Sampling Distributions

and where to find them

Objectives

- Define the sampling distribution of a statistic, give two examples.
- State the Central Limit Theorem.
- Use the bootstrap to approximate the sampling distribution of a statistic.
- Use the Central Limit Theorem to describe the sampling distribution of the mean.
- Use either the Central Limit Theorem or the Bootstrap to compute a confidence interval for a sample statistic.

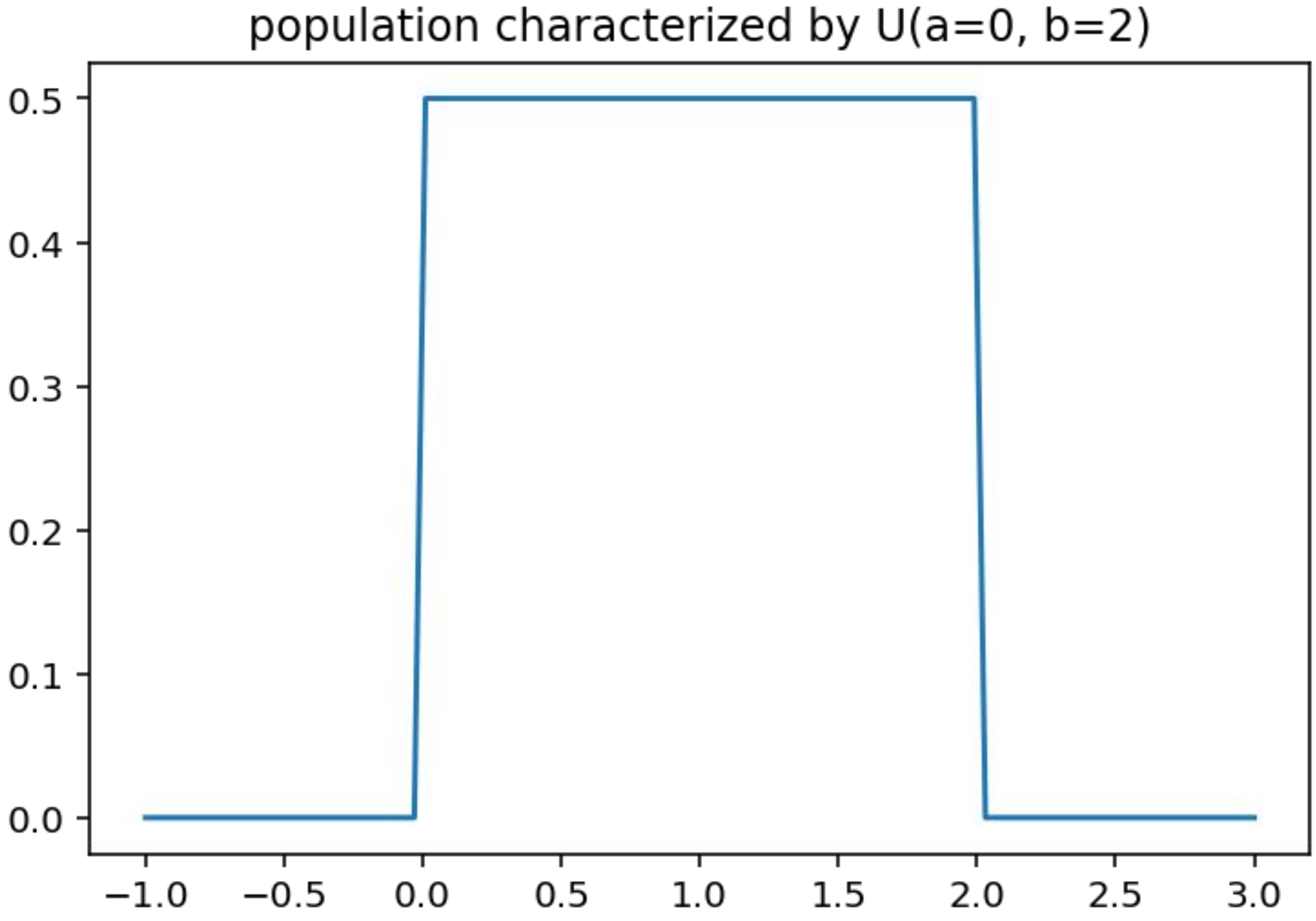
A Sampling Distribution

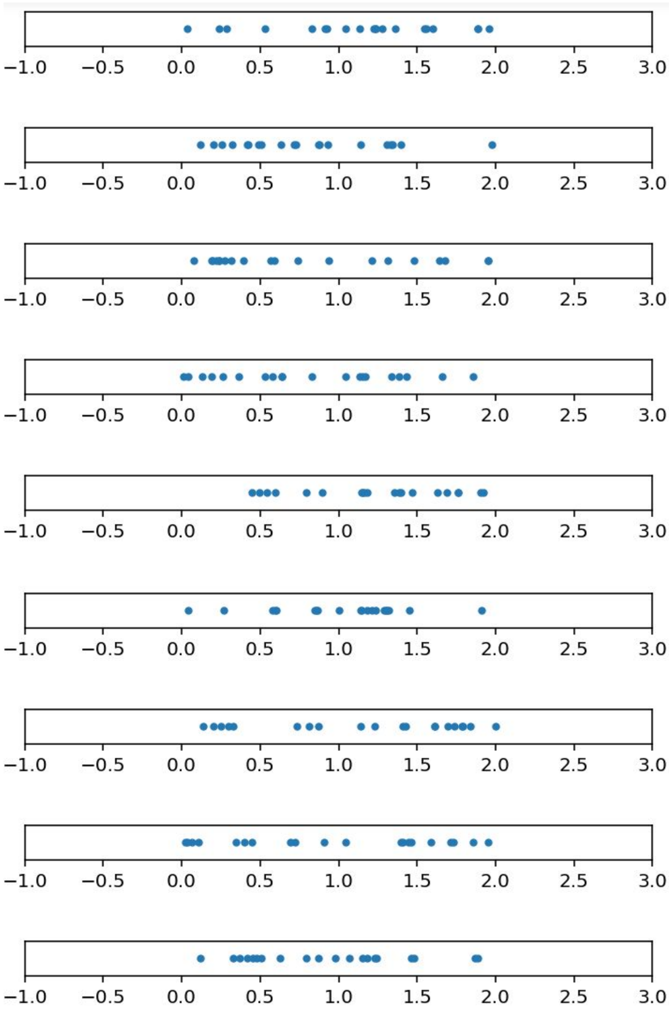
The distribution of a sample statistic over repeated re-samplings of the population.

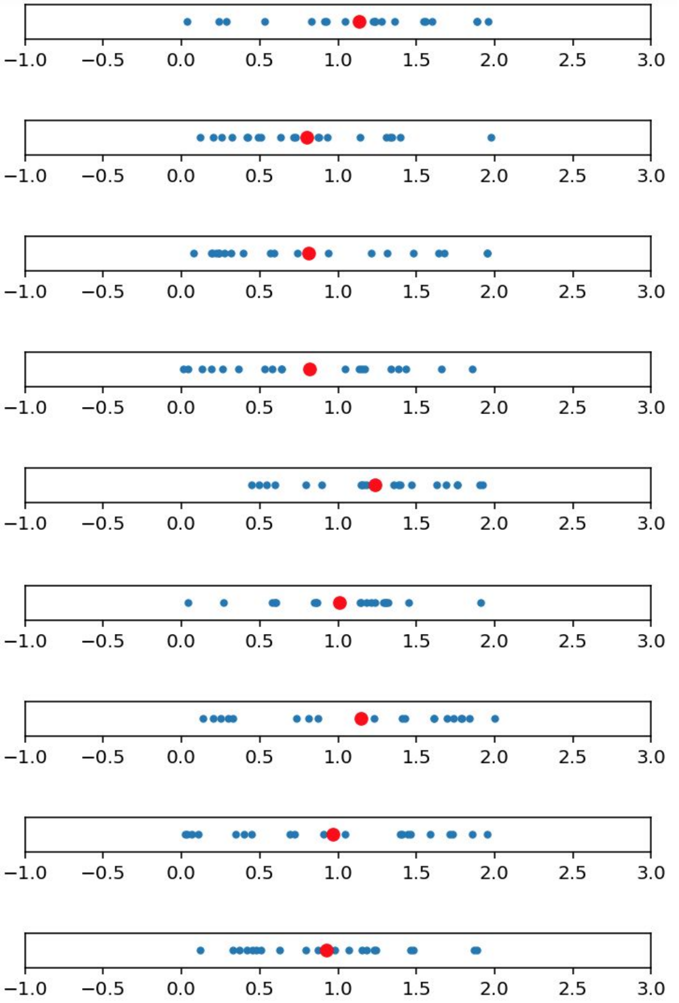
For example: the **sampling distribution of sample means.**

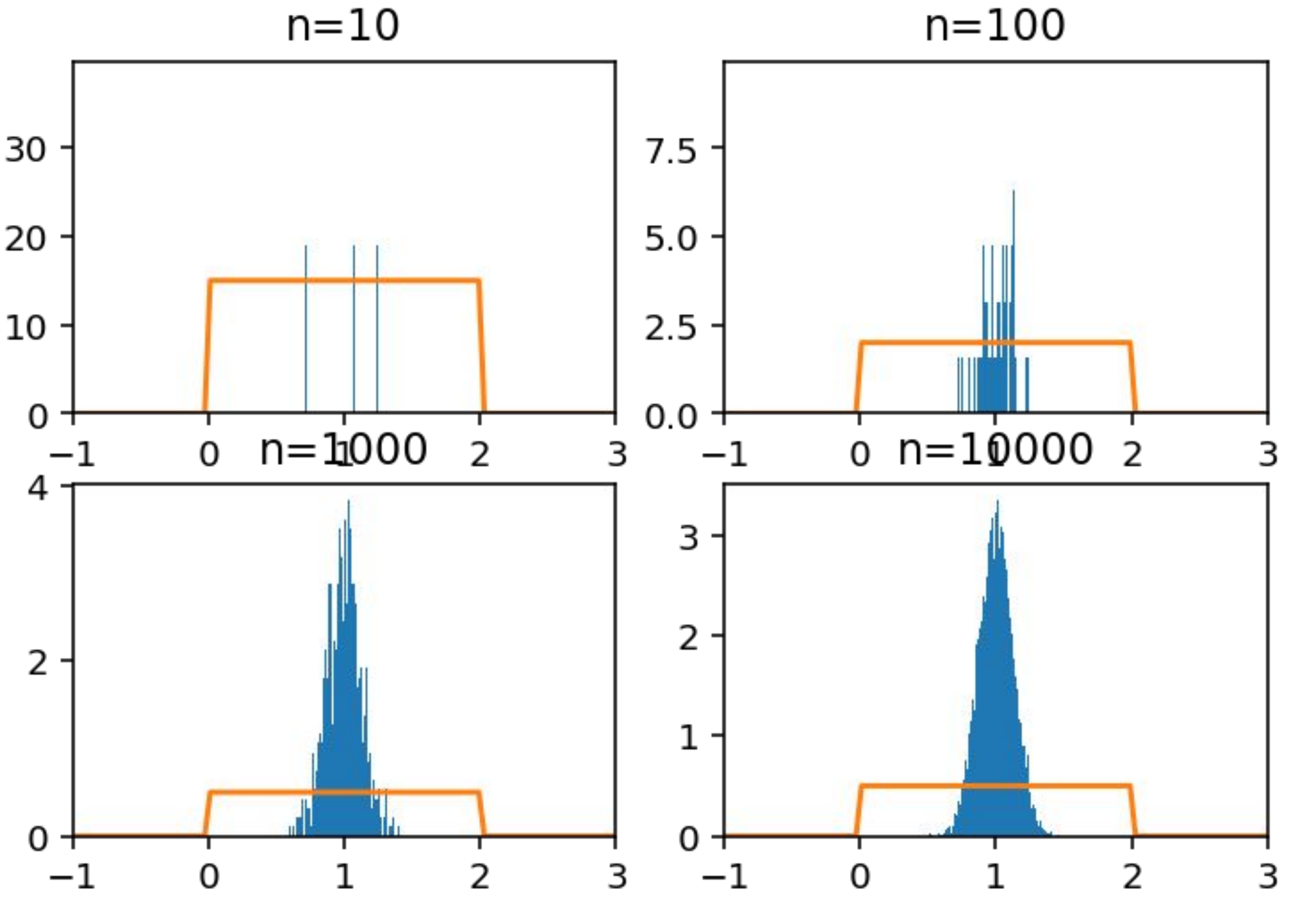
Or, the sampling distribution of:

