Z-Distribution and Probability

Introduction

This week, we have just have one exercise. Its objective is for you gain comfort with the concept of distribution and probability by working with the Z-distribution. We will learn how to use the plot and probability function we used throughout the lesson.

Case Study: Barley Data

We were introduced to the barley dataset last week. These data are from a trial that was conducted to measure the uniformity of plots, that is, how consistent were yield measurements from plots that were treated identically.

```
barley = read.csv("data/barley_uniformity.csv")
```

Let's take a look at the top six rows of the dataset using the `head()` function. We can see the plots are identified by their row nad column locations. The yields are not per acre, but per plot.

```
head(barley)
```

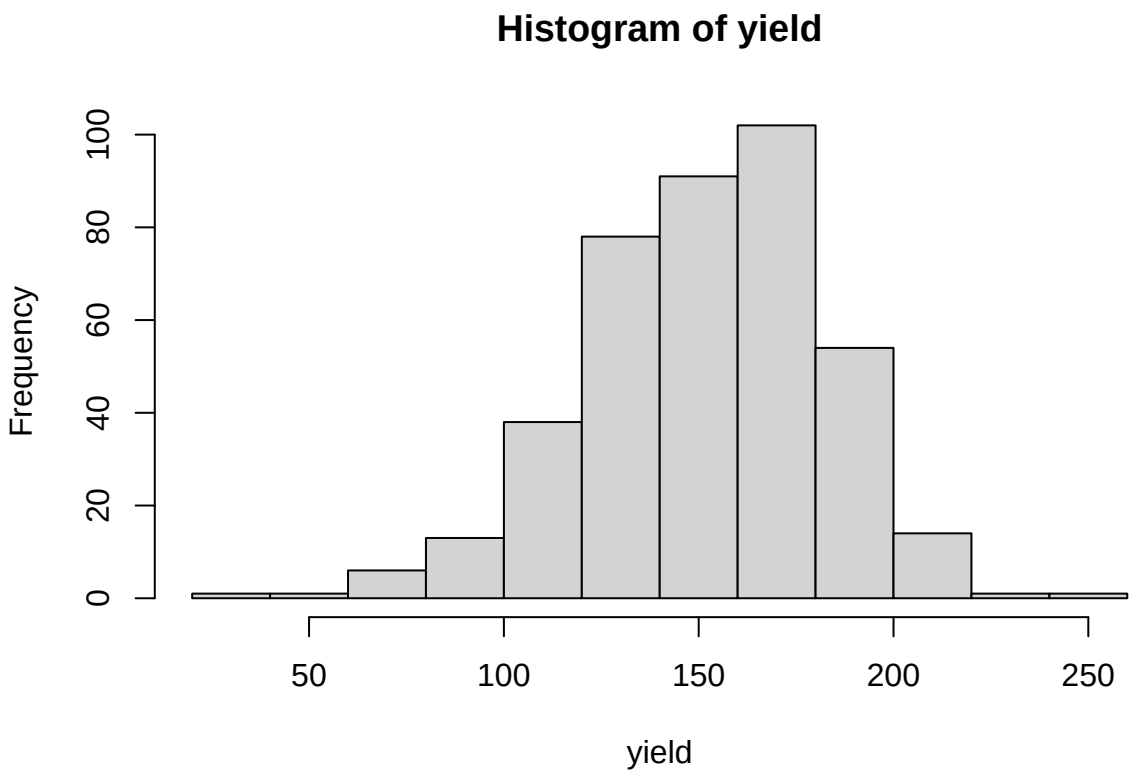
```
##   row col yield
## 1  20   1   185
## 2  19   1   169
## 3  18   1   216
## 4  17   1   165
## 5  16   1   133
## 6  15   1   195
```

Measuring the Population Distribution: Mean and Standard Deviation

Before we can calculate probability, we need to construct the normal distribution curve. And before we can do that, we need to measure the mean and standard deviation of the barley population.

