when we collect a sufficiently large sample of n independent observations from a population with mean μ and standard deviation σ , the sampling distribution the sample means will be nearly normal with mean $= \mu$ and standard error $= \sigma/\sqrt{n}$



[Get started](#)[Open in app](#)

Visualizing the Central Limit Theorem. Taking samples from the population, getting their mean and creating the sample means distribution.

The Central Limit Theorem tells you that we don't have to visit every single store in the region and get their seltzer sales numbers for the week to know how many cases to put in the next order. What you can do is collect many samples from weekly sales in your stores (the population), calculate their mean (the average number of seltzer cases sold) and build the distribution of the sample means. This distribution is also referred to as sampling distribution.

If these samples meet Central Limit Theorem's criteria, you can assume the distribution of the sample means can be approximated to the Normal distribution. So now you can use all the statistical tools the Normal distribution provides.

From this point on, since you know the distribution at hand, you can calculate probabilities and confidence intervals, and perform statistical tests.

Population sample criteria for Central Limit Theorem

But before you use the Central Limit Theorem and use the Normal distribution approximation, your samples must meet a specific set of criteria that extends the characteristics of what is a good sample.

Your samples should be:

- *Picked at random*, so you're not biased towards certain characteristics in the population and you guarantee each observation in the sample is independent of all other observations. This also helps enforce that each observation in the sample is independent.

[Get started](#)[Open in app](#)

big enough to draw conclusions from, which in statistics is a sample size greater or equal to 30.

- *Include less than 10% of the population*, if you're sampling *without* replacement. Since observations in the population are not all independent of each other, if you collect a sample that is too big you may end up collect observations that are not independent of each other. Even if those observations were picked at random.

You don't need to know the population distribution — Central Limit Theorem's super power

If you want to use any kind inferential statistical methods, i.e., understand the characteristics of probability distribution of your data, you need to know the distribution your data follows. Otherwise, you might end up using the wrong tools for the job.

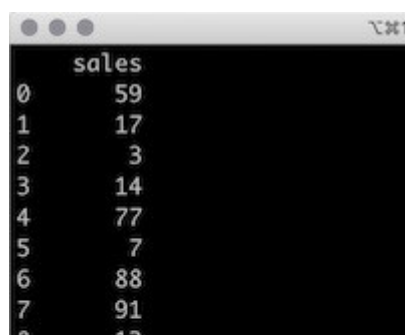
So one question that comes to mind is *Do I need to know the distribution of my population to use the Central Limit Theorem?*

The short answer is No 😊

What is really powerful about the Central Limit Theorem is that you don't need to know the distribution your population in advance. All you need to do is collect enough samples that follow the criteria and you can be sure that the distribution of the sample means will follow a Normal distribution.

How many cases of seltzer do you need to re-stock every week?

To answer this question let's generate a random dataset to represents the population, where each data point is the total number of seltzer cases sold per week in each store of the region you supervise.



