```yaml
Course: Statistical_Inference
Lesson: Expectations

- Class: text
  Output: "Expectations. (Slides for this and other Data Science courses may be found at github https://github.com/DataScienceSpecialization/courses/. If you care to use them, they must be downloaded as a zip file and viewed locally. This lesson corresponds to 06_Statistical_Inference/04_Expectations.)"

- Class: text
  Output: In this lesson, as you might expect, we'll discuss expected values. Expected values of what, exactly?

- Class: text
  Output:  The expected value of a random variable X, E(X), is a measure of its central tendency. For a discrete random variable X with PMF p(x), E(X) is defined as a sum, over all possible values x, of the quantity x*p(x). E(X) represents the center of mass of a collection of locations and weights, {x, p(x)}.

- Class: text
  Output: Another term for expected value is mean. Recall your high school definition of arithmetic mean (or average) as the sum of a bunch of numbers divided by the number of numbers you added together. This is consistent with the formal definition of E(X) if all the numbers are equally weighted.

- Class: cmd_question
  Output: Consider the random variable X representing a roll of a fair dice. By 'fair' we mean all the sides are equally likely to appear. What is the expected value of X?
  CorrectAnswer: 3.5
  AnswerTests: equiv_val(3.5)
  Hint: Add the numbers from 1 to 6 and divide by 6.

- Class: cmd_question
  Output: We've defined a function for you, expect_dice, which takes a PMF as an input. For our purposes, the PMF is a 6-long array of fractions. The i-th entry in the array represents the probability of i being the outcome of a dice roll. Look at the function expect_dice now.
  CorrectAnswer: expect_dice
  AnswerTests: omnitest(correctExpr='expect_dice')
  Hint: Type 'expect_dice' at the command prompt.

- Class: cmd_question
  Output: We've also defined PMFs for three dice, dice_fair, dice_high and dice_low. The last two are loaded, that is, not fair. Look at dice_high now.
  CorrectAnswer: dice_high
  AnswerTests: omnitest(correctExpr='dice_high')
  Hint: Type 'dice_high' at the command prompt.

- Class: cmd_question
  Output: Using the function expect_dice with dice_high as its argument, calculate the expected value of a roll of dice_high.
  CorrectAnswer: expect_dice(dice_high)
  AnswerTests: omnitest(correctExpr='expect_dice(dice_high)')
  Hint: Type 'expect_dice(dice_high)' at the command prompt.

- Class: cmd_question
  Output: See how the expected value of dice_high is higher than that of the fair dice. Now calculate the expected value of a roll of dice_low.
  CorrectAnswer: expect_dice(dice_low)
  AnswerTests: omnitest(correctExpr='expect_dice(dice_low)')
  Hint: Type 'expect_dice(dice_low)' at the command prompt.

- Class: text
  Output: You can see the effect of loading the dice on the expectations of the rolls. For high-loaded dice the expected value of a roll (on average) is 4.33 and for low-loaded dice 2.67. We've stored these off for you in two variables, edh and edl. We'll need them later.

- Class: text
```

```
  Output: One of the nice properties of the expected value operation is that it's
  linear. This means that, if c is a constant, then E(cX) = c*E(X). Also, if X and Y
  are two random variables then E(X+Y)=E(X)+E(Y). It follows that E(aX+bY)=aE(X)+bE(Y).

- Class: cmd_question
  Output: Suppose you were rolling our two loaded dice, dice_high and dice_low. You can
  use this  linearity property of expectation to compute the expected value of their
  average. Let X_hi and X_lo represent the respective outcomes of the dice roll. The
  expected value of the average is E((X_hi + X_lo)/2) or .5 *( E(X_hi)+E(X_lo) ).
  Compute this now. Remember we stored the expected values in edh and edl.
  CorrectAnswer: 3.5
  AnswerTests: equiv_val(3.5)
  Hint: Type '.5*(edh+edl)' at the command prompt.

- Class: mult_question
  Output: Did you expect that?
  AnswerChoices: Yes; No
  CorrectAnswer: Yes
  AnswerTests: omnitest(correctVal='Yes')
  Hint: The dice were loaded in opposite ways so their average should be fair. No?

