Sampling and Standard Error

Inspired by MIT OpenCourseWare "Sampling and Standard Error" by Prof John Guttag, Lecture 8 in the 2016 series *Introduction to Computational Thinking and Data Science.*

The MIT lecture video: https://ocw.mit.edu/courses/6-0002-introduction-to-computational-thinking-and-data-science-fall-2016/resources/lecture-8-sampling-and-standard-error/

Python code and Temperature data in file Lecture8.zip from https://ocw.mit.edu/courses/6-0002-introduction-to-computational-thinking-and-data-science-fall-2016/resources/lecture8/

Setup

For information about getting started, see the getting_started file in the folder above this one.

This script uses a file called `temperature.csv` , downloaded from the link above. The file needs to be on your MATLAB path or in the expected location.

```
% Set the filename for the downloaded file: temperatures.csv
rpathData = 'data/MITOCW/IntroCompThinkingandDataScienceLecture8' ;

fileData = 'temperatures.csv' ;

dataFilename = fullfile(rpathData, fileData) ;

% Check file is on path, if not try setting automatically
if ~exist(dataFilename,"file")
    pathThisScript  = fileparts(matlab.desktop.editor.getActiveFilename) ;
    pathAboveScript = fileparts(pathThisScript) ;
    pathData = fullfile( pathAboveScript, rpathData) ;
    dataFilename = fullfile(pathData, fileData)  ;
end
```

Check we can see the file. (No errors in file or path names.)

```
% Check data file is accessible
if ~exist(dataFilename, "file")
    warning(['Check path or name for file: ', dataFilename])
    disp("Current folder is: "+pwd)
end
```

The Population

It is always good to visualise data. Read it in as a table, and then just plot it.

```
T = readtable( dataFilename )  % displays a summary of the data as a
scrollable table
```

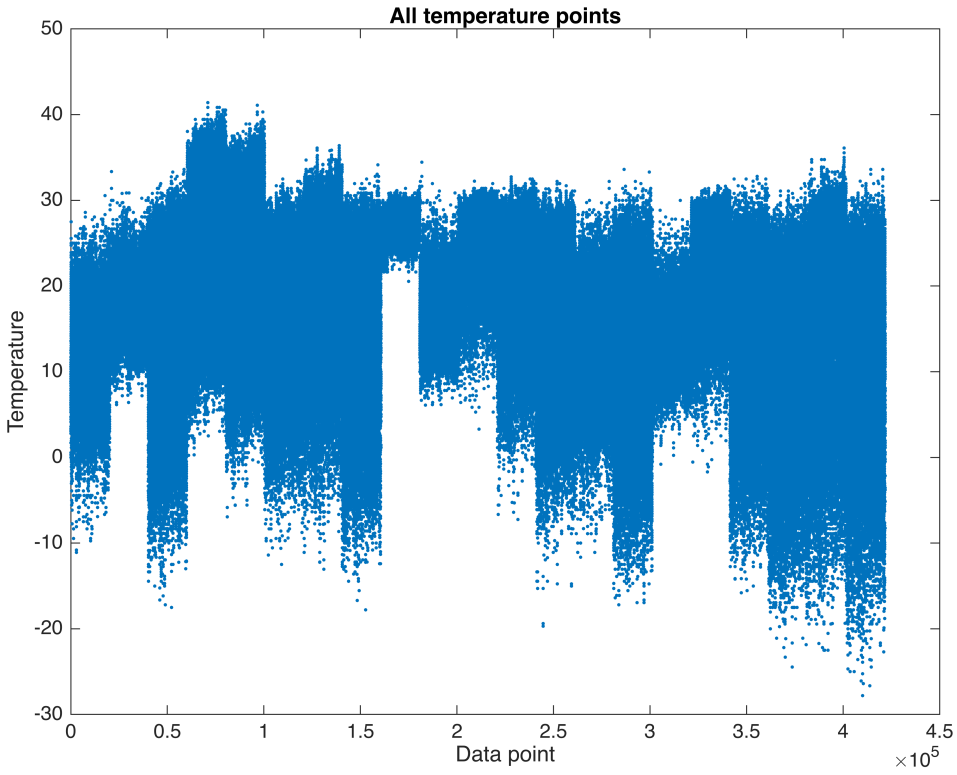
T = 421848×3 table

	CITY	TEMP	DATE
1	'SEATTLE'	3.1	19610101
2	'SEATTLE'	0.55	19610102
3	'SEATTLE'	0	19610103
4	'SEATTLE'	4.45	19610104
5	'SEATTLE'	8.35	19610105
6	'SEATTLE'	6.7	19610106
7	'SEATTLE'	9.7	19610107
8	'SEATTLE'	7.2	19610108
9	'SEATTLE'	9.45	19610109
10	'SEATTLE'	10.3	19610110
11	'SEATTLE'	7.8	19610111
12	'SEATTLE'	6.35	19610112
13	'SEATTLE'	7.8	19610113
14	'SEATTLE'	11.1	19610114