	sales
0	59
1	17
2	3
3	14
4	77
5	7
6	88
7	91
8	13

Get started

Open in app



```

12      5
13      6
14      5
15     47
16     85
17     82

```

Snippet from the randomly generated seltzer sales dataset.

```

import pandas as pd
import random
import glob

def create_dataset(dataset_size):
    """ Creating the population dataset """
    dataset = []

    while dataset_size > 0:
        dataset.append(random.randrange(3, 100))
        dataset_size -= 1

    return dataset

# Initializing the random number generator
random.seed(1234)

# Reading the output directory in case we've already generated the
population dataset
dataset_file_list = glob.glob("output/sales_dataset.csv")

sales_data = None

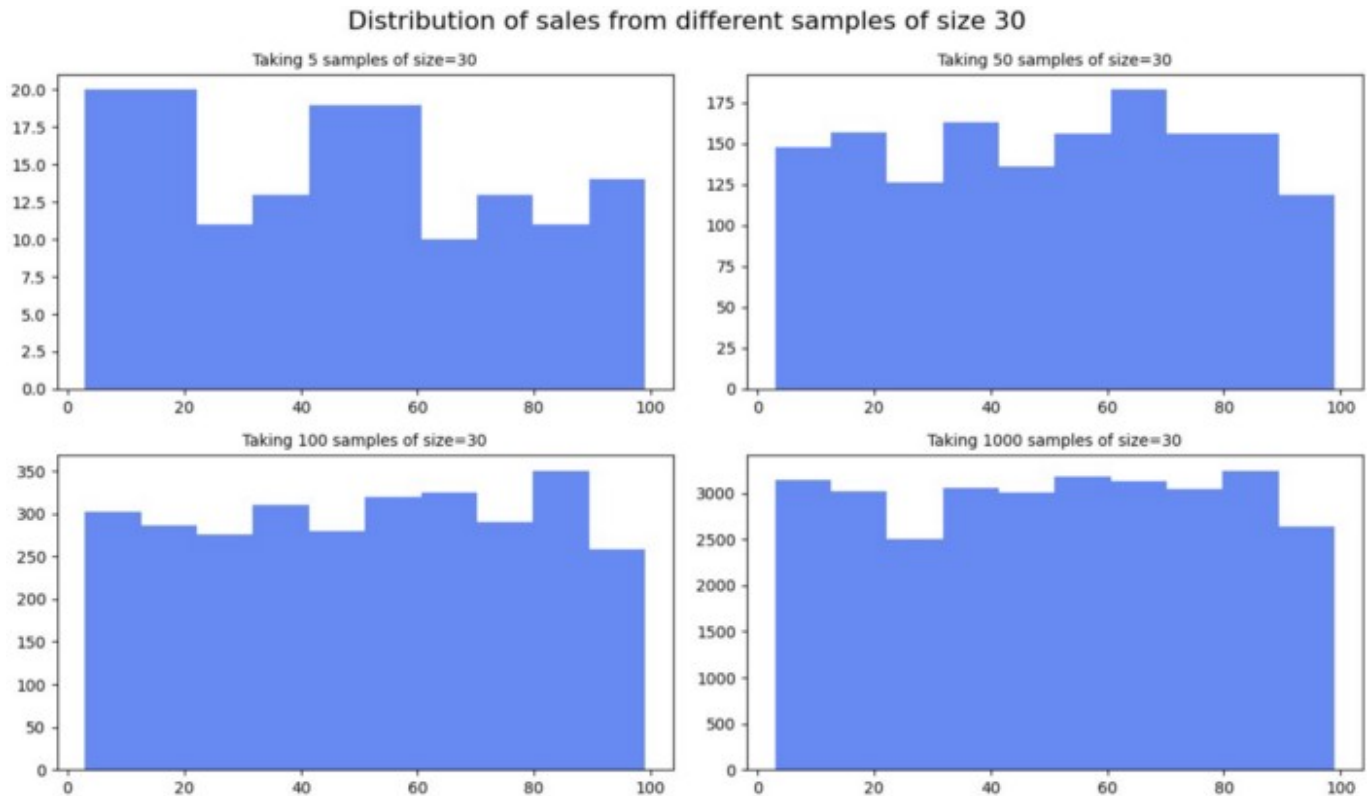
# Creating the population dataset and saving it to avoid always
recreating the dataset
if len(dataset_file_list) == 0:
    sales_data = pd.DataFrame(data=create_dataset(4200))
    sales_data.columns = ['sales']
    sales_data.to_csv("output/sales_dataset.csv", index=False)
else:
    sales_data = pd.read_csv('output/sales_dataset.csv')

```

Then you can take a different number of samples, all with the same size, and plot the sales data just to see how it looks like.

[Get started](#)[Open in app](#)

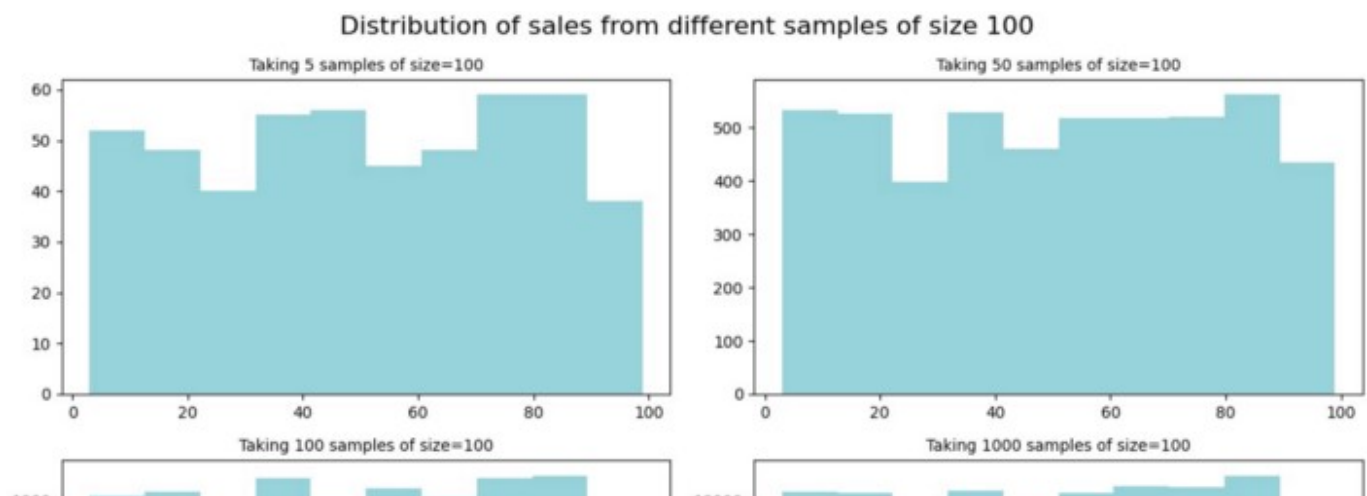
distribution of the population.



Distribution of different number of samples taken from the population, each sample with 30 data points.

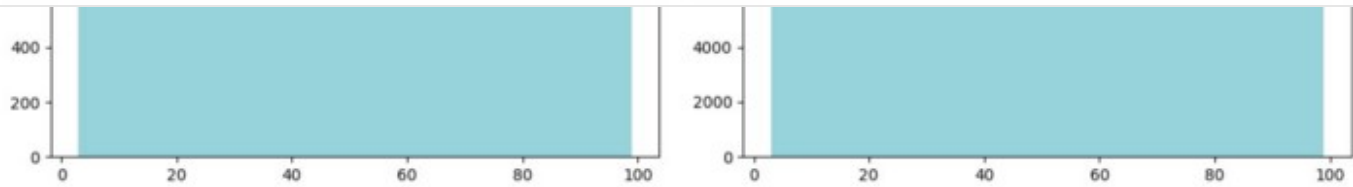
In this example, each chart has a different numbers of samples, all with size 30, and none of the distributions look like the classic *Bell curve*. Not even close.

That doesn't change much when you take another set of samples, this time with 100 data points each.



Get started

Open in app



Distribution of different number of samples taken from the population, each sample with 100 data points.

```
import numpy as np
import matplotlib.pyplot as plt

def picking_n_samples(population, number_samples, sample_size):
    """ Sampling without replacement with fixed size
        Returning the array of sample and array with their respective
        mean
    """
    results = []
    sample_mean = []

    while number_samples > 0:
        new_sample = random.sample(population, sample_size)
        results += new_sample
        sample_mean += [np.mean(new_sample)]
        number_samples -= 1

    return [results, sample_mean]

def generate_sample_sets(dataset, number_samples, sample_size):
    """ Generate multiple sets samples with fixed size
        Returns all sample sets and their corresponding set of means
    """

    samples_array = []
    sample_means_array = []

    for sample_count in number_samples:
        new_sample, sample_mean = picking_n_samples(dataset,
        sample_count, sample_size)
        samples_array.append(new_sample)
        sample_means_array.append(sample_mean)

    return [samples_array, sample_means_array]

def plot_samples(sample_array, number_samples, default_size,
plot_color='#6689F2', title='', x_axis_title='', filename='plot'):
```

Get started

Open in app



```

    ax1.hist(sample_array[0], color=plot_color)
    ax1.set_title("Taking " + str(number_samples[0]) + " samples of
size=" + str(default_size), fontsize=10)
    ax1.set_xlabel(x_axis_title)

    ax3.hist(sample_array[2], color=plot_color)
    ax3.set_title("Taking " + str(number_samples[2]) + " samples of
size=" + str(default_size), fontsize=10)
    ax3.set_xlabel(x_axis_title)

    ax2.hist(sample_array[1], color=plot_color)
    ax2.set_title("Taking " + str(number_samples[1]) + " samples of
size=" + str(default_size), fontsize=10)
    ax2.set_xlabel(x_axis_title)

    ax4.hist(sample_array[3], color=plot_color)
    ax4.set_title("Taking " + str(number_samples[3]) + " samples of
size=" + str(default_size), fontsize=10)
    ax4.set_xlabel(x_axis_title)

    fig.savefig("output/" + filename)

#####

### Example 1

#####

# Setting the defaults for this example
example1_number_of_samples_array = [5, 50, 100, 1000, 10000]
example1_default_sample_size = 30

# Picking multiple samples of size 30
example_1_samples, example_1_means =
generate_sample_sets(list(sales_data['sales'].values),
example1_number_of_samples_array, example1_default_sample_size)

# Plot the different sets of samples
plot_title = 'Distribution of sales from different samples of size '
+ str(example1_default_sample_size)

plot_samples(example_1_samples, example1_number_of_samples_array,
example1_default_sample_size, title=plot_title,
filename="example_1_samples_distribution")

#####

```

Get started

Open in app



```
# Setting the defaults for this example
example_2_number_of_samples_array = [5, 50, 100, 1000, 10000]
example_2_default_sample_size = 100

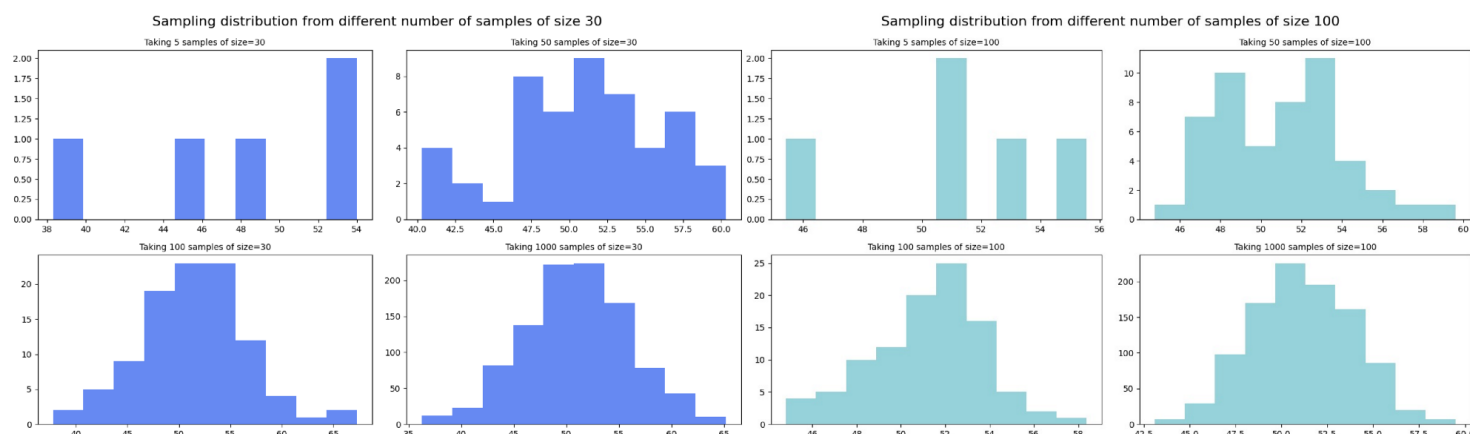
example_2_samples, example_2_means =
generate_sample_sets(list(sales_data['sales'].values),
example_2_number_of_samples_array, example_2_default_sample_size)

# Plot the different sets of samples
plot_title = 'Distribution of sales from different samples of size '
+ str(example_2_default_sample_size)

plot_samples(example_2_samples, example_2_number_of_samples_array,
example_2_default_sample_size, title=plot_title,
filename="example_2_samples_distribution", plot_color="#96D2D9")
```