- Class: text
  Output: For a continuous random variable X, the expected value is defined analogously
  as it was for the discrete case. Instead of summing over discrete values, however,
  the expectation integrates over a continuous function.

- Class: text
  Output: It follows that for continuous random variables, E(X) is the area under the
  function t*f(t), where f(t) is the PDF (probability density function) of X. This
  definition borrows from the definition of center of mass of a continuous body.

- Class: figure
  Output: Here's a figure from the slides. It shows the constant (1) PDF on the left
  and the graph of t*f(t) on the right.
  Figure: plot1.R
  FigureType: new

- Class: mult_question
  Output: Knowing that the expected value is the area under the triangle, t*f(t), what
  is the expected value of the random variable with this PDF?
  AnswerChoices: 1.0; 2.0; .5; .25
  CorrectAnswer: .5
  AnswerTests: omnitest(correctVal='.5')
  Hint: The area of the triangle is base*height/2.

- Class: figure
  Output: For the purposes of illustration, here's another figure using a PDF from our
  previous probability lesson. It shows the triangular PDF f(t) on the left and the
  parabolic t*f(t) on the right. The area under the parabola between 0 and 2 represents
  the expected value of the random variable with this PDF.
  Figure: plot2.R
  FigureType: new

- Class: cmd_question
  Output: To find the expected value of this random variable you need to integrate the
  function t*f(t). Here f(t)=t/2, the diagonal line. (You might recall this from the
  last probability lesson.) The function you're integrating over is therefore t^2/2.
  We've defined a function myfunc for you representing this. You can use the R function
  'integrate' with parameters myfunc, 0 (the lower bound), and 2 (the upper bound) to
  find the expected value. Do this now.
  CorrectAnswer: integrate(myfunc,0,2)
  AnswerTests: omnitest(correctExpr='integrate(myfunc,0,2)')
  Hint: Type 'integrate(myfunc,0,2)' at the command prompt.


- Class: text
  Output: As all the examples have shown, expected values of distributions are useful
  in characterizing them. The mean characterizes the central tendency of the
  distribution. However, often populations are too big to measure, so we have to sample
```

```
  them and then we have to use sample means. That's okay because sample expected values
  estimate the population versions. We'll show this first with a very simple toy and
  then with some simple equations.

- Class: cmd_question
  Output: We've defined a small population of 5 numbers for you, spop. Look at it now.
  CorrectAnswer: spop
  AnswerTests: omnitest(correctExpr='spop')
  Hint: Type 'spop' at the command prompt.

- Class: cmd_question
  Output: The R function mean will give us the mean of spop. Do this now.
  CorrectAnswer: mean(spop)
  AnswerTests: omnitest(correctExpr='mean(spop)')
  Hint: Type 'mean(spop)' at the command prompt.

- Class: cmd_question
  Output: Suppose spop were much bigger and we couldn't measure its mean directly and
  instead had to sample it with samples of size 2. There are 10 such samples, right?
  We've stored this for you in a 10 x 2 matrix, allsam. Look at it now.
  CorrectAnswer: allsam
  AnswerTests: omnitest(correctExpr='allsam')
  Hint: Type 'allsam' at the command prompt.

- Class: cmd_question
  Output: Each of these 10 samples will have a mean, right? We can use the R function
  apply to calculate the mean of each row of the matrix allsam. We simply call apply
  with the arguments allsam, 1, and mean. The second argument, 1, tells 'apply' to
  apply the third argument 'mean' to the rows of the matrix. Try this now.
  CorrectAnswer: apply(allsam,1,mean)
  AnswerTests: omnitest(correctExpr='apply(allsam,1,mean)')
  Hint: Type 'apply(allsam,1,mean)' at the command prompt.

- Class: text
  Output: You can see from the resulting vector that the sample means vary a lot, from
  2.5 to 11.5, right? Not unexpectedly, the sample mean depends on the sample. However...

- Class: cmd_question
  Output: ... if we take the expected value of these sample means we'll see something
  amazing. We've stored the sample means in the array smeans for you. Use the R
  function mean on the array smeans now.
  CorrectAnswer: mean(smeans)
  AnswerTests: omnitest(correctExpr='mean(smeans)')
  Hint: Type 'mean(smeans)' at the command prompt.

- Class: text
  Output: Look familiar? The result is the same as the mean of the original population
  spop. This is not because the example was specially cooked. It would work on any
  population. The expected value or mean of the sample mean is the population mean.
  What this means is that the sample mean is an unbiased estimator of the population
  mean.

- Class: text
  Output: Formally, an estimator e of some parameter v is unbiased if its expected
  value equals v, i.e., E(e)=v. We can show that the expected value of a sample mean
  equals the population mean with some simple algebra.

- Class: text
  Output: Let X_1, X_2, ... X_n be a collection of n samples from a population with
  mean mu. The  mean of these is (X_1 + X_2 + ... + X_n)/n.

- Class: text
  Output: What's the expected value of the mean? Recall that E(aX)=aE(X), so E(
  (X_1+..+X_n)/n ) =

- Class: text
  Output: 1/n * (E(X_1) + E(X_2) + ... + E(X_n)) = (1/n)*n*mu = mu. Each E(X_i) equals
  mu since X_i is drawn from the population with mean mu. We expect, on average, a
  random X_i will equal mu.
```

```
- Class: text
  Output: Now that was theory.  We can also show this empirically with more simulations.


- Class: figure
  Output: Here's another figure from the slides. It shows how a sample mean and the mean of averages spike together. The two shaded distributions come from the same data. The blue portion represents the density function of randomly generated standard normal data, 100000 samples. The pink portion represents the density function of 10000 averages, each of 10 random normals. (The original data was stored in a 10000 x 10 array and the average of each row was taken to  generate the pink data.)
  Figure: normalMeans.R
  FigureType: new

- Class: figure
  Output: Here's another figure from the slides. Rolling a single die 10000 times yields the first figure. Each of the 6 possible outcomes appears with about the same frequency. The second figure is the histogram of outcomes of the average of rolling two dice. Similarly, the third figure is the histogram of averages of rolling three dice, and the fourth four dice. As we showed previously, the center or mean of the original distribution is 3.5 and that's exactly where all the panels are centered.
  Figure: diceRolls.R
  FigureType: new

- Class: text
  Output: Let's recap. Expected values are properties of distributions. The average, or mean, of random variables is itself a random variable and its associated distribution itself has an expected value. The center of this distribution is the same as that of the original distribution.

- Class: text
  Output: Now let's review!

- Class: mult_question
  Output: Expected values are properties of what?
  AnswerChoices: demanding parents; distributions; fulcrums; variances
  CorrectAnswer: distributions
  AnswerTests: omnitest(correctVal='distributions')
  Hint: What would you expect to have a center?

- Class: mult_question
  Output: A population mean is a center of mass of what?
  AnswerChoices: a family; a distribution; a population; a sample
  CorrectAnswer: a population
  AnswerTests: omnitest(correctVal='a population')
  Hint: What word appears in the question?

- Class: mult_question
  Output: A sample mean is a center of mass of what?
  AnswerChoices: a family; a distribution; a population; observed data
  CorrectAnswer: observed data
  AnswerTests: omnitest(correctVal='observed data')
  Hint: If you're sampling you need to observe data, right?

- Class: mult_question
  Output: True or False? A population mean estimates a sample mean.
  AnswerChoices: True; False
  CorrectAnswer: False
  AnswerTests: omnitest(correctVal='False')
  Hint: We can only sample a population and calculate the sample mean.

- Class: mult_question
  Output: True or False? A sample mean is unbiased.
  AnswerChoices: True; False
  CorrectAnswer: True
  AnswerTests: omnitest(correctVal='True')
  Hint: The sample mean is the population mean, so by definition it's unbiased.
```

```yaml
- Class: mult_question
  Output: True or False? The more data that goes into the sample mean, the more concentrated its density / mass function is around the population mean.
  AnswerChoices: True; False
  CorrectAnswer: True
  AnswerTests: omnitest(correctVal='True')
  Hint: It's better to have more data than less, right?

- Class: text
  Output: Congrats! You've concluded this lesson on expectations. We hope it met yours.

- Class: mult_question
  Output: "Would you like to receive credit for completing this course on
    Coursera.org?"
  CorrectAnswer: NULL
  AnswerChoices: Yes;No
  AnswerTests: coursera_on_demand()
  Hint: ""
```