Unit 8 Notebook: Common Types of Random Variables and How to Use Them - Building Blocks for Inference

1. A well-known type of discrete random variable: Bernoulli Random Variable

See unit 8 slides (section 1).

2. A well-known type of continuous random variable: Normal Random Variable

2.1. Normal Random Variable: Definition

See unit 8 slides (section 2.1).

2.2. Normal Random Variable: Mean, Variance, and, Standard Deviation

See unit 8 slides (section 2.2).

2.3. Normal Random Variable: Other Properties

See unit 8 slides (section 2.3).

2.4. Normal Random Variable: How do you KNOW when a distribution is approximately normal?

See unit 8 slides (section 2.4).

2.5. Calculating probabilities involving normal random variables - in Python?

See unit 8 slides (section 2.5) for more explanations.

In [25]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns; sns.set()
```

norm distribution object

Like with other well-studied random variables we've looked at, you can import the **norm** object from the **scipy.stats** package to perform various tasks related to **normal random variables.**

In [1]:
```
from scipy.stats import norm
```

Parameters needed for functions involving the norm object.

Most random variables have one or more parameters associated with them, that dictate various aspects of the shape and nature of the distribution. As we discussed, the normal distribution has two parameters:

- the **mean** parameter that dictates the **"location** of the normal distribution, and
- the **standard deviation** parameter that dictates the **scale** of the normal distribution.

In all Python functions involving a normal random varible, we need to provide:

- loc=mean
- scale=standard deviation.

Normal Distribution CDF:

If X is a random variable we can use the function

norm.cdf(x , loc = mean, scale = standard deviation)

to calculate

$$P(X \leq x) = P(X < x).$$

Ex: If X is a normal random variable with mean 64 and standard deviation 2.5, what is the probability that a randomly selected woman in the U.S. is **at most** 70"?

Answer:

$$P(X \leq 70) = P(X < 70) = 0.9918$$

```
In [2]: norm.cdf(70, loc=64, scale=2.5)
```

```
Out[2]: 0.9918024640754038
```

Ex: If X is a normal random variable with mean 64 and standard deviation 2.5, what is the probability that a randomly selected woman in the U.S. is **greater than** 70"?

Answer:

We can also indirectly use this function to calculate:

$$P(X > 70) = 1 - P(X \geq 70) = 1 - 0.9918 = 0.0082$$

```
In [4]: 1-norm.cdf(70, loc=64, scale=2.5)
```

```
Out[4]: 0.008197535924596155
```

Important Note: the .cdf() always finds areas/probabilies to the LEFT of the observation supplied. To find areas to the right you need to use 1 - .cdf()!

Ex: If X is a normal random variable with mean 64 and standard deviation 2.5, what is the probability that a randomly selected woman height is between 60" and 70"?

Answer: We can use our cdf properties (from Unit 7) to calculate:

$$P(60 < X < 70) = P(X < 70) - P(X < 60) = 0.9918 - 0.0548 = 0.9370$$

```
In [10]: norm.cdf(70, loc=64, scale=2.5) - norm.cdf(60, loc=64, scale=2.5)
Out[10]: 0.9370031723758458
```

2.6. Calculating percentiles involving normal random variables - in Python

See unit 8 slides (section 2.6) for more explanations.

Normal Percentile Function

If X is a random variable we can use the function

norm.ppf(p , loc = mean, scale = standard deviation)

to calculate the value of x in which

$$P(X \leq x) = P(X < x) = p.$$

Ex: The average height of a woman in the U.S. is about 64” with a standard deviation of 2.5”. Assume that the heights of women in the U.S. has a normal distribution. So if we let X = height of a randomly selected woman in the U.S., then we know that X is a normal random variable.

How tall does a woman in the U.S. have to be in order to be in the top 10% of women’s heights?

Answer: This is equivalent to asking what is the value of x in which $P(X < x) = 0.90$?

```
In [6]: norm.ppf(0.90, loc=64, scale=2.5)

Out[6]: 67.2038789138615
```

Thus, a woman that is 67.2" will be taller than 90% of women in the U.S. and shorter than 10% of women in the U.S.

Important Note: the .ppf() always finds x-axis values that correspond to the LEFT-TAIL AREA SUPPLIED. To find the x-axis value that corresponds to a RIGHT-TAIL AREA, you should supply the LEFT-TAIL AREA = 1 - RIGHT TAIL AREA to the function.

3. z-scores

See unit 8 slides (section 3).

3.1. z-scores: Definitions

See unit 8 slides (section 3.1).

3.2. z-scores: Relationship between a Random Variable X and the z-score of X

See unit 8 slides (section 3.2).

4. A well-known type of continous random variable: Standard Normal Random Variable

4.1. Standard Normal Random Variable: Definitions

See unit 8 slides (section 4.1).

4.2 Relationship between the z-Score of a Normal Random Variable X and a standard normal random variable

See unit 8 slides (section 3.2).

4.3 Calculating probabilities involving standard normal random variables - in Python

See unit 8 slides (section 4.3) for more information.

Because the **standard normal random variable** is just a special kind of **normal random variable**, specifically one in which the mean = 0 and the standard deviation = 1, we use the same **norm** object from **scipy.stats** and the corresponding functions to perform any tasks related to **standard normal random variables** as well.

In each of these **norm** functions, we can either:

- set **loc = 0** (ie. *explicitly* set the mean to 0) and **scale = 1** (ie. *explicitly* set the standard deviation to 1 OR
- NOT specify values for the **loc** and **scale** parameters. (The **default values** for the **norm** object are set to **0** and **1** automatically).

Ex: What is the probability that a standard normal random variable is between -1.96 and 1.96?

Answer:

$$P(-1.96 < Z < 1.96) = P(Z < 1.96) - P(Z < -1.96) = 0.975 - 0.025 = 0.95$$

```
In [14]: norm.cdf(1.96,loc=0,scale=1) - norm.cdf(-1.96,loc=0,scale=1)
Out[14]: 0.950004209703559
```

```
In [15]: norm.cdf(1.96) - norm.cdf(-1.96)
Out[15]: 0.950004209703559
```

4.4 Calculating percentiles involving standard normal random variables - in Python

See unit 8 slides (section 4.4) for more information.

Ex: What is the value in the standard normal distribution that is greater than 20% of observations?

Answer:

The value of z in which

$$P(Z \geq z) = 0.20$$

is $z = -0.8416$.

```
In [17]: norm.ppf(0.20, loc=0, scale=1)
Out[17]: -0.8416212335729142
```

```
In [18]: norm.ppf(0.20)
Out[18]: -0.8416212335729142
```

4.5 Using standard normal random variables to help solve problems involving normal random variables.

See unit 8 slides (section 4.5) for more information.

Ex: The average time it took for all seniors at a local high school to finish a race was 8 minutes. The finishing times followed a normal distribution.

You ran the race in 7 minutes and your time was in the top 20% (ie. your time was higher than 20% of the participants). What was the standard deviation of the finishing times?

Answer :

Let X = finishing time of a randomly selected senior from the high school.

Because **X is a normal random variable**, the **z-score of X** $(Z = \frac{X-E[X]}{SD[X}$) is a **standard normal random variable**.

Two things we know about the z-score of *your* finishing time (x=7).

1. The z-score of your finishing time of 7 minutes = $\frac{7-\mu}{\sigma} = \frac{7-8}{\sigma}$.
2. The z-score of your finishing time of 7 minutes is greater than 20% of other z-scores from the standard normal distribution. Using the ppf function, we know that this value is $z = -0.8416$.

So we can set these two pieces of information equal to eachother to get:

The z-score of your finishing time of 7 minutes = $-0.8416 = \frac{7-8}{\sigma}$ and solve for σ to get $\sigma = 1.188$ minutes.

```
In [19]: norm.ppf(0.20)
```

```
Out[19]: -0.8416212335729142
```

5. Representing a Sample Statistic with Multiple Random Variables

Go to unit 8 slides (section 5).

5.1. Sample Statistics of the Coin Flip Experiment in Python.

Ex: Consider our coin toss experiment from Unit 7. This is where we keep flipping a coin until we get a head. We let X=number of flips until stopping. We also learned that $X \sim geom(p = 0.5)$.

Suppose we decided to repeat this coin flip experiment multiple 20 times, we can represent X_i=number of flips in the ith experiment until stopping.

Generate a random value for each X_i. Use these values to generate a random values for:

- $\bar{X} = (X_1 + X_2 + \cdots + X_{20})/20$: the mean number of flips until stopping from the 20 experiments
- M: the median number of flips until stopping from the 20 experiments
- S: the standard deviation number of flips until stopping from the 20 experiments
- P_2: the proportion of the 20 experiments in which the number of flips (until stopping) was at most 2.

Also use these values to generate a random sample distribution.

5.1.1. First we randomly generate values for $X_1, X_2, \ldots, X_{20}$.

In [22]:
```
from scipy.stats import geom
```

In [28]:
```
sample = geom.rvs(p=0.5, size=20)
sample
```

Out[28]:
```
array([2, 2, 2, 1, 1, 2, 2, 2, 3, 2, 2, 1, 1, 1, 1, 1, 1, 2, 2, 2])
```

In [29]:
```
# convert the numpy array into a pandas series
sample = pd.Series(sample)
sample
```

Out[29]:
```
0     2
1     2
2     2
3     1
4     1
5     2
6     2
7     2
8     3
9     2
10    2
11    1
12    1
13    1
14    1
15    1
16    1
17    2
18    2
19    2
dtype: int32
```

5.1.2. Next, we use these randomly generated values to randomly generate values for the sample statistics described above.

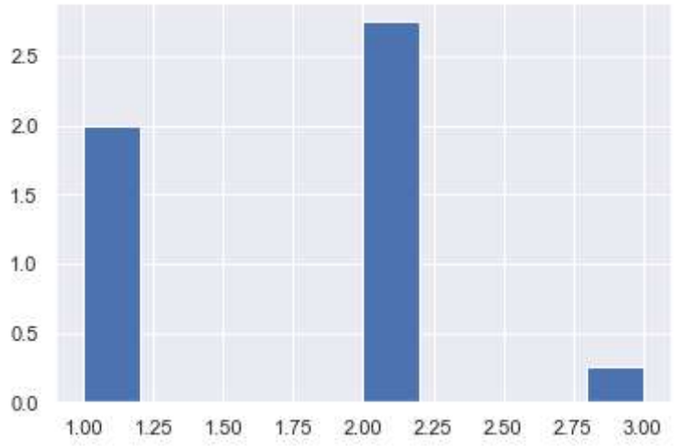
In [31]:
```
params = ['mean', 'median', 'std', 'prop <= 2']
samp = [sample.mean(),
        sample.median(),
        sample.std(),
        (sample<=2).mean()]
pd.DataFrame({'sample': samp}, index=params)
```

Out[31]:

	sample
mean	1.650000
median	2.000000
std	0.587143
prop <= 2	0.950000

5.1.3 Finally, we use these randomly generated values to create a randomly generated sample distribution.

In [32]:
```
sample.hist(density=True)
plt.show()
```



How do the sample statistics compare to the population parameters?

In [ ]: