conf_interval

November 3, 2022

Confidence Intervals + Quick Reference to my fantastic CI tutorial + This code is based on this blog post

We are taking the data from Uniform distribution. According to the CLT. Means of any other distribution will follow a normal distribution. with std of $\frac{\sigma}{\sqrt{n}}$ where n is the sample size an σ is the global std which usually is estimated by sample std. If we can say (based on CLT) that the sample emeans center around the global mean with std $\frac{\sigma}{\sqrt{n}}$ then based on the characteristics of normal distribution we can also say that that 66 % percent of the means are +/- 1 std away from the global mean. ~95 % is ~2 std away from the global mean and ~99% means is 3 std away from global mean. Thus if we have the sample mean and based on the sample size and sample std the std for the sample means, we can now say with specific confidence an interval where the means should be located.

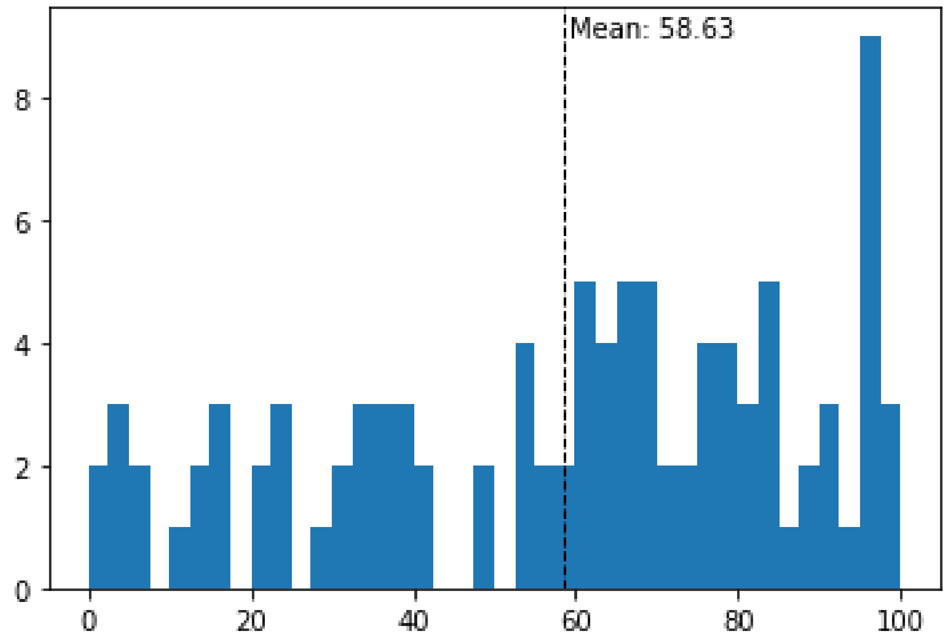
[ ]:
```
import numpy as np
from scipy.stats import t,norm
import math
import matplotlib.pyplot as plt
from dataclasses import dataclass

@dataclass
class ConfidenceInterval:
    mean:float
    lower_bound:float
    upper_bound:float
    test_type:str
    extremum_val:float
    def get_size(self)->float:
        return self.upper_bound-self.lower_bound
    def print_statistics(self):
        print(f"Median: {self.mean}")
        print(f"Min boundary {self.lower_bound}")
        print(f"Max boundary {self.upper_bound}")
        print(f"Width is {self.get_size()}" )
        print()

def␣
 ↪calculate_confidence_interval(sample_mean,sample_std,sample_size,confidence,test_type)->Con
 ↪
```

```
    alpha=1-confidence
    # Only thing that is different for t and z confidence intervale is the␣
↪extremum value
    if test_type == "t":
        degrees_of_freedom=sample_size-1
        extremum_val=float(np.abs(t.ppf(alpha/2,degrees_of_freedom))) # t_val -␣
↪based on confidence we have chosen, how many standard distributions is the␣
↪ci width. If Confidence is 95%, then std is  ~2
    elif test_type == "z":
        extremum_val=float(np.abs(norm.ppf(alpha/2))) # z_val - based on␣
↪confidence we have chosen, how many standard distributions is the ci width.␣
↪If Confidence is 95%, then std is ~2.
    else:
        raise Exception("Wrong test type provided")

    clt_based_std = sample_std / math.sqrt(sample_size) # The std for sample␣
↪means. Calculated based on CLT
    max_expected_difference = extremum_val *  clt_based_std # std * (how many␣
↪stds based on chosen confidence interval size)
    return␣
↪ConfidenceInterval(mean=sample_mean,lower_bound=sample_mean-max_expected_difference,upper_b



def perform_comparison(sample_size:int=100, confidence:float=0.95):
    # Generate sample data from uniform distritubution on the range 0..100
    x = np.random.uniform(size=100)
    x_positive = x-min(x)
    x_scaled = x_positive/max(x_positive)*100
    x = x_scaled
    # Calculate confidence intervals based on both Normal distribution (Z val)␣
↪and t distribution (t val)
    ci_z=calculate_confidence_interval(sample_mean=x.mean(),sample_std=x.
↪std(),sample_size=sample_size,confidence=confidence,test_type='z')
    ci_t=calculate_confidence_interval(sample_mean=x.mean(),sample_std=x.
↪std(),sample_size=sample_size,confidence=confidence,test_type='t')
    return x, ci_z, ci_t
```

