Table of Contents

- Bootstrapping
- Bootstrapping to get confidence interval
- Bootstrapping

Bootstrapping

- We do bootstrapping only on the sample, if we already have population, we don't need to do bootstrappin.g
- Sampling with replacement
- We need just one sample from a large population.
- We create thousands of bootstrapped samples from that single sample.
- We get parameter estimate from each bootstrap sample.
- Then we get the statistic as mean and confidence interval from ordered data.

pseudocode

```
statistics = []
for i in bootstraps:
    sample = select_sample_with_replacement(data)
    stat = calculate_statistic(sample)
    statistics.append(stat)

ordered = sort(statistics)
lower = percentile(ordered, (1-alpha)/2)
upper = percentile(ordered, alpha+((1-alpha)/2))
```

```
import numpy as np
import tqdm
from tqdm import trange
np.random.seed(0)

population = np.random.randint(0,500,size=1000) # 1000 persons
pop_mean = population.mean() # average ht of 1000 students.
print('population mean = ', pop_mean)

# use bootstrapping
#*** We have only one sample, still we get 1 million bootsample from this**
sample = np.random.choice(population,size=30)
boot_means = []
reps = 1000_000

for _ in tqdm.trange(reps):
    bootsample = np.random.choice(sample, size=len(sample),replace=True)
    boot_means.append(bootsample.mean())

boot_mean = np.mean(boot_means)
print('bootstrapped mean = ', boot_mean)

# using more memory
#==============================================
np.random.seed(0)
reps = 1000_000

x = np.random.choice(population,size=30)
xb = np.random.choice(sample, (len(x),reps),replace=True)
print('xb shape = ', xb.shape) # (30, 1000000)

mb = xb.mean(axis=0)
print('mb shape = ', mb.shape) # (1000000,)

mbb = mb.mean()
print('boot mean = ', mbb) # 255.7214197999999

xb.mean() # 255.7214198
```

Bootstrapping to get confidence interval

```
import numpy as np
import tqdm
from scipy import stats

alpha = 0.05
a = np.array([1,2,3,4,4,4,5,5,5,5,4,4,4,6,7,8])


reps = 1_000

sample = a # suppose that a is sample drawn from big population
ci_points = [] # point estimate
for _ in tqdm.trange(reps):
    bootsample = np.random.choice(sample, size=len(sample),replace=True)

    # make sure to use bootsample, not sample!
    ci_lo, ci_hi = stats.t.interval(1-alpha,
                                    df = len(bootsample)-1,
                                    loc=np.mean(bootsample),
                                    scale=stats.sem(bootsample))

    ci_point = (ci_lo+ci_hi)/2
    ci_points.append(ci_point)

ci_point = np.mean(ci_points)
ci_lo, ci_hi = np.percentile(ci_points, [alpha/2*100,100-alpha/2*100]) # alpha/2*100 is 2.5
```

Bootstrapping

- Duke university: Resampling And Monte Carlo Simulations

```
# For example, what is the 95% confidence interval for
# the mean of this data set if you didn't know how it was generated?

x = np.concatenate([np.random.exponential(size=200),
                    np.random.normal(size=100)])
plt.hist(x, 25, histtype='step');

n = len(x)
reps = 10000
xb = np.random.choice(x, (n, reps),replace=True)
mb = xb.mean(axis=0)
mb.sort()

np.percentile(mb, [2.5, 97.5])
```