Histogram Workshop

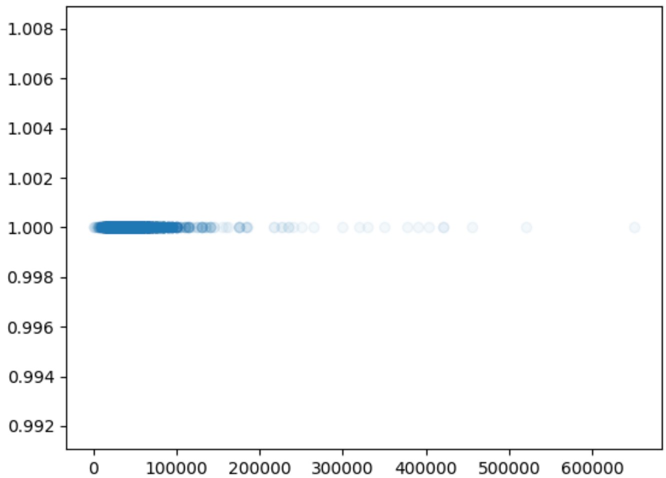
January 17, 2022

Reading Data

The file `salary.txt` contains 10000 yearly salaries obtained from job ads posted in March 2019 on an online job board. Put the data file `salary.txt` in the same directory you are going to run code from. This data file can be read in a number of ways, for example

```
with open("salary.txt", 'r') as infile:
    data = [ float(s) for s in infile.readlines() ]
```

Exercise 1: Download the file `salary.txt` and read it into an array called `x`. Create an array, `y`, of the same length with all the entries set to 1. Make a scatter plot of `x` against `y` and give each point a low transparency (`alpha`) value. You should get something like



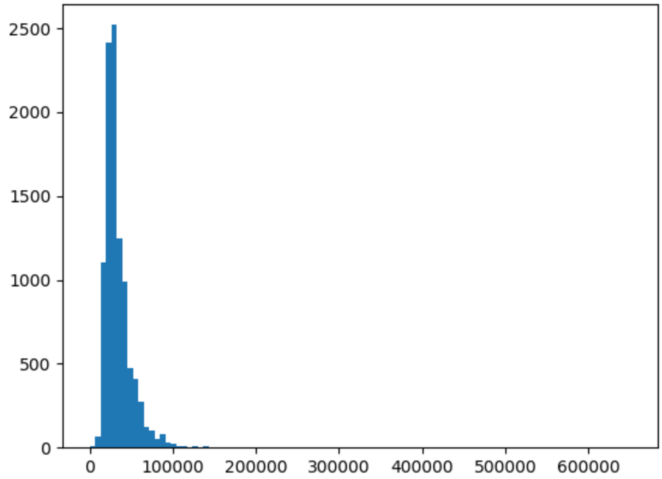
This is called a **rug plot**, it is usually combined with a histogram.

Binning and Histograms

A useful way to examine our data is to ask questions like "What proportion of jobs have salaries in the range £25000 to £30000?". Choosing an increment and counting up all the data points that lie in each interval is called **binning** and the plot of the counts in each bin is called a **histogram**. Matplotlib has a function which constructs histograms from lists of data points.

```
plt.hist(data, bins = 100)
plt.show()
```

Should produce this plot:



The argument `bins = 100` splits the **range** (the minimum to maximum salary) into 100 equal sized bins.

Exercise 2: Produce the plot above. Try using different numbers of bins and see how the histogram changes. Look at `https://matplotlib.org/api/_as_gen/matplotlib.pyplot.hist.html` and study the different arguments that the `hist` function takes.

Variable Bins

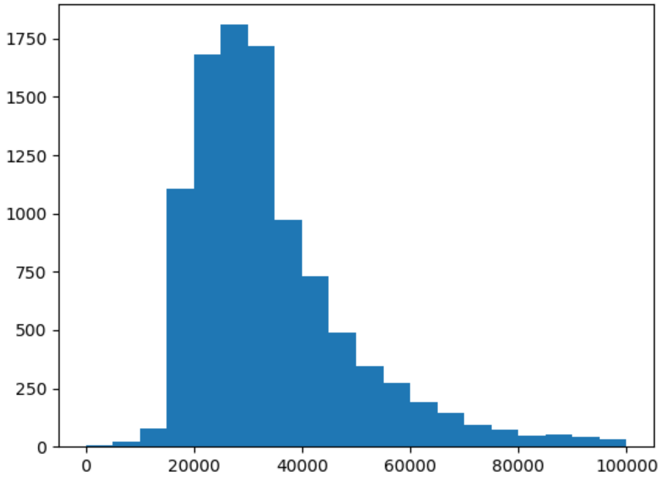
We can see from the histogram that the frequency/probability of jobs with salaries higher than £100,000 is quite low. Most of our data is squashed into the left hand side. To focus on only the jobs paying less than £100,000 per year we can pass a list instead of an integer for the `bins` argument. The following code constructs bins of size 5,000, from 0 up to 100,000.

```
bins = [ 5000*i for i in range(21) ]
```

Do you understand what it does?

```
plt.hist(data, bins = bins)
plt.show()
```

Gives:

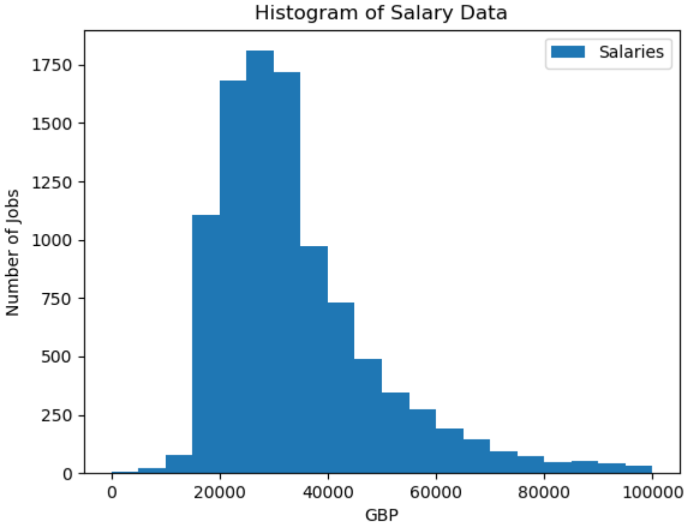


Exercise 3: Produce the plot above.

Making Your Plots Look Pretty

So far none of our plots have had labels, it is time to fix that. Matplotlib allows all kinds of annotations:

```
plt.hist(data, bins = bins, label = "Salaries")
plt.xlabel("GBP")
plt.ylabel("Number of Jobs")
plt.legend()
plt.title("Histogram of Salary Data")
plt.savefig("salary_histogram.png");
```



Note we are using `plt.savefig` instead of `plt.show` which will save the plot in a file with the name specified. We can call `plt.show` as well, but it must be called **after** `plt.savefig`.

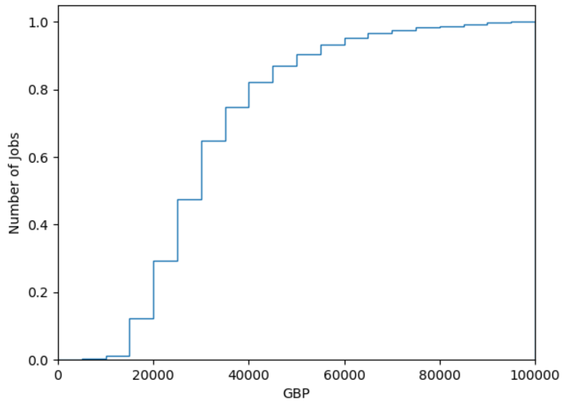
Exercise 4: Produce the plot above. Look at the documentation for `label` and `legend`. Restrict the range of the x-axis to only show salaries between £0 and £100000 and put the legend in a different place.

Exercise 5: Minimum wage laws make an annual salary of less than £5000 per year unlikely. Using bins of size 500, plot the histogram in the range 0 to 10000. What do you think is going on?

Cumulative Frequency Histogram

We can make another kind of probability plot called a **cumulative frequency distribution** or, if we normalise it, a **cumulative probability function**. This shows the number of data points less than x as a function of x.

```
plt.hist(data, bins=bins, cumulative=True, histtype='
    step', density=True)
```



The `histtype` argument changes the appearance of the histogram bars. From this plot is is easy to read that e.g. 80% of jobs have salaries less than £40000 per year. Cumulative probability distributions are very useful for mathematical analysis, but can be less intuitive than probability distributions as a means of visualising data, they often look very similar even if the probability distributions look very different.

Exercise 6: Use the cumulative frequency distribution to estimate (you don't have to be exact) the 1% salary i.e. the amount of money you have to earn per year so that you make more than 99% of people.

Exercise 7: Use the cumulative frequency distribution to estimate the salary **quartiles**. The lower quartile splits off the bottom 25% of the data, the second quartile splits off the bottom 50% and the upper quartile splits off the bottom 75%.

Exercise 8: Estimate the quartiles and the 99^{th} percentile more precisely by sorting the data.

Stem and Leaf plot

An 'old-fashioned' alternative to a histogram is the so-called *stem and leaf plot*. I don't recommend these for data visualisation but it is interesting to see them. They are a kind of histogram which can be useful for summarising a list of not too many (say between 20 and 100) data points. It's best illustrated by example. Let's use the data from the Online Stats Book

```
data = [37, 33, 33, 32, 29, 28, 28, 23, 22, 22, 22, 21,
     21, 21, 20, 20, 19, 19, 18, 18, 18, 18, 16, 15,
    14, 14, 14, 12, 12, 9, 6]
```

The 'stem' is the 10s digit, and the the 'leaf' is the 1s digit. e.g. the number 45 the stem is 4 and the leaf is 5. Obviously you can define stems and leaves using any multiples you like, but 10 and

1 is most common. The stem and leaf plot collects all the ;eaves with the same stem. The data above is transformed into:

```
3  |  2337
2  |  001112223889
1  |  2244456888899
0  |  69
```

Exercise 9 (Optional): Write python code to produce the stem and leaf plot above.