LESSON: Sampling

FOCUS QUESTION: Do the characteristics of a sample reflect an entire population?

In this lesson you will:
- Simulate random sampling.
- Investigate how sample size effects accuracy of population estimates.
- Understand SEM (standard error of the mean).
- Understand 95% confidence intervals.

Contents
- SETUP for the SAMPLING LESSON
- EXAMPLE 1: Create a collection of 1000 samples of N(0,1), each of size 10
- EXAMPLE 2: Calculate the sample means
- EXAMPLE 3: Show the distribution of sample means
- EXAMPLE 4: Calculate the actual and unbiased sample standard deviations
- EXAMPLE 5: Calculate the estimated standard error of the mean (SEM) for each sample
- EXAMPLE 6: Output times the true population mean is above SEM error bar
- EXAMPLE 7: Output times true population mean is above 95% confidence interval
- EXAMPLE 8: Output times actual and unbiased sample stds underestimate pop std
- SUMMARY OF SYNTAX

SETUP for the SAMPLING LESSON
- Create a Sampling directory on your V: drive and make it your current directory.
- Create a SamplingLesson.m script file in your Sampling directory. Enter each of the examples in a new cell in this script.

EXAMPLE 1: Create a collection of 1000 samples of N(0,1), each of size 10

Create a new cell in which you type and execute:

```matlab
sampleSize = 10;
popStd = 1;
popMean = 0;
numSamples = 1000;
samples = random('norm', popMean, popStd, sampleSize, numSamples);
```

You should see the following 5 variables in your Workspace Browser:
- `numSamples` - variable containing the number of samples to generate
- `popMean` - actual mean of the underlying population
- `popStd` - actual standard deviation of the underlying population
- `samples` - an array in which each column corresponds to a random sample of 10 values drawn at random from the normal distribution which has 0 mean and standard deviation 1. The `samples` array has 1000 columns corresponding to 1000 samples.
- `sampleSize` - variable containing the size of the individual samples

**EXAMPLE 2: Calculate the sample means**

Create a new cell in which you type and execute:

```matlab
    sampleMeans = mean(samples); % Means of the samples
```

You should see the following variable in your Workspace Browser:

- `sampleMeans` - a row vector with the mean of each sample

**EXERCISE 1: Calculate and output the average of the sample means.**

**EXAMPLE 3: Show the distribution of sample means**

Create a new cell in which you type and execute:

```matlab
    figure
colormap summer
hist(sampleMeans)
xlabel('Value');
ylabel('Frequency');
title(['Sample mean distribution (sample size=' num2str(sampleSize) ')'])
```

You should a Figure Window with a histogram:
EXAMPLE 4: Calculate the actual and unbiased sample standard deviations

Create a new cell in which you type and execute:

```
actualSampleStds = std(samples, 1); % RMS errors of the samples from their mean
unbiasedSampleStds = std(samples); % Unbiased sample standard deviations
```

You should see the following variables in your Workspace Browser:

- `actualSampleStds` - row vector with the actual sample standard deviations
- `unbiasedSampleStds` - row vector with the unbiased sample standard deviations

EXERCISE 2: Calculate and output the true SEM of the sample means

The SEM (Standard Error of the Mean) is the true population standard deviation (`popStd`) divided by the square root of the sample size. Statisticians have shown that the actual standard deviation of the population of all possible sample means is the original population standard deviation divided by the square root of the sample size. In most cases, we don't actually know the true standard deviation of the original population, but in this case we know it exactly because we are creating data.

EXAMPLE 5: Calculate the estimated standard error of the mean (SEM) for each sample

Create a new cell in which you type and execute:

```
sampleSEMs = unbiasedSampleStds./sqrt(sampleSize);
```
You should see the following variable in your Workspace Browser:

- sampleSEMS - a row vector the SEM for each sample

**Note:** In real life, we don't know the true population standard deviation, so we can't calculate the SEM exactly. Instead we estimate the SEM for each sample based on the unbiased standard deviation. It's the best we can do.

**EXAMPLE 6: Output times the true population mean is above SEM error bar**

Create a new cell in which you type and execute:

```matlab
timesAbove = (popMean > sampleMeans + sampleSEMs); fprintf('Times actual population mean above SEM error bars: %g\n', sum(timesAbove)); fprintf('Fraction actual above SEM error bars: %g', mean(timesAbove));
```

You should see the following variable in your Workspace Browser:

- timesAbove - a logical vectors with ones in positions where population mean is above the sum of the sample mean plus sample SEM.

You should also see the following output:

Times actual population mean above SEM error bars: 178
Fraction actual above SEM error bars: 0.178

**EXERCISE 3: Output times the true population mean is below SEM error bar**

Also find and output the fraction of times the true population mean is below the SEM error bar.

**EXAMPLE 7: Output times true population mean is above 95% confidence interval**

Create a new cell in which you type and execute:

```matlab
confInt95 = 1.96*unbiasedSampleStds./sqrt(sampleSize); timesAbove95 = (popMean > sampleMeans + confInt95); fprintf('Times actual population mean above 95%% CI error bars: %g\n', sum(timesAbove95)); fprintf('Fraction actual above 95%% CI error bars: %g', mean(timesAbove95));
```

You should see the following variable in your Workspace Browser:

- confInt95 - sizes of upper wings of 95% confidence intervals
- timesAbove95 - times actual pop mean above 95% confidence interval

You should also see the following output:

Times actual population mean above 95% CI error bars: 55
Fraction actual above 95% CI error bars: 0.055

**EXERCISE 4: Output times true population mean below 95% CI error bar**

Also find and output the fraction of times the true population mean is below the 95% CI error bars.

**EXERCISE 5: Output times true population mean outside 95% CI error bar**

**EXAMPLE 8: Output times actual and unbiased sample stds underestimate pop std**

Create a new cell in which you type and execute:

```matlab
unbiasedBelow = (popStd > unbiasedSampleStds);
fprintf('Times unbiased sample std underestimates pop std: %g\n', ...
    sum(unbiasedBelow));
fprintf('Fraction unbiased sample std underestimates: %g\n', ...
    mean(unbiasedBelow));

actualBelow = (popStd > actualSampleStds);
fprintf('Times actual sample std underestimates population std: %g\n', ...
    sum(actualBelow));
fprintf('Fraction actual sample std underestimates: %g\n', ...
    mean(actualBelow));
```

You should see the following variables in your Workspace Browser:

- unbiasedBelow - a logical vectors with ones where unbiased below
- actualBelow - a logical vectors with ones where actual below

You should also see the following output:

Times unbiased sample std underestimates pop std: 596
Fraction unbiased sample std underestimates: 0.596
Times actual sample std underestimates population std: 687
Fraction actual sample std underestimates: 0.687

**SUMMARY OF SYNTAX**

<table>
<thead>
<tr>
<th>MATLAB syntax</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>A &gt; B</td>
<td>Return an array of 0's and 1's that is the same size as the arrays A and B. The element has a value of 1 if the corresponding element of A is greater than the corresponding element of B.</td>
</tr>
</tbody>
</table>

*This lesson was written by Kay A. Robbins of the University of Texas at San Antonio and last modified on April 1, 2015. Please contact kay.robbins@utsa.edu with comments or suggestions. The image is a photograph of a nocturnal instrument photographed by Michael Daly on 8/22/2009. The image is available on Wikipedia as http://commons.wikimedia.org/wiki/File:AUGUSTUS_RIC_I_359-78001668.jpg.*