[ ]:
```
sample_size=15
confidence=0.95
x, ci_z, ci_t = perform_comparison(sample_size=sample_size,confidence=0.95)
print(f"Sample size: {sample_size}, Sample std: {x.std()}, Confidence␣
↪{confidence}, Max value {max(x)}, Min value: {min(x)}")



# plt.text((m+s)*1.1, max_ylim*0.9, 'Mean+Std: {:.2f}'.format(m+s))
```

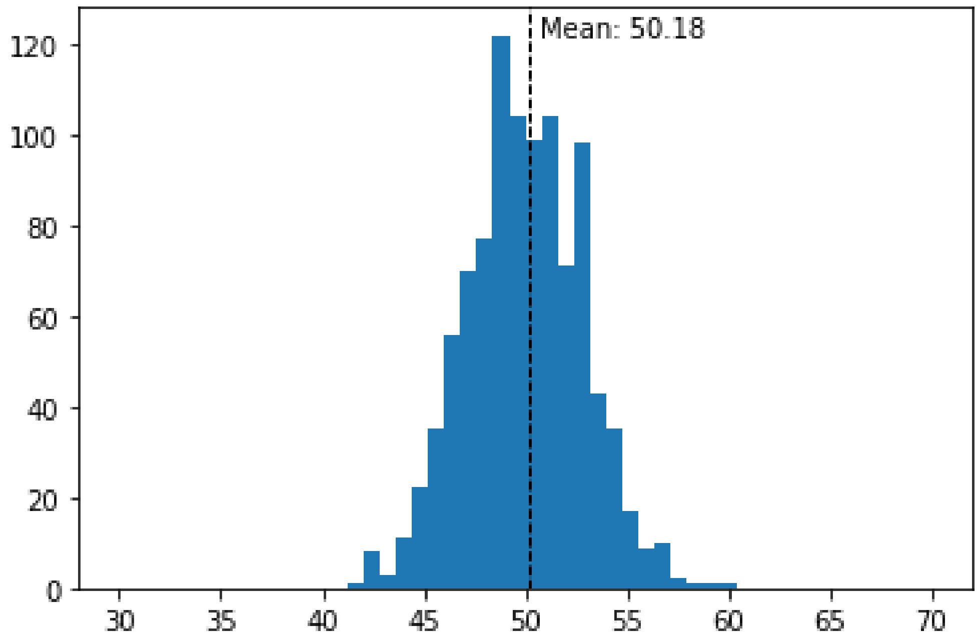
```python
# plt.text((m-s)*1.1, max_ylim*0.9, 'Mean+Std: {:.2f}'.format(m-s))

ci_z.print_statistics()
ci_t.print_statistics()
print(f"Differences in CI size ci_t-ci_z {ci_t.get_size()-ci_z.get_size()}")

# PLOT THE GENERATED DISTRIBUTION X

def plot_default_histogram(x,bins=40,start=0,end=100):
    counts, bins = np.histogram(x,bins=bins)
    plt.hist(bins[:-1],len(bins)-1,weights=counts,range=(start,end))
    plt.axvline(x.mean(), color='k', linestyle='dashed', linewidth=1)
    # plt.axvline(m+s, color='r', linestyle='dashed', linewidth=1)
    # plt.axvline(m-s, color='r', linestyle='dashed', linewidth=1)
    min_ylim, max_ylim = plt.ylim()
    plt.text(x.mean()*1.01, max_ylim*0.95, 'Mean: {:.2f}'.format(x.mean()))

plot_default_histogram(x)
```

```
Sample size: 15, Sample std: 28.635469814784415, Confidence 0.95, Max value
100.0, Min value: 0.0
Median: 58.62569127706479
Min boundary 44.13441039605592
Max boundary 73.11697215807366
Width is 28.982561762017745

Median: 58.62569127706479
Min boundary 42.76789665960571
Max boundary 74.48348589452387
Width is 31.715589234918156

Differences in CI size ci_t-ci_z 2.7330274729004103
```



