Statistical Distributions with NumPy ¶

Imagine that you work as an admissions officer at a university. Part of your job is to collect, analyze, and visualize data that's relevant to interested applicants.

Recently, you've become interested in how histograms can show different distributions of populations and even occurences. You think that histograms would be useful in visualizing different trends, such as changes in department numbers and participation in extracurriculars. You also want to learn more about how you can use randomly generated distributions to make statistical calculations and predict the probability of future events, such as the sucess of your ultimate frisbee team.

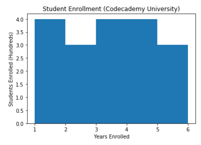
For this lesson, we'll be using NumPy to calculate distributions and Matplotlib to graph our calculations.

```
In [2]: import numpy as np
        from matplotlib import pyplot as plt
```

One set of data you want to analyze is enrollment in different degree programs. By looking at histograms of the number of years students are enrolled in a program, you can identify what programs are becoming more popular, which are falling out of favor, and which have steady, continual enrollment.

First, let's look at how many hundreds of students decide to enroll in Codecademy University and how many years they've been enrolled.

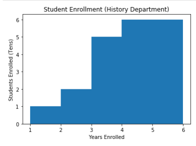
```
In [3]: total_enrollment = [1, 1, 1, 1, 2, 2, 2, 3, 3, 3, 3, 4, 4, 4, 4, 5, 5, 5]

        plt.hist(total_enrollment, bins=5, range=(1, 6))
        plt.title('Student Enrollment (Codecademy University)')
        plt.xlabel('Years Enrolled')
        plt.ylabel('Students Enrolled (Hundreds)')
        plt.show()
```



The histogram above shows the University's total enrollment, which is fairly consistent. This is a *uniform distribution* and is what the University wants to see. Total enrollment is staying at a good level.

Now let's take a look at the enrollment specifically for students seeking a degree in History:

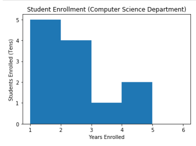
```
In [4]: history_enrollment = [1, 2, 2, 3, 3, 3, 3, 3, 4, 4, 4, 4, 4, 4, 5, 5, 5, 5, 5, 5]

        plt.hist(history_enrollment, bins=5, range=(1, 6))
        plt.title('Student Enrollment (History Department)')
        plt.xlabel('Years Enrolled')
        plt.ylabel('Students Enrolled (Tens)')
        plt.show()
```



What does this histogram tell us? Well this is somewhat *skewed left* dataset, we can see that there are a lot more students who have been enrolled for 3 or 4 years over 1 and 2 years. This indicates that the History program is becoming less popular. Where are all the students going then?

The school recently invested a lot of money in a new building for the Computer Science Department. Let's take a look at enrollment and see if the investment is paying off.

```
In [5]: cs_enrollment = [1, 1, 1, 1, 1, 2, 2, 2, 2, 3, 4, 4]

        plt.hist(cs_enrollment, bins=5, range=(1, 6))
        plt.title('Student Enrollment (Computer Science Department)')
        plt.xlabel('Years Enrolled')
        plt.ylabel('Students Enrolled (Tens)')
        plt.show()
```

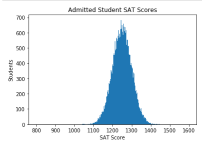


It looks like enrollment has skyrocketed for the Computer Science department in recent years. This could be because the University invested in the department, or it could be a sign that the sought after job skills in the real world are changing. Whatever the reason, the histograms let us clearly see the trends.

Interested applicants would like to know what kinds of SAT scores accepted students had. You've previously calculated that the mean score is 1250, with a standard deviation of 50.

Rather than gather every students score, you take what you know about the data and use a random number generator to generate a model.

```
In [6]: sat_scores = np.random.normal(1250, 50, size=100000)

        plt.hist(sat_scores, bins=1000, range=(800,1600))
        plt.title('Admitted Student SAT Scores')
        plt.xlabel('SAT Score')
        plt.ylabel('Students')
        plt.show()
```



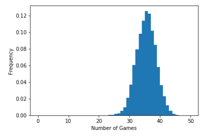
95% of Students score within two standard deviations of the mean. An interested student scores an 1130 and wants to know if they are within that range.

```
In [7]: mean = 1250
        one_std = 50

        two_below = (mean - 2*one_std)
        print two_below

        1150
```

Looks like they're just below it! Better re-take that test.

One of the big draws to your school is your excellent ultimate frisbee team. The team wins about 70% of their 50 games each season, or 35 games. An interested applicant wants to know what the chance is that they could improve their record to 40 games. You use what you know about binomial distributions to calculate the probability of such an occurence:

```
In [8]: ultimate = np.random.binomial(50, 0.70, size=10000)
        plt.hist(ultimate, range=(0, 50), bins=50, normed=True)
        plt.xlabel('Number of Games')
        plt.ylabel('Frequency')
        plt.show()
```



Since it's a little hard to see from the graph, let's calculate exactly what chance they have of winning 40 games:

```
In [10]: ultimate = np.random.binomial(50, 0.70, size=10000)
         np.mean(ultimate == 40)

Out[10]: 0.041000000000000002
```

Hmm, looks like it might be tough for the team to reach that number of wins, given the current data - but even more of a reason for this applicant to sign up and help the team improve!