⋮

Although there are 421848 data points, it is still possible to just use the basic `plot` command,

```
figure(Name='All Temperature Points')
plot(T.TEMP,'.')
xlabel('Data point'), ylabel('Temperature'), title('All temperature points')
```

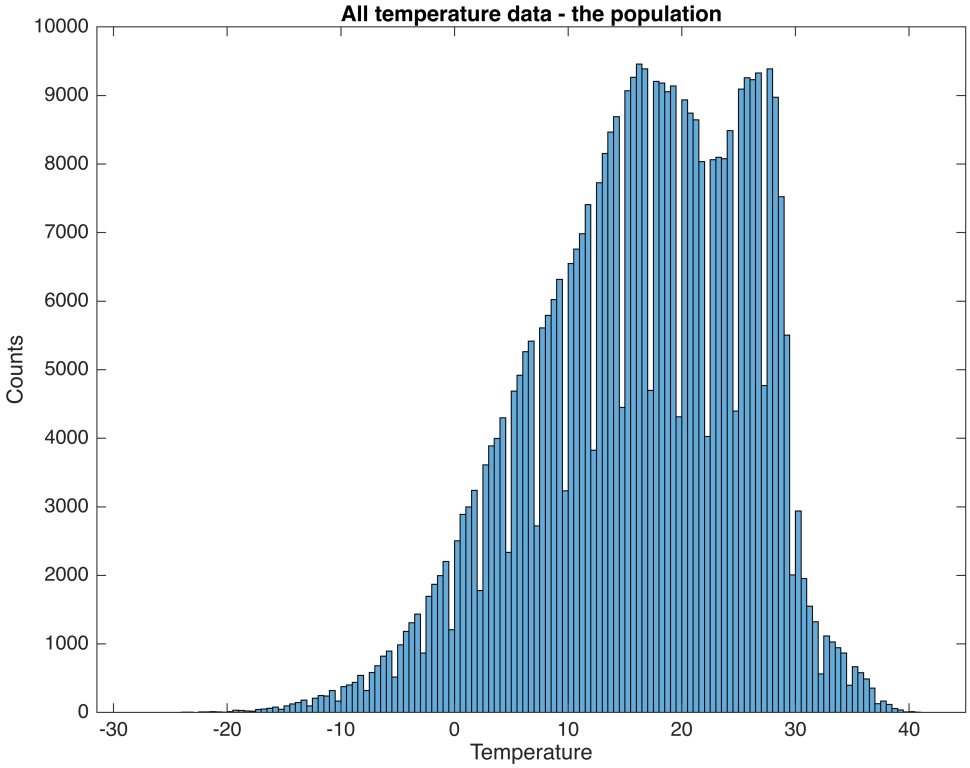


Although there are too many points to resolve clearly, we can see by eye that the "middle" temperature is somewhere in the region of 20 and the spread around this is about +/- 10. These give hints as to the mean and standard deviation of the data. Also, the table summary suggests the data is listed by city and we can see from the plot there seems to be at least one city with a small temperature variation, but we will ignore this for now.

