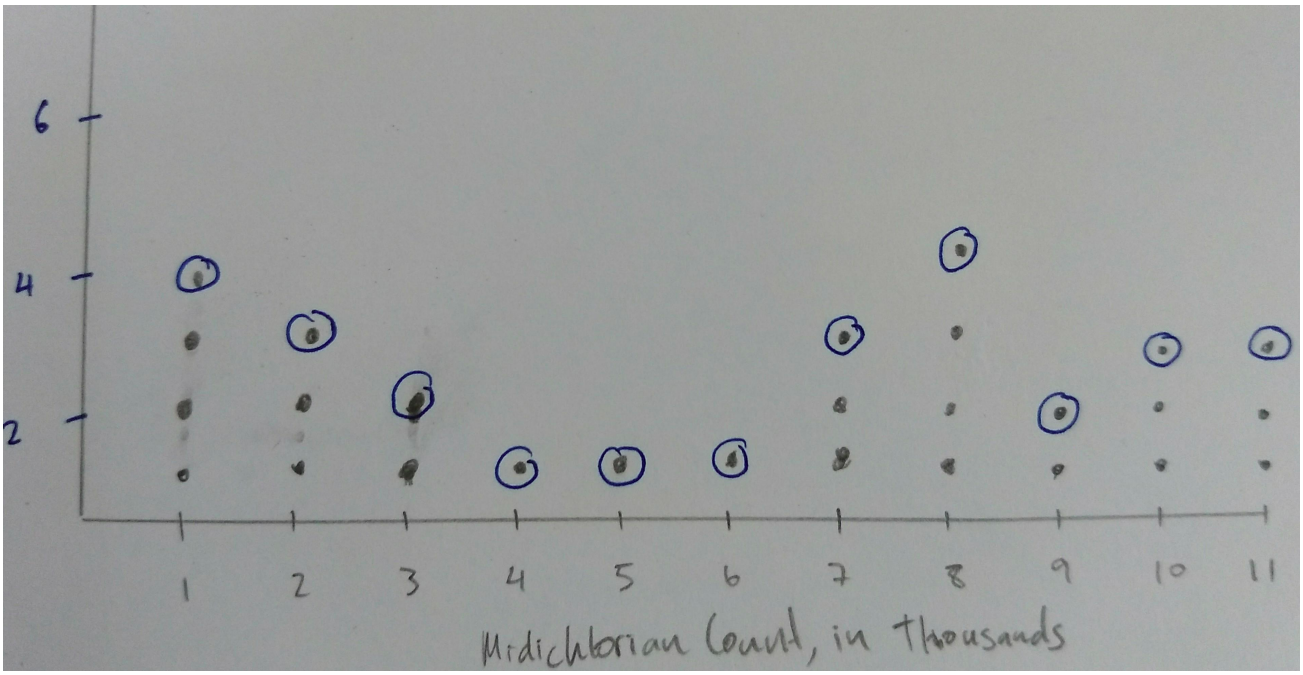
An Intuitive Introduction to Kernel Density Estimators

Say that, unfortunately, only sample of Berkeley undergrads are die-hard Star Wars fans who know their midichlorian counts. How can we represent the density of this sample?



What a nice dot plot. But what if we wanted to use this sample to estimate the midichlorian count density of the entire UC Berkeley undergraduate population?

Let us assume that the following sample is representative of the population, at least when it comes to midichlorian counts.

In [1]:
```
fan_mcounts = [19, 22, 3, 20, 11, 1, 8, 7, 15, 12, 9, 23, 18, 8, 5, 25, 14, 19, 2,
21]
```

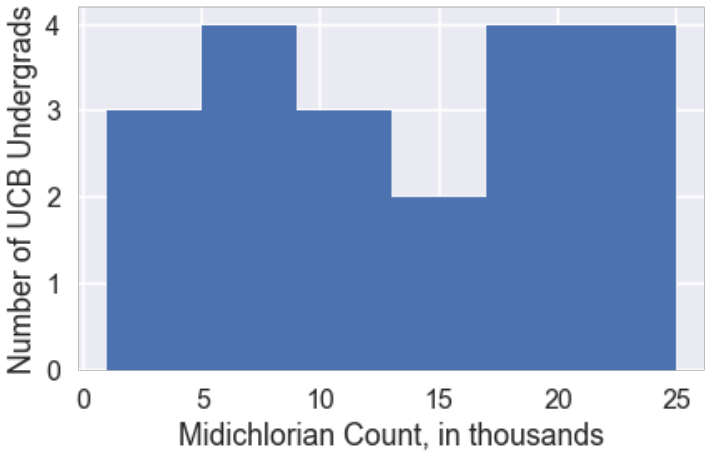
What methods did we learn in elementary school to visualize and estimate the density of a one-dimensional numeric variable? Histograms!

In [2]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

plt.style.use('fivethirtyeight')
sns.set()
sns.set_context("poster")
%matplotlib inline
```

In [3]:
```
plt.hist(fan_mcounts, 6)
plt.xlabel("Midichlorian Count, in thousands")
plt.ylabel("Number of UCB Undergrads")
```

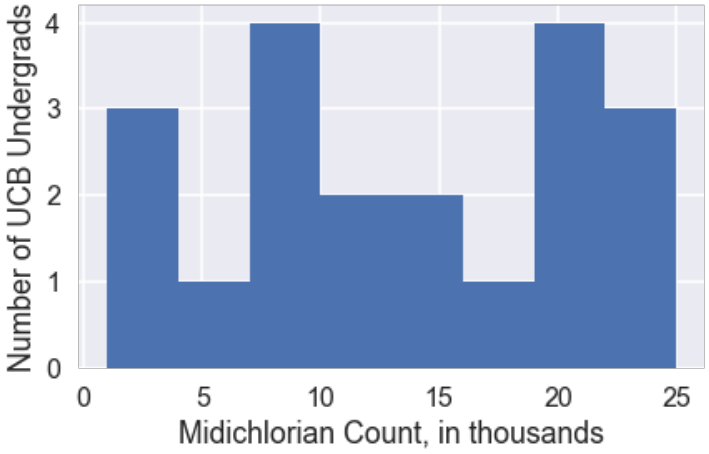
Out[3]: <matplotlib.text.Text at 0x10ea67cf8>



However, histograms are sensitive to parameter choices, as we can see by simply increasing the number of bins by 2:

In [4]:
```
plt.hist(fan_mcounts, 8)
plt.xlabel("Midichlorian Count, in thousands")
plt.ylabel("Number of UCB Undergrads")
```

Out[4]: <matplotlib.text.Text at 0x10ebc1eb8>



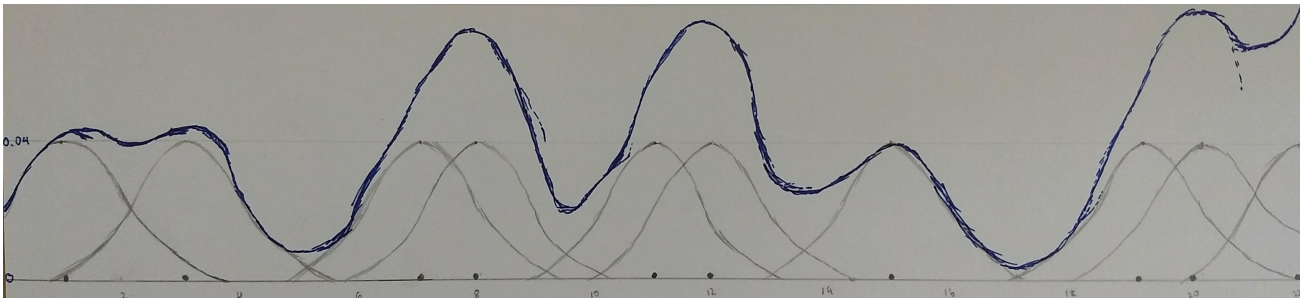
Other parameters include bin offset and bin openness. The caveats of histograms are explored in much more detail in this blog post: http://tinlizzie.org/histograms/ (http://tinlizzie.org/histograms/). How can we improve upon this?

Enter kernel density estimators!

Intuitively, they work very similarly to histograms. We could say that histograms are simple density estimators, while kernel density estimators are like smoothed histograms. While histograms use strict binning, kernel density estimators sum up the heights of "kernels", which are identically distributed curves placed at each data point. Applying the kernel turns the one data point into a weighted average of several x-values before and after the current datapoint to help determine the y-value of the current datapoint--this is what we call smoothing.

The following image shows how a Kernel Density Estimator

1. applies a kernel centered at each datapoint
2. sums up the heights of the kernel to form the final KDE curve



In the illustration above, we used a Gaussian (normal) function as the kernel function. The kernel determines the *shape* of the curve, but it also has a smoothing parameter called the bandwidth that determines how far left and right to consider x-values, and how to weigh the different x-values.

The bandwidth of the kernel function is the only free parameter for kernel density estimators, so though we still have to pay attention to how our bandwidth selection affects our estimation, we have fewer parameters to worry about than with histograms.

In [ ]: