140.615.HW.2.Delahanty.Jeremy

Jeremy Delahanty

2024-02-21

1 Suppose U has a Uniform(5,10) distribution.

Without using R, calculate the following:

1a

$E(U)$

A uniform distribution, sometimes called a *box distribution* because of how the probability density function appears, assigns equal probabilities to intervals of equal lengths. The expected value of a continuous random variable is the continuous analog of the expected value of a discrete random variable, where instead of summing of all possible values we integrate them. With our interval of $[5, 10]$, we can calculate the expected value through

$$E[X] = \int_a^b x \cdot \frac{1}{b-a} dx = \frac{b^2 - a^2}{2} \cdot \frac{1}{b-a} = \frac{(b-a)(b+a)}{2} \cdot \frac{1}{b-a} = \frac{b+a}{2}$$

This is simply:

$$\begin{aligned} E[X] &= \frac{10+5}{2} \\ &= \frac{15}{2} \\ &= 7.5 \end{aligned}$$

In other words, the expected value of this distribution is given by the average of the parameters or the midpoint of the interval. The expected value can be thought of as the "center of mass" and, since the PDF is constant over the interval $[a, b]$, the center of mass is simply the endpoint.

1b

$Pr(U = 6)$

In a continuous uniform distribution where all events being equally likely to occur, there is an infinite count of real number values within the range. In this case, it is the range $[5, 10]$. Further, the sum of these probabilities must be equal to one. Therefore, the probabilities must be infinitesimally small.

Further, this is effectively taking the integral not over a range but a single point or line segment which has no area!

So: $Pr(U = 6) = 0$

1c

$Pr(U > 6)$

This is effectively asking what the sum of probabilities for values greater than 6 up to 10. In other words, we want the area under the curve in the range $[a > 6, b = 10]$ with respect to the entire interval size.

This can be evaluated as:

$$\begin{aligned} Pr(U > 6) &= \frac{b - x}{10 - 5} \\ &= \frac{10 - 6}{10 - 5} \\ &= \frac{4}{5} \\ &= 0.80 \end{aligned}$$

We can also try and describe the probability density function (PDF) by understanding how this is integrated to form the cumulative density function.

The function for the PDF can be described by:

$f(U) = \frac{1}{b-a} \; for \; a \leq x \leq b$

The problem can be described as:

$$\begin{aligned} Pr(U > 6) \;&=\; 1 \;-\; \int_5^6 \cdot f(u)du \; where \; f(u) \;=\; \frac{1}{5} \\ &=\; 1 \;-\; \int_5^6 \cdot \frac{1}{5}du \\ &=\; 1 \;-\; (\frac{6}{5} - \frac{5}{5}) \\ &=\; 1 \;-\; \frac{1}{5} \\ &=\; \frac{4}{5} \\ &=\; 0.80 \end{aligned}$$

1d

$Pr(7 < U < 9)$

Similar to 1c, we are trying to determine the area under the curve for the range specified with respect to the entire interval.

This can be evaluated as:

$$\begin{aligned} Pr(7 < U < 9) &= \frac{upper\ bound - \ lower\ bound}{10 - 5} \\ &= \frac{9 - 7}{10 - 5} \\ &= \frac{2}{5} \\ &= 0.40 \end{aligned}$$

I am not brave enough to try and compute this with the integral for now.

2

Suppose $X \sim N(\mu = 4,\ \sigma = 2)$. Calculate the following:

2a

$Pr(X = 4)$

Similar to question 1b, the probability of an exact value in a continuous distribution with infinitely many values in the defined range is 0. There is no area under a curve at a point.

Thus: $Pr(X = 4) = 0$

2b

$Pr(X < 6)$

We can use `R`'s `pnorm` function to calculate these values. The `pnorm` function returns the value of the cumulative density function in the normal distribution when supplied with a random variable, mean, and standard deviation.

```r
# Use pnorm with q=6, mean=4, SD=2
# pnorm returns value to the left of the variable by default, so lower.tail=TRUE
# is not necessary

probability <- pnorm(q=6, mean=4, sd=2)

# Display the value

cat("Probability that x is less than 6:", probability)
```

```
## Probability that x is less than 6: 0.8413447
```

2c

$Pr(X > 0)$

We can again use the `pnorm` function in a similar way. However, this time we must supply the `lower.tail=FALSE` argument so the function returns the area to the right of the curve.

```r
# Use pnorm with q=0, mean=4, SD=2
# pnorm returns value to the left of the variable by default, so lower.tail=FALSE
# is not necessary

probability <- pnorm(q = 0, mean = 4, sd = 2, lower.tail = FALSE)

# Display the value

cat("Probability that x is greater than 0:", probability)
```

```
## Probability that x is greater than 0: 0.9772499
```

2d

$Pr(1 < X < 4)$

In this problem, we again must use pnorm for two different values and subtract the smaller bound's probability from the upper bound's probability to obtain only the area under the curve between them. In `R`, this is accomplished by:

```r
# Use pnorm for both values of x=1 and x=4, mean=4, SD=2
# subtract smaller bound cdf from larger bound cdf

probability <- pnorm(q = 4, mean = 4, sd = 2) - pnorm(q = 1, mean = 4, sd = 2)

# Display the value

cat("Probability 1<x<4:", probability)
```

```
## Probability 1<x<4: 0.4331928
```

2e

$Pr(|X - 4| > 1)$

This problem is slightly more complicated. We must first solve for the values that make the inequality $|x - 4| > 1$ true.

This is done through breaking the inequality into two "cases". Generally, these are described by:

$|\ f(x)\ | > a \ = \ f(x) > a \ or \ f(x) < -a$

In our case:

$|\ X - 4\ |> a \ = \ x - 4 > 1 \ or \ x - 4 < -1$

Solving for each of these scenarios gives us:

$x > 5 \ or \ x < 3$

Now, we must solve for the probability that X is *not* in the range of $[3, 5]$. We need to take the right tail of $Pr(x > 5)$ and the left tail of $Pr(x < 3)$ provided by the `pnorm` function. Finally, we will add them together to find the probability for this problem.

```r
# Use pnorm for both values of x=4 and x=5, mean=4, SD=2
# Add left tail from pnorm of 3 (all values less than 3) to
# right tail of pnorm of 5 (all values greater than 5)

probability <- pnorm(q = 3, mean = 4, sd = 2) + pnorm(q = 5, mean = 4, sd = 2, lower.tail = FALSE)

# Display the value

cat("Probability of |X-4|>1:", probability)
```

```
## Probability of |X-4|>1: 0.6170751
```

3

These are problems from section 6 of John Verzani's simpleR notes. Make your results reproducible using the `R` command `set.seed(10)` at the beginning of your code.

3a

Generate 10 random numbers from a Uniform distribution on [0,10]. Use R to find the maximum and minimum values.

We can do the following within `R`:

```r
# Set consistent seed
set.seed(10)

# Generate random numbers from uniform distribution with runif
# n = 10, minimum = 0, maximum = 10

uniform_distribution <- runif(n = 10, min = 0, max = 10)

# Get minimum value

minimum_value <- min(uniform_distribution)

# Get maximum value

maximum_value <- max(uniform_distribution)

cat("The minimum value generated is:", minimum_value, "\n")
```

```
## The minimum value generated is: 0.8513597
```

```r
cat("The maximum value generated is:", maximum_value)
```

```
## The maximum value generated is: 6.931021
```

3b

Generate 10 random normal numbers with mean 5 and standard deviation 5. Use R to find out how many of these are less than 0.

In `R`, we can use the `rnorm` function:

```r
# Set consistent seed
set.seed(10)

# Generate random numbers from normal distribution with rnorm
# n = 10, mean = 5, sd = 5

random_normal_numbers <- rnorm(n = 10, mean = 5, sd = 5)
```

```
# Create logical vector to count the number of entries less than zero
# This creates a boolean array of the indices of which values selected are
# less than zero. Booleans are evaluated to numerics as 1 for true, 0 for false
# so total of our sum is how many numbers are indeed less than 0

tot_less_than_zero <- sum(random_normal_numbers<0)

# Display the value

cat("Number of values less than zero:", tot_less_than_zero)
```

```
## Number of values less than zero: 3
```

3c

Generate 100 random normal numbers with mean 100 and standard deviation 10. How many are within 2 standard deviations from the mean?

We can use `R` to first generate 100 numbers and then define upper and lower bounds that are 2 SD away from the defined mean. Then, we can generate logical arrays for each condition and sum them together.

```
# Set consistent seed
set.seed(10)

# Generate random numbers from normal distribution with rnorm
# n = 100, mean = 100, sd = 5
random_normal_numbers <- rnorm(n = 100, mean = 100, sd = 5)

# Define upper and lower bounds using the provided SD of 10
upper_bound <- 100 + 2*10
lower_bound <- 100 - 2*10

# Create logical array that satisfies numbers which return TRUE for both
# being larger than the lower bound and smaller than the upper bound (within 2 sd)
# and finally sum them together
tot_within_2_sd <- sum(random_normal_numbers > lower_bound & random_normal_numbers < upper_bound)

#Display the values
cat("Total numbers within 2 standard deviations:", tot_within_2_sd)
```

```
## Total numbers within 2 standard deviations: 100
```

3d

Toss a fair coin 50 times using the R function sample(). How many heads do you have?

We can use the `R` function `sample()` for this question.

```
# Set consistent seed
set.seed(10)

# Use sample() with:
```

```r
# x: vector of outcomes ("H", "T") for Heads/Tails, respectively
# size = 50 for number of trials
# replace must be TRUE because, presumably, we are using the same coin each time for our sample
coin_tosses <- sample(c("H", "T"), size = 50, replace = TRUE)

# Create logical vector for the number of heads achieved where the string is
# either "H" or "T" and then sum that boolean array to see total number of heads
tot_heads <- sum(coin_tosses == "H")

# Display the value
cat("Total number of heads flips:", tot_heads)
```

```
## Total number of heads flips: 25
```

3e

Roll a fair six-sided die 100 times. How many 6's did you see?

We can use the sample function once again.

```r
# Set consistent seed
set.seed(10)

# Use sample with:
# x: vector of outcomes 1 through 6 (1:6)
# size = 100
# replace = TRUE because we are rolling the same dice and want the possibility of
# the same values being available
dice_rolls <- sample(c(1:6), size = 100, replace = TRUE)

# Create logical vector for number of heads achieved where the integer is a 6
# and sum up the booleans
tot_six <- sum(dice_rolls == 6)

# Display the value
cat("Total number of 6's rolled:", tot_six)
```

```
## Total number of 6's rolled: 16
```

3f

For $X \sim Normal(0, 1)$, find the number x so that $Pr(X \leq x) \ = \ 0.05$.

In this case, we want to know which value will gives us a probability of 0.05. In other words, we want to know what quantile gives us this exact area under the curve.

We can use the `qnorm` function for this in `R`.

```r
# Set consistent seed
set.seed(10)

# Define target probability, p
p <- 0.05
```

```
# Use qnorm with p = 0.05, mean = 0, sd = 1
value <- qnorm(p = p, mean = 0, sd = 1)

# Display the value
cat("The value that yields 0.05 is:", value)
```

```
## The value that yields 0.05 is: -1.644854
```

3g

For $X \sim Normal(0,1)$, find the number x that solves $Pr(-x \leq X \leq x) \ = \ 0.10$

In this case, we want to know the range of values that will give us a specific probability. We can again use the quantile function `qnorm` for this and subtract the upper bound from the lower bound probability to get the region between them.

```
# Set consistent seed
set.seed(10)

# Each side of the distribution between the mean has a total probability of 0.5
# Since we are going around the mean and have 0.10 across it, this means that
# each remaining half of the distribution has a probability of 0.45 occurring.
# We need to find those values of the upper and lower bounds first

# Determine lower bound. This cumulative probability will be outside the interval
# to the LEFT of this value
lower_bound <- qnorm(p = 0.45, mean = 0, sd = 1) # lower.tail is true by default

# Determine upper bound. This cumulative probability will be outside the interval
# to the RIGHT of this value, so lower.tail must be true
upper_bound <- qnorm(p = 0.45, mean = 0, sd = 1, lower.tail = FALSE)

# Display the values
cat("The value of the lower bound is:", lower_bound,"\n",
    "The value of upper bound is:", upper_bound, "\n",
    "The value of x is:", upper_bound)
```

```
## The value of the lower bound is: -0.1256613
##  The value of upper bound is: 0.1256613
##  The value of x is: 0.1256613
```

If desired, this can be checked by taking the `pnorm` values for each of these bounds and subtracting the lower bound from the upper bound probability.

3h

How much area (probability) is to the right of 1.5 for a Normal(0,1)

This is a similar problem to earlier questions, but using the normal distribution.

We can uses `pnorm` again but use the `lower.tail=FALSE` parameter.

```r
# Set consistent seed
set.seed(10)

# Use pnorm with q = 1.5, mean = 0, sd = 1, lower.tail=FALSE because
# we want area to the right
probability <- pnorm(q = 1.5, mean = 0, sd = 1, lower.tail = FALSE)

# Display the value
cat("Probability to the right of x=1.5:", probability)
```

```
## Probability to the right of x=1.5: 0.0668072
```

Question 4

A biologist made a certain pH measurement in each of 24 frogs. Typical values were 7.43, 7.16, 7.51, etc. She calculated a mean of 7.373 and a standard deviation of 0.129 for these original pH measurements. Next, she transformed the data by subtracting 7 from each observation and then multiplying by 100. The transformed data are 43, 16, 51, etc. What are the mean and standard deviation of these transformed data?

This question is asking what transformations upon the data do to the mean and standard deviation. We need to figure out the new expected value of the dataset $E(Y)$ as well as a new SD σ. It can be represented as:

$Y \ = \ (X - 7) \times 100$

Where Y is the new dataset and X is the original dataset being transformed. This can be simplified to the following for the mean of the data:

$$\begin{aligned} E(Y) \ &= \ E((x-7) \times 100) \\ &= E(100x - 700) \\ &= 100E(x) - 700 \end{aligned}$$

This can be simply computed in `R`:

```r
# Define original mean
mean_original <- 7.373

# Mean of transformed dataset, E(Y)
mean_transformed <- 100*mean_original - 700

# Display the value
cat("Transformed mean:", mean_transformed)
```

```
## Transformed mean: 37.3
```

For standard deviation, we can demonstrate the effects of our transformations as follows. First, translations of data that are shifted by a constant value do not influence the standard deviation despite changing the mean.

The mean is changed in the following way through adding individual values:

$$Y = X + c \qquad \begin{aligned} E(Y) &= \frac{\sum_{i=1}^{n}(x_i + c)}{n} \\ &= \frac{\sum_{i=1}^{n} x_i + nc}{n} \\ &= \frac{\sum_{i=1}^{n} x_i}{n} + \frac{nc}{n} \\ &= \mu + c \end{aligned}$$

Thus, adding a constant to every value and no other transformation adds only that constant to the mean. We can use this for our next description of the standard deviation.

$$\sigma_Y = \sqrt{\frac{\sum_{i=1}^{n}(y_i - E(Y))^2}{n}}$$

We saw previously that these transformed values for y and E(Y) simply become the following:

$$\begin{aligned} \sigma_Y &= \sqrt{\frac{\sum_{i=1}^{n}(x_i + c - (\mu + c))^2}{n}} \\ &= \sqrt{\frac{\sum_{i=1}^{n}(x_i - \mu)^2}{n}} \end{aligned}$$

This is just the formula for standard deviation of x! So with just this constant added to the data, we do not change the SD. $\sigma_Y = \sigma_X$.

Second, multiplications upon data act as a scaling factor for the data. It changes both the mean and the standard deviation. The mean can be described as follows:

$E(Y) = c \cdot E(X)$ where c is a constant.

When we use this in our standard deviation formula for Y, we see:

$$\begin{aligned} \sigma_Y &= \sqrt{\frac{\sum_{i=1}^{n}(y_i - (E(Y))^2}{n}} \\ &= \sqrt{\frac{\sum_{i=1}^{n}(x_i c - (c \ \cdot \mu_x)^2}{n}} \\ &= \sqrt{\frac{\sum_{i=1}^{n} c^2 (x_i - \mu_x)^2}{n}} \\ &= \sqrt{c^2} \ \cdot \ \sqrt{\frac{\sum_{i=1}^{n}(x_i - \mu_x)^2}{n}} \\ &= \mid c \mid \ \cdot \ \sqrt{\frac{\sum_{i=1}^{n}(x_i - \mu_x)^2}{n}} \\ &= \mid c \mid \ \cdot \sigma_X \end{aligned}$$

This can be computed very simply in R!

```r
# From description above, we simply take the original standard deviation
# and multiply by the absolute value of our constant

# Define original standard deviation
original_sd <-0.129
```

```r
# Define the constant within the absolute value because we can only use positive
# numbers in the SD
c = abs(100)

# Calculate the result
transformed_sd <- original_sd * c

# Display the value
cat("Transformed standard deviation:", transformed_sd)
```

```
## Transformed standard deviation: 12.9
```

5

Suppose we have 100 independent draws from some population distribution whose shape is unknown but where the population mean is 10 and SD is 2.5. Suppose that n=100 is sufficiently large that for the sample mean to have an approximately normal distribution.

5a

What is the chance that the sample mean is within 0.1 units of the population mean?

In this case, our n is sufficiently high to assume our data fits an approximately normal distribution. Thus, our sample mean can be thought of sitting near the true population mean. Due to this assumption, we will have to alter our standard deviation to use the square root of n. We will use `pnorm` for solving this calculation.

```r
# Define population mean
mu = 10

# Define population standard deviation
sigma_population = 2.5

# Number of events
n = 100

# Calculate appropriate sample standard deviation
sigma_sample <- sigma_population/sqrt(n)

# Calculate upper and lower bounds
upper_bound <- mu + 0.1
lower_bound <- mu - 0.1

# Calculate the probability that the data is within the bounds set above
# similarly to how we did it in problem 2d
probability <- pnorm(q = upper_bound, mean = mu, sd = sigma_sample) -
  pnorm(q = lower_bound, mean = mu, sd = sigma_sample)

# Display the value
cat("Probability sample mean is within 0.1 units of mu:", probability)
```

```
## Probability sample mean is within 0.1 units of mu: 0.3108435
```

5b

What is the chance that the sample mean exceeds the population mean by at least 0.25 units?

This is a simple `pnorm` problem using the `lower.tail=FALSE` parameter.

```r
# Calculate the threshold we're determining probability of going past
threshold <- mu + 0.25

# Use pnorm function with lower.tail=FALSE and values computed from cell above
# for calculations
past_threshold_probability <- pnorm(q = threshold, mean = mu, sd = sigma_sample,
                                    lower.tail = FALSE)

# Display the values
cat("Probability sample mean exceeds by at least 0.25 units:",
    past_threshold_probability)
```

```
## Probability sample mean exceeds by at least 0.25 units: 0.1586553
```