Histogram

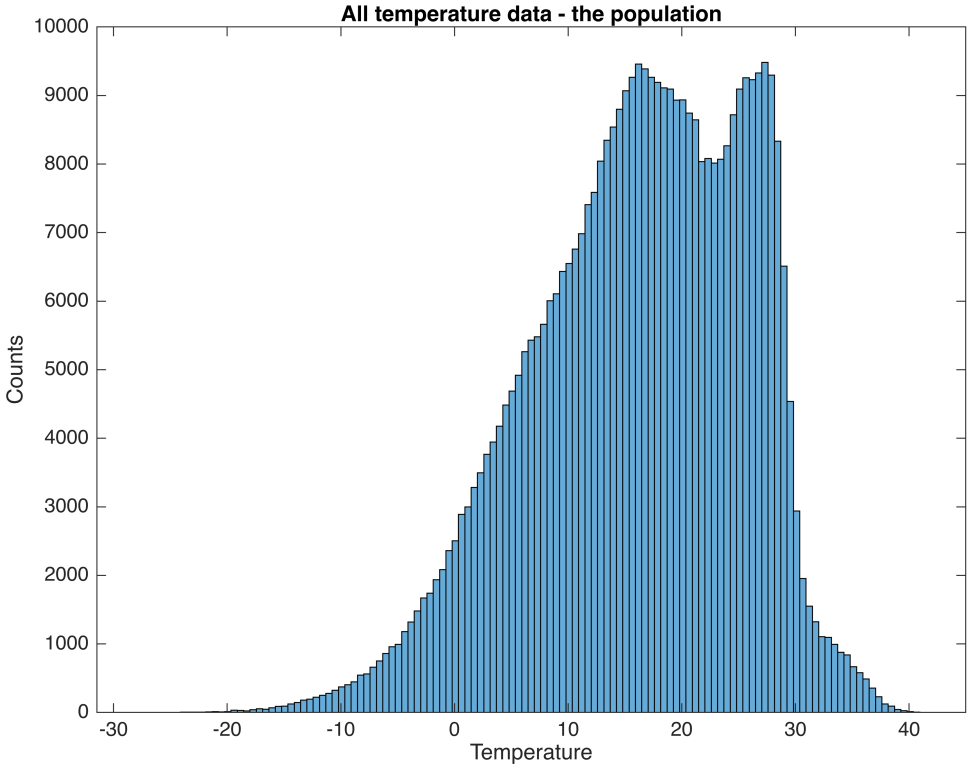
Display a histogram of all temperature points - here we call all the points the **population**

```
figure(Name="Histogram of all temperature data")
hAllTemp = histogram(T.TEMP) ;
xlabel('Temperature'), ylabel('Counts'), title('All temperature data - the
population')
```



There appears to be something a bit odd in the histogram (the sudden dips), which could be because of rounding or other discretisation issues. We can use fewer histogram bins:

```
nbins = fewerbins(hAllTemp) ;
```

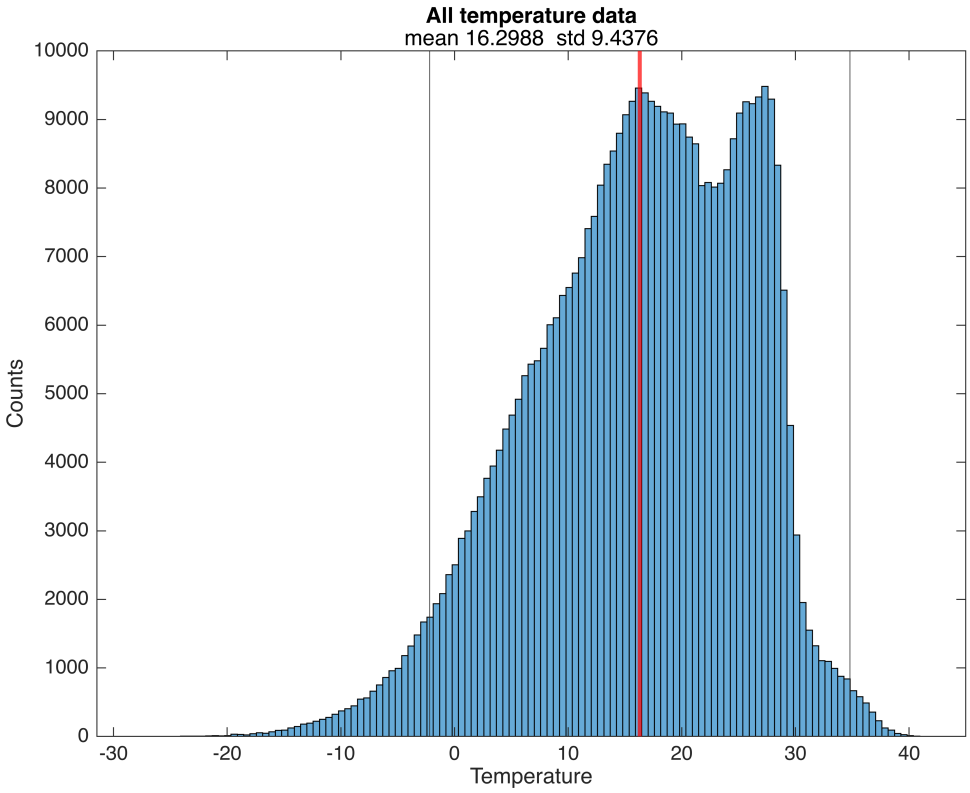


Note the distribution is only very roughly normal (gaussian) in shape.

Mean and Standard Deviation of the Population

Find the mean and standard deviation of the population

```
meanPop = mean(T.TEMP) ;
stdPop  =  std(T.TEMP)  ;
% Update the histogram title with this mean and standard deviation
title("All temperature data", "mean " + meanPop + "  std " + stdPop)

% draw vertical lines at mean and +/- 1.96 stddev
xline(meanPop, LineWidth=2, Color='r')
xline(meanPop - 1.96*stdPop)
xline(meanPop + 1.96*stdPop)
```



We see that the mean of all the data (the population mean) is 16.30 and standard deviation is 9.44.

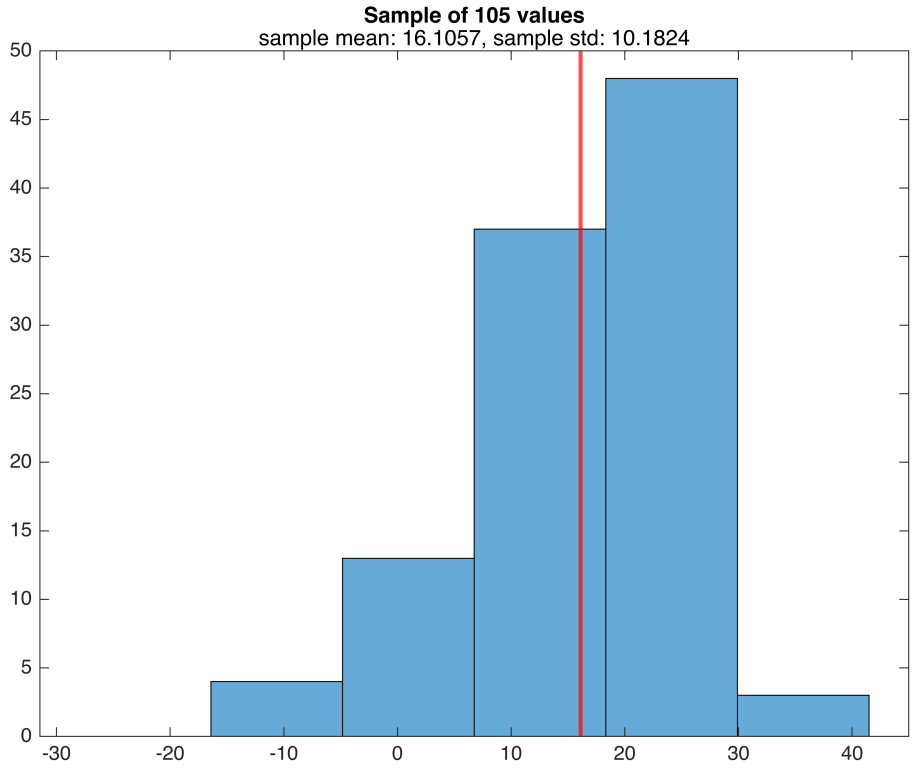
Sampling

In many cases, collecting data is difficult and / or expensive so we sample only a smaller set of data points.

To simulate sampling only `num_in_sample` points, we pick out temperatures points at random from the whole population. The population data has been read into a table with rows numbered in the range from 1 to 421848. We first create a list of integers, `randIndices`, drawn at random from this range, and then use the list to pick out the actual temperature values from the corresponding rows of the table.

```
num_in_sample = 105 ;
nPop = length(T.TEMP) ;

% Get num_in_sample random integers in range 1 to nPop
randIndices = randi(nPop, [num_in_sample 1]) ;

% Pick out the temperatures from the table rows listed in randIndices
sampleTemperatures = T.TEMP(randIndices) ;

figure(Name='Histogram of sampled points')
hSample = histogram(sampleTemperatures) ;
% set x axis limits to be the same as population histogram
hSample.BinLimits = hAllTemp.BinLimits ;

fewerbins(hSample) ;
```

```
meanSample = mean( sampleTemperatures ) ;
 stdSample =  std( sampleTemperatures ) ;

xline(meanSample, LineWidth=2, Color='r')
title("Sample of "+num_in_sample+" values", ...
      "sample mean: "+meanSample+", sample std: "+stdSample)
```

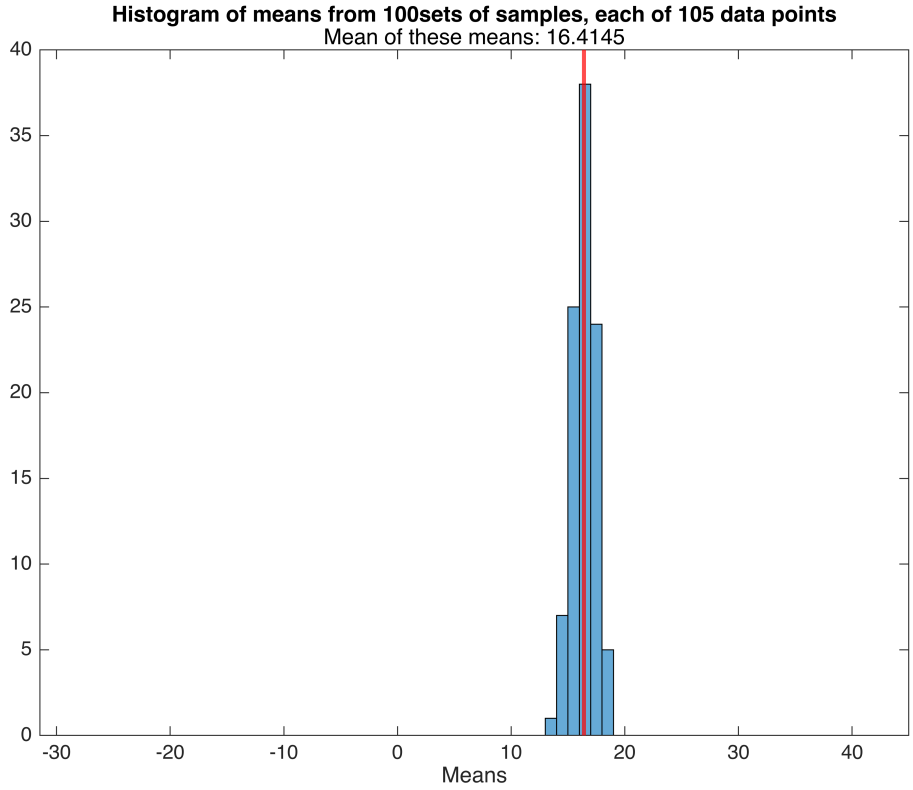


The distribution of this sample is roughly the same as the population and it has a similar mean. How similar is this mean?

As this is a simulation, we are able to experiment with repeating the sampling. What if we repeat the sampling multiple times with different random combinations of values from the population?

```
num_repeat_samples = 100 ;
num_in_sample = 105 ;

repeatedSample_means = zeros([num_repeat_samples 1]) ;

for irepeat = 1:num_repeat_samples
    randIndices = randi(nPop, [num_in_sample 1]) ;  %as above, but this
will be a new set of indices
    temperatures_this_repeat = T.TEMP(randIndices) ;
    repeatedSample_means(irepeat,1) = mean(temperatures_this_repeat) ;
```

```
end

figure(Name='Sample Means')
hSampleMeans = histogram(repeatedSample_means) ;
mean_repeatedSample_means = mean( repeatedSample_means ) ;
xline(mean_repeatedSample_means, LineWidth=2, Color="r")
title("Histogram of means from "+num_repeat_samples+"sets of samples, each
of "+ ...
    num_in_sample+" data points", "Mean of these means:
"+mean_repeatedSample_means)
xlabel("Means")
hSampleMeans.BinLimits = hAllTemp.BinLimits ;
```



This "mean of means" is close to the true population mean and has a perhaps suprisingly narrow range of values. The distribution of these means is close to a normal distribution - the Central Limit Theorem - though it is not easy to see on this scale.

Standard Error of the Mean

The Standard Error of the Mean is SEM = `stdPop / sqrt( num_in_sample )`

```
disp("Standard Error of the Mean = " + stdPop/sqrt(num_in_sample))
```

```
Standard Error of the Mean = 0.92101
```

This tells us how much error there might be in a mean taken from a limited sized sample.

At first sight, this number is remarkably small. We have 412848 measurements and yet if we only sample 105 of these, we get an estimate of the true population mean within about 1 degree C. The reason for this is that roughly speaking, all we are doing is finding the middle of the population distribution, so all we need to do is sample enough points to roughly map the distribution and then we can calculate its mean.

For more details and further examples, watch the lecture video linked at the top.

David Atkinson, December 2023.