- Sample maximum
- Sample 75th percentiles
- Sample medians
- Sample correlation (given multi-variate samples).


population characterized by U(a=0, b=2)
0.5
0.4
0.3
0.2
0.1
0.0
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
2.5
3.0


−1.0 −0.5 0.0 0.5 1.0 1.5 2.0 2.5 3.0
−1.0 −0.5 0.0 0.5 1.0 1.5 2.0 2.5 3.0
−1.0 −0.5 0.0 0.5 1.0 1.5 2.0 2.5 3.0
−1.0 −0.5 0.0 0.5 1.0 1.5 2.0 2.5 3.0
−1.0 −0.5 0.0 0.5 1.0 1.5 2.0 2.5 3.0
−1.0 −0.5 0.0 0.5 1.0 1.5 2.0 2.5 3.0
−1.0 −0.5 0.0 0.5 1.0 1.5 2.0 2.5 3.0
−1.0 −0.5 0.0 0.5 1.0 1.5 2.0 2.5 3.0
−1.0 −0.5 0.0 0.5 1.0 1.5 2.0 2.5 3.0


−1.0 −0.5 0.0 0.5 1.0 1.5 2.0 2.5 3.0



n=10
n=100
n=1000
n=10000
30
20
10
0
7.5
5.0
2.5
0.0
4
2
3
1
−1
0
1
2
3

- Compiling sampling distributions involves re-sampling the population.
- Generally we only get one chance to sample.
- How can we get a variety of samples when we only have one sample?

Numerical technique: The Bootstrap

- We don't have the population to sample from.
- We have the next best thing - a representative of the population, the sample.
- We can sample **from the sample**.
- The **bootstrap sample** is a set of items, the same size as our sample, drawn from our sample uniformly and with replacement.

```
def bootstrap(data):
    indices = np.random.randint(0, len(data), size=len(data))
    return data[indices]
```

Question: will data be repeated in the bootstrap? Will data be left out? How much?

Closed-form technique: The Central Limit Theorem

- One situation where the bootstrap is both less convenient and less accurate than a closed-form solution.
- The **central limit theorem** asserts that as we take the mean of larger and larger samples, the distribution of sample means becomes more and more normal.

Statement of the Central Limit Theorem

Suppose $X_1,\ X_2,\ \ldots$ are i.i.d. copies of a random variable with finite expectation and variance

$$Var(X_1) = Var(X_2) = \cdots = \sigma^2$$

Then the distribution of sample means tends to a normal distribution with the appropriate mean and standard deviation:

$$\frac{X_1 + X_2 + \cdots + X_k}{k} \to N\left(\mu,\ \frac{\sigma}{\sqrt{k}}\right)$$

as $k \to \infty$.

What's the point?

- Give us a means to state the **confidence interval** for the population statistic.
- For example, if 95% of the time the mean of our samples falls between a and b, we would be 95% confident reporting that the population mean is between a and b.
- Between the Central Limit Theorem and The Bootstrap, we have the means of deriving this confidence interval for almost any statistic.
- **Consider**: sometimes population statistics are used to fit a model.

Appendix A: Compare/contrast BS and CLT for sample distribution of sample means

In [240]:
```
#imagine we have a chance to weight 100 puppies a birth
np.random.seed(2)
sampleset = np.random.uniform(0, 2, size=100)
```

In [241]:
```
sampleset[0:20]
```

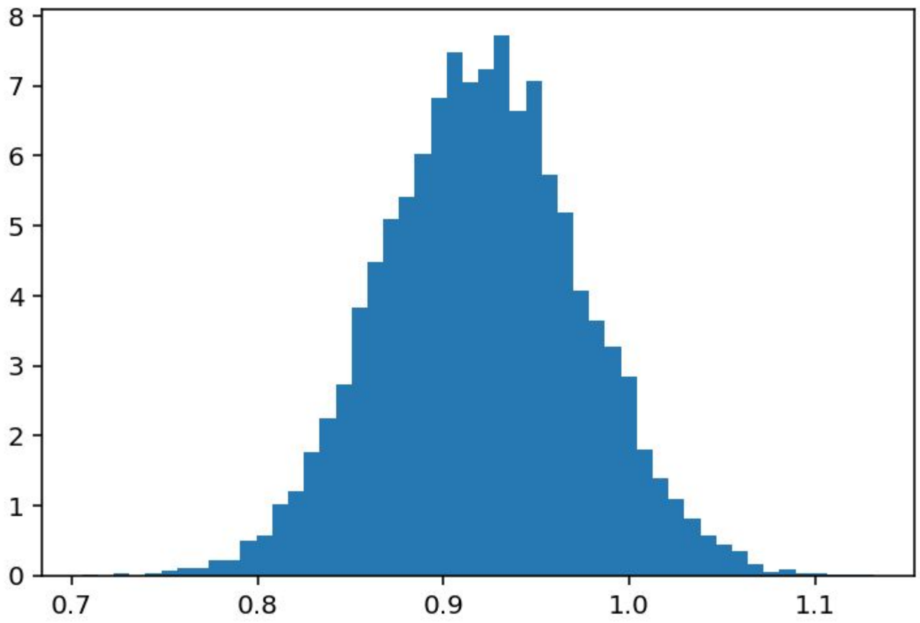
Out[241]:
```
array([0.8719898 , 0.05185246, 1.09932496, 0.87064479, 0.8407356 ,
       0.66066964, 0.40929727, 1.23854193, 0.59930935, 0.53365455,
       1.24226767, 1.05828419, 0.26915989, 1.02715624, 0.36887973,
       1.5706703 , 1.70795059, 0.98847367, 1.69312297, 0.15929095])
```

In [242]:
```
# define the bootstrap
def bootstrap(sampleset):
    indices = np.random.randint(0, len(sampleset), size=len(sampleset))

    return sampleset[indices]
```

In [243]:
```
# compute a large number of bootstrap samples
bootstraps = []
for i in range(10000):
    a_bootstrap = bootstrap( sampleset )

    bootstraps.append( a_bootstrap )
```

In [244]:
```
# find the means of those bootstrap samples
bootstrap_means = [bs.mean() for bs in bootstraps]
```

In [245]:
```
# observe the empirical sample distribution of sample means
plt.hist(bootstrap_means, bins=50, density=True)
plt.show()
```



```
In [246]: # alternatively, compute the mean and standard deviation of our sampleset
          sample_mean = sampleset.mean()
          sample_std = sampleset.std()
          sample_mean, sample_std

Out[246]: (0.9200564907043876, 0.5314990916225957)

In [247]: from scipy.stats import norm

In [248]: # and use them to create a normal distibution
          # in accordance with the central limit theorem

          standard_error = sample_std/len(sampleset)**0.5
          N = norm( loc=sample_mean, scale=standard_error )
```

In [249]:

```
# plot the PDF of normal derived from CLT
# the empirical distribution of sample means derived
# from the bootstrap is included, because it's cool

meanspace = np.linspace(0.7, 1.2)
p_meanspace = N.pdf(meanspace)
plt.plot(meanspace, p_meanspace)
plt.hist(bootstrap_means, bins=50, density=True)
plt.show()
```