Because the key is to take a sample and the calculate the mean!

Looking at the distribution of sample means of the previous examples it becomes clear. As the number of samples taken increases, the closer you get to the shape of a Normal distribution.



Sampling distribution for different of samples taken from the population, each sample with 30 (left) and 100 data points (right).

The higher number samples will also reduce the variability in the sampling distribution.

Get started

Open in app



If you collect a bigger sample you'll have fewer chances of getting extreme values, so your values will be more clustered together. Therefore the standard deviation, or the distance from the mean, will be smaller.

To approach it from formulaic way, looking back to the definition of the Central Limit Theorem, the standard deviation of the sampling distribution, also called standard error, is equal to σ/\sqrt{n} . So, as the sample size increases the denominator also increases, and makes the overall standard value smaller.

```
Example 1: Summary statistics for sampling distribution with 5 samples taken (size= 30)
count    5.000000
mean     47.746667
std       6.274755
min      38.300000
25%      45.333333
50%      48.600000
75%      52.500000
max      54.000000
```

```
Example 1: Summary statistics for sampling distribution with 50 samples taken (size= 30)
count    50.000000
mean     50.907333
std       4.949730
min      40.300000
25%      47.708333
50%      51.283333
75%      54.383333
max      60.300000
```

Summary statistics for sampling distributions with (top) 5 samples of size 30 and (bottom) 50 samples of size 30.

```
example_1_sampling_distribution_5_samples =
pd.Series(example_1_means[0])

print("Example 1: Summary statistics for sampling distribution with "
+ str(len(example_1_sampling_distribution_5_samples)) + " samples
taken (size= " + str(example1_default_sample_size) + ")")
print(example_1_sampling_distribution_5_samples.describe())

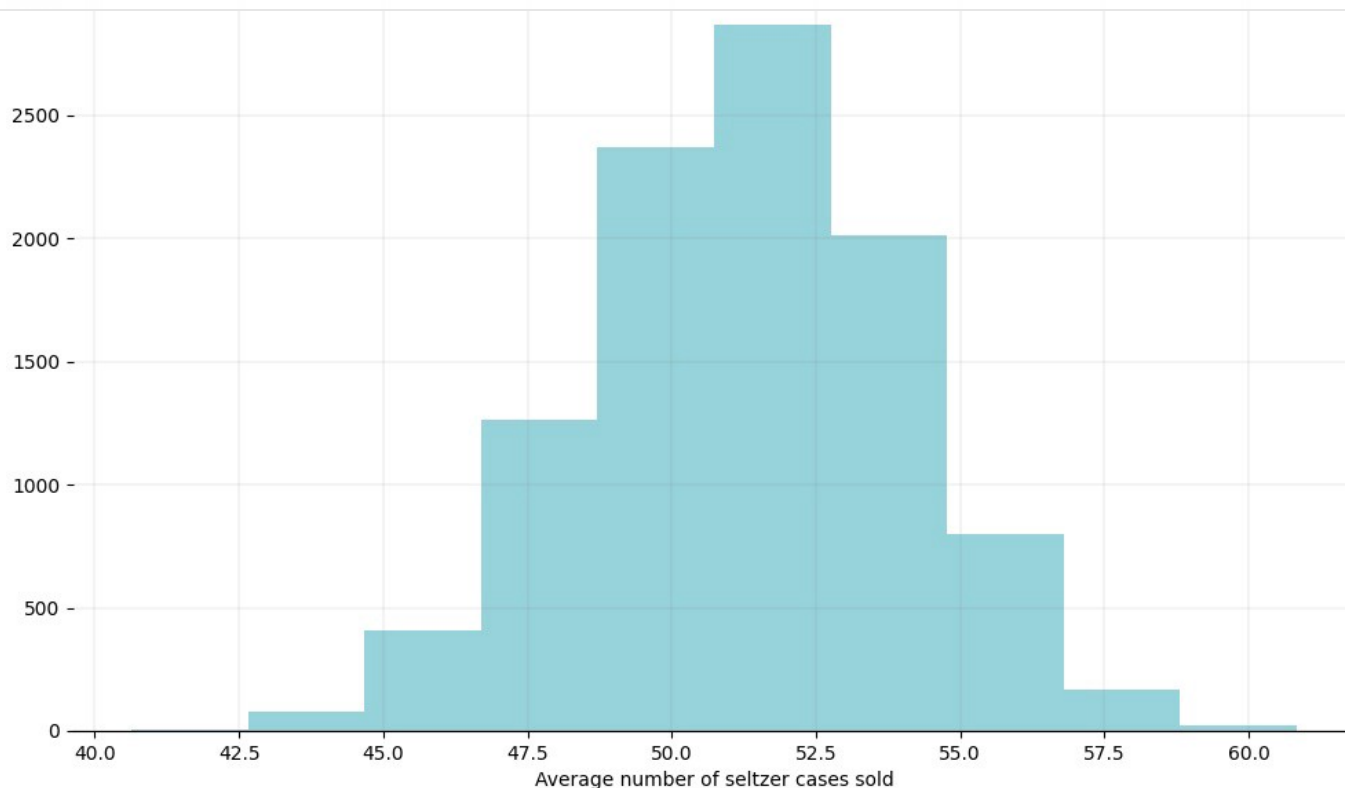
example_1_sampling_distribution_5_samples =
pd.Series(example_1_means[1])

print("Example 1: Summary statistics for sampling distribution with "
+ str(len(example_1_sampling_distribution_5_samples)) + " samples
taken (size= " + str(example1_default_sample_size) + ")")
print(example_1_sampling_distribution_5_samples.describe())
```

And if you take 10,000 samples of size 100 from the randomly generated sales dataset, you'll get a sampling distribution that resembles the *bell curve* characteristic of the Normal distribution.

Get started

Open in app



```
def plot_sample_means(sample_means_array, plot_color='#A9CBD9',
title='', filename='plot'):
    fig, ax = plt.subplots(figsize=(12, 7))
    fig.suptitle(title, fontsize=16)
    ax.hist(sample_means_array, color=plot_color)

    # removing top and right border
    ax.spines['top'].set_visible(False)
    ax.spines['right'].set_visible(False)

    # adding major gridlines
    ax.grid(color='grey', linestyle='-', linewidth=0.25, alpha=0.5)

    ax.set_xlabel("Average number of seltzer cases sold")
    fig.savefig("output/" + filename)

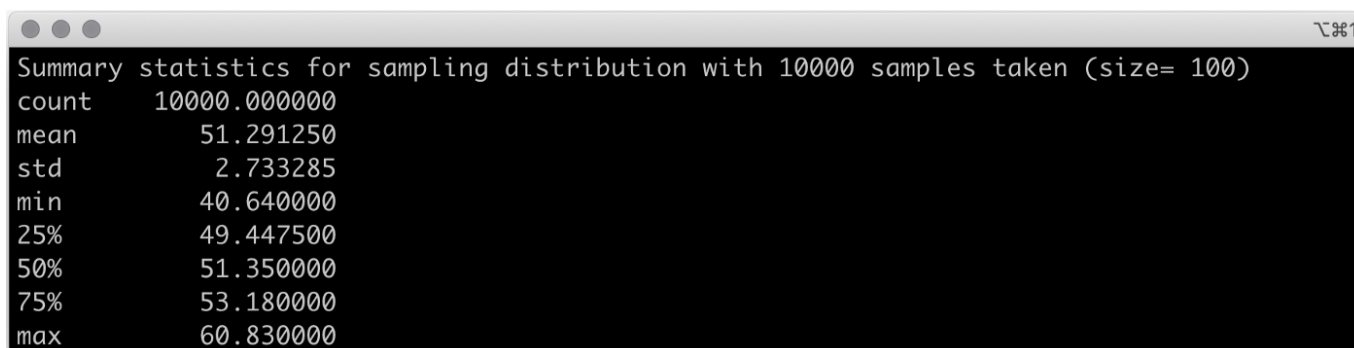
plot_title = 'Sampling distribution from taking 10,000 samples of
size 30 ' + str(example1_default_sample_size)

plot_sample_means(example_1_means[4], title=plot_title,
filename="example_1_sampling_dist_10ksamples")
```

[Get started](#)[Open in app](#)

Back to the original, if you want to know **how many cases of seltzer water you need to re-stock every week**, take a look at the summary statistics of this last sampling distribution, the one with 10,000 samples.

The mean of the sampling distribution is 51, so you'll need an *average of 51 cases per store per week*.



```
Summary statistics for sampling distribution with 10000 samples taken (size= 100)
count      10000.000000
mean        51.291250
std         2.733285
min         40.640000
25%         49.447500
50%         51.350000
75%         53.180000
max         60.830000
```

Summary statistics for the sampling distribution with 10,000 samples of size 100.

```
example_2_sampling_distribution = pd.Series(example_2_means[4])

print("Summary statistics for sampling distribution with " +
      str(example_2_number_of_samples_array[4]) + " samples taken (size= " +
      str(example_2_default_sample_size) + ")")

print(example_2_sampling_distribution.describe())
```

This is the average across all stores in your region. If you wanted a more precise number per store, you'd have to do this process for each of them. Each store becomes the population, and you only take samples from that that are from that store.

How close are you to the population mean?

Since you generated the sales dataset, you can do another interesting check. See how far the mean of sampling distribution is from the *real* population mean.

The average of the population is 51!

[Get started](#)[Open in app](#)

```
mean    51.233552
std     27.631507
min      3.000000
25%     27.000000
50%     52.000000
75%     75.000000
max     99.000000
```

Summary statistics for the population.

```
# Population summary statistics

print("Summary statistics for the population (sales dataset)")
print(sales_data['sales'].describe())
```

Conclusion

We just experienced the power of the Central Limit Theorem!

With a randomly generated set and not knowing any details about the original distribution (you only checked at the very end 😊) you:

1. Took an increasing number of samples and saw the distribution of the sample means becoming closer and closer to the shape of a Normal Distribution.
2. Confirmed that the average of the sampling distribution was very close to the population distribution, with a small margin of error.
3. Used the Central Limit Theorem to solve a real life problem.

Hope you enjoyed this article, thanks for reading!

References

[1][OpenIntro Statistics — Fourth Edition \(2019\)](#)

Sign up for The Variable

By Towards Data Science

Get started

Open in app



Get this newsletter

Central Limit Theorem

Statistics

Data Science

Probability

Editors Pick

About Write Help Legal

Get the Medium app