Now we perform this experiment 1000 times and see how the distribution of means will look like. E.g. we take the sample with size 100 and we do this 1000 times. Then we plot the means of these 1000 samples. Based on the Central Limit Theorem (CLT) this distribution should now have ~ global mean as the mean and the standard distribution $\frac{\sigma}{\sqrt{n}}$ where σ= Global std and n is sample size for 1 sample. **PS! In reality we almost always limit our experiment with taking 1 sample only. The current setup aims to showcase how CLT works and how this allows us to build Confidence Intervals, Hypothesis tests and all other good things.**

[ ]:
```
means = []
for i in range(1000):
    x, ci_z, ci_t = perform_comparison(sample_size=sample_size,confidence=0.95)
    means.append(x.mean())

means = np.asarray(means)
plot_default_histogram(means,bins=50,start=30,end=70)
```

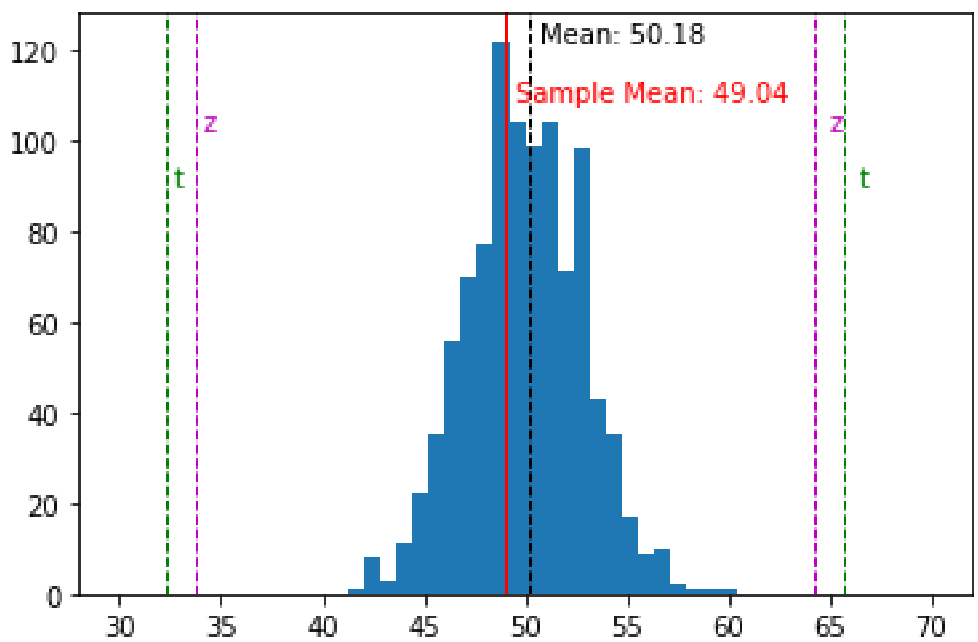


Lets now take a few additional samples to show how we derive the confidence interval from 1 sample. **Run this cell at least 5 times to see how the confidence intervals are generated for each sample ADDITIONAL SAMPLE 1 COMPARED TO MEAN**

[ ]:
```
def take_sample_and_show(means):
    x_sample, sample_ci_z, sample_ci_t =␣
 ↪perform_comparison(sample_size=sample_size,confidence=0.95)
    plot_default_histogram(means,bins=50,start=30,end=70)

    # ADD THE SAMPLE
    plt.axvline(x_sample.mean(), color='r', linewidth=1)
    min_ylim, max_ylim = plt.ylim()
    plt.text(x_sample.mean()*1.01,max_ylim*0.85, 'Sample Mean: {:.2f}'.
 ↪format(x_sample.mean()),color='r')

    print(f"Sample size: {len(x_sample)}, Sample std: {x_sample.std()},␣
 ↪Confidence {confidence}, Max value {max(x_sample)}, Min value:␣
 ↪{min(x_sample)}")

    # PLOT CI INFOR FOR Z TESTS
    z_color='m'
    plt.axvline(sample_ci_z.upper_bound, color=z_color, linestyle='dashed',␣
 ↪linewidth=1)
    plt.axvline(sample_ci_z.lower_bound, color=z_color, linestyle='dashed',␣
 ↪linewidth=1)
    plt.text(sample_ci_z.upper_bound*1.01, max_ylim*0.8, 'z',color=z_color)
```

```
    plt.text(sample_ci_z.lower_bound*1.01, max_ylim*0.8, 'z',color=z_color)

    # PLOT CI INFO FOR T TESTS
    t_color='g'
    plt.axvline(sample_ci_t.upper_bound, color=t_color, linestyle='dashed',␣
 ↪linewidth=1)
    plt.axvline(sample_ci_t.lower_bound, color=t_color, linestyle='dashed',␣
 ↪linewidth=1)
    plt.text(sample_ci_t.upper_bound*1.01, max_ylim*0.7, 't',color=t_color)
    plt.text(sample_ci_t.lower_bound*1.01, max_ylim*0.7, 't',color=t_color)
take_sample_and_show(means)
```

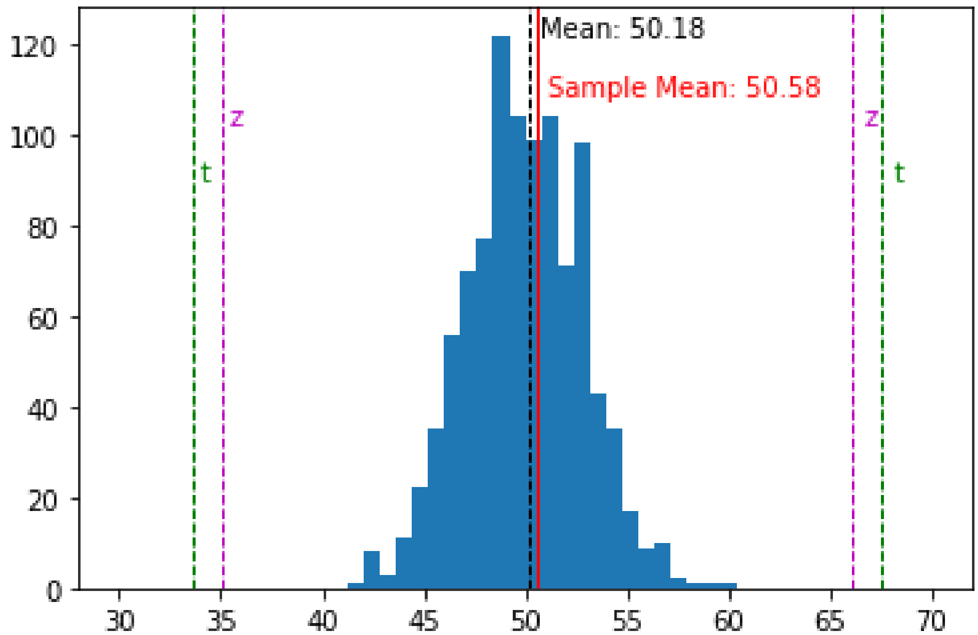
```
Sample size: 100, Sample std: 30.160875214238498, Confidence 0.95, Max value
100.0, Min value: 0.0
```



ADDITIONAL SAMPLE 2 COMPARED TO MEAN

[ ]:
```
take_sample_and_show(means)
```

```
Sample size: 100, Sample std: 30.53389768088036, Confidence 0.95, Max value
100.0, Min value: 0.0
```



ADDITIONAL SAMPLE 3 COMPARED TO MEAN

[ ]:
```
take_sample_and_show(means)
```

```
Sample size: 100, Sample std: 29.464664243459364, Confidence 0.95, Max value
100.0, Min value: 0.0
```