Let's do a quick visual check to make sure the data are approximately normally-distributed. We will create a new object, `yield`, from the yield column of the original data.frame and then plot it using the `histogram()` function. Remember, we reference a column from the barley dataframe by typing "barley", the dollar sign "$", and then the name of the column of interest.

```
yield = barley$yield
hist(yield)
```



The histogram shows the barley distribution is approximately normal. Remember, rarely is a distribution perfectly normal. When we fit the data with the normal distribution, we are using a model distribution, not the actual distribution. But the model tends to perform very well, in predicting the frequency of values in the population it represents.

We can quickly calculate the mean and standard deviation using the `mean()` and `sd()` functions on `yield`.

```
mean(yield)
```

```
## [1] 151.92
```

```
sd(yield)
```

```
## [1] 31.14029
```

In the next chunk, we will assign these statistics to variables we can reference later on. This way, we don't have to enter the mean and standard deviation repeatedly in our analyses. Remember, the population mean is symbolized by the Greek letter μ, which we pronounce as "mu". The population standard deviation is symbolized by the Greek letter σ, which we pronounce as "sigma".

```
mu = mean(yield)
sigma = sd(yield)
```

The shadeDist Function

To calculate probabilities, we will use the `shadDist()` function which is part of the `fastGraph` package in R. Remember, packages are groups of functions (calcluations and plots) beyond those in the base R software. The first time we use `fastGraph`, we must install it. It is already installed in this week's unit. If you needed to install it, however, you would just remove the hashtag at the beginning of the line below. This would "uncomment" that code and allow it to run.

```
# install.packages("fastGraph") #delete pound sign at far left to use this command
```

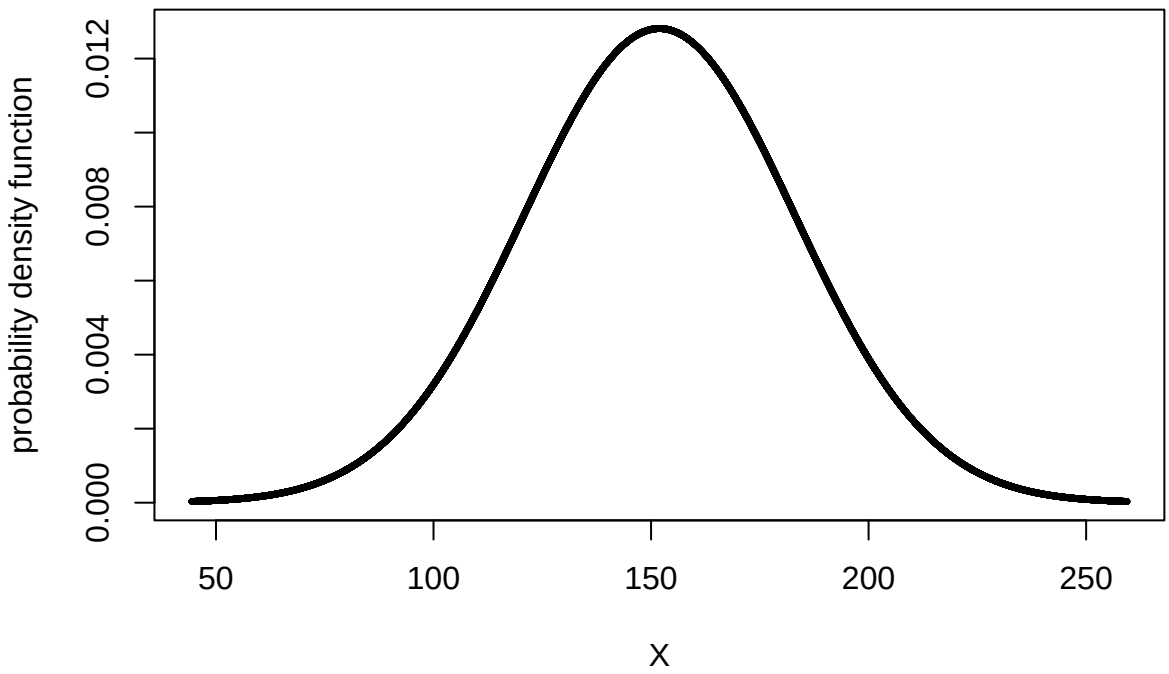
We can then load the package using the `library(fastGraph)` command.

```
library(fastGraph)
```

Now we are ready to look at the barley distribution. We will define the column we are interested in as "an independent"yield" and then plot it's distribution model using shadeDist. To plot the basic distribution, we have to enter four arguments into shadeDist:

- `xshade = NULL`: this tells R not to shade any part of the curve
- `ddist = "dnorm"`: this tells R we are working with the normal (Z) distribution
- `parm1 = mu`: the `parm1` argument is used to pass the appropriate parameter to R for the distribution we selected with the `ddist` argument. For a normal distribution, the first parameter is the mean.
- `parm2 = sigma`: the `parm2` argument passes the next appropriate parameter to R for the normal distribution, the standard deviation.

```
yield = barley$yield
shadeDist(xshade = NULL,
          ddist = "dnorm",
          parm1 = mu,
          parm2 = sigma)
```



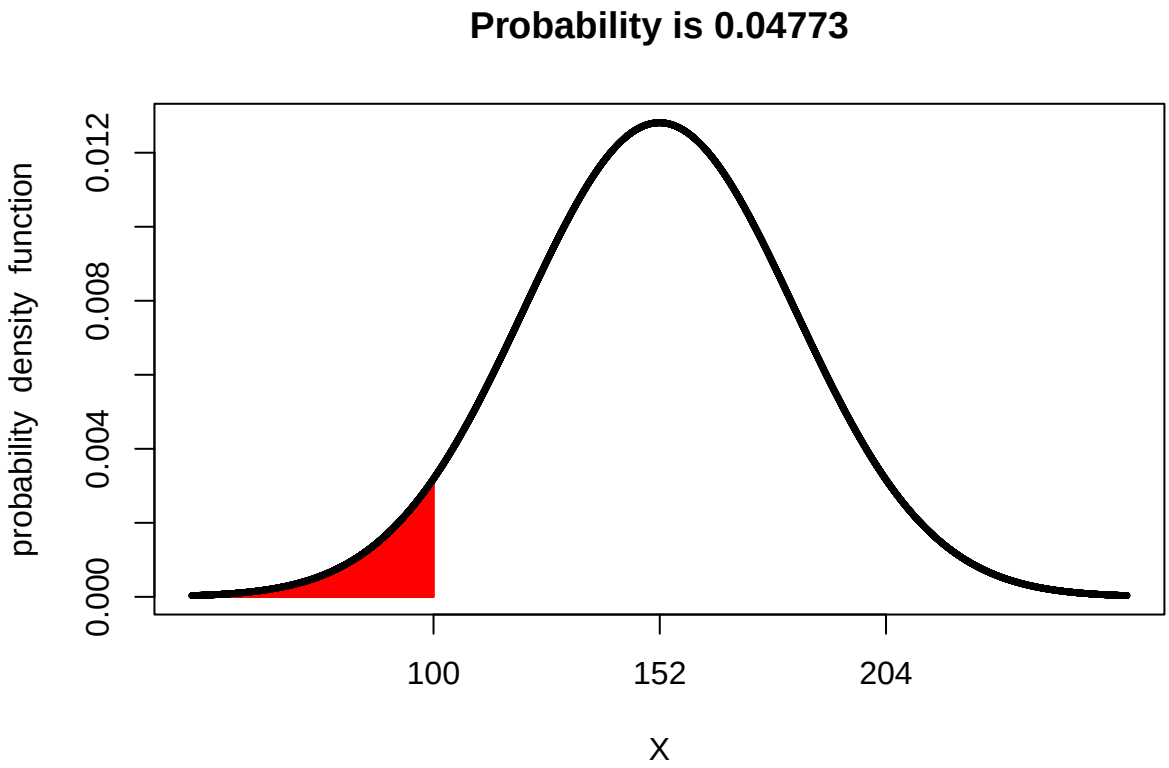
And *voila!*, we have our basic normal distribution curve. The horizontal (X) axis represents the range of observed values. The vertical axis, titled (probability density function), indicates the proportion of the population that would include a given range of X values.

Probabilities in Lower Tails

Lets run our first calculation. What percentage of this population is expected to have a yield value less than 100? to answer this, we will pass the following two arguments to shadeDist:

- `xshade = 100`: This tells R we want to bisect the curve at X = 100
- `lower.tail = TRUE`: This tells R we want to measure the area of the curve where X<100.

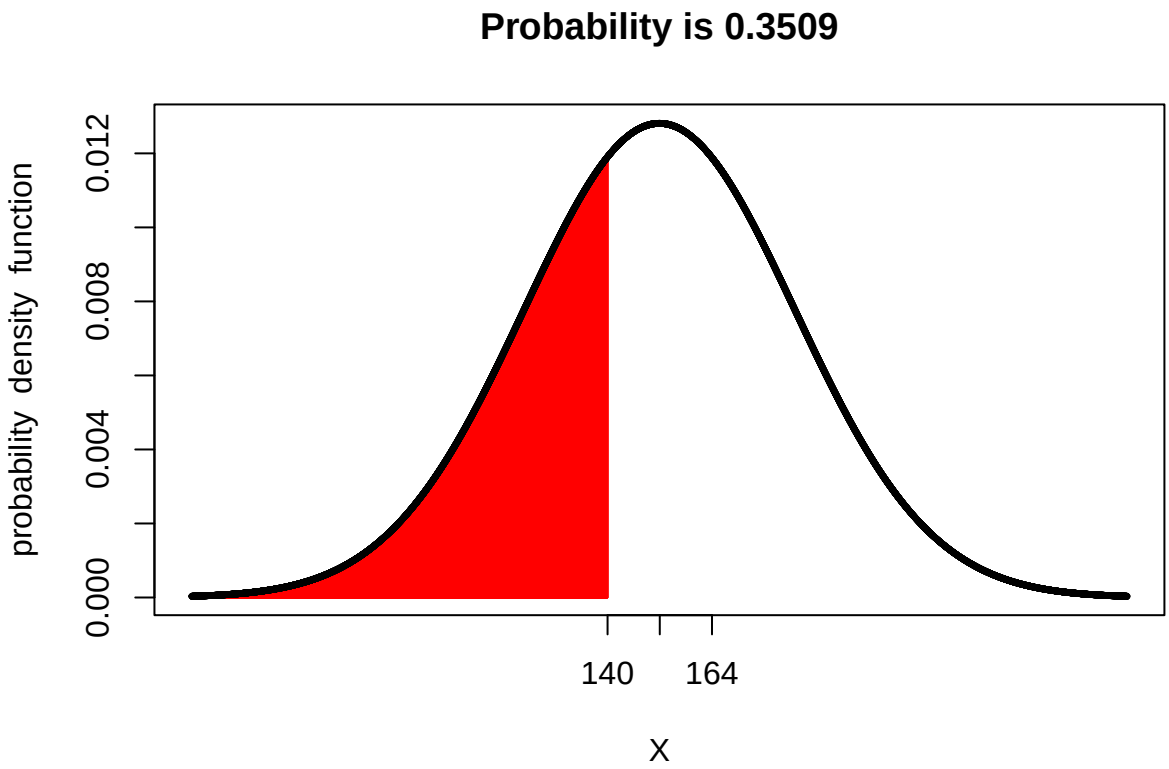
```
shadeDist(xshade = 100,
          ddist = "dnorm",
          parm1 = mu,
          parm2 = sigma,
          lower.tail = TRUE)
```



When we run the curve, we see that approximately 0.048 – or 4.8 percent – of the population is expected to have a yield value less than 100. If we were randomly subsets of this field, we would expect to measure a yield less than 100 in about 4.8 our of 100 samples. In other words, there is about a 4.8% probability our subset would have a value less than 100.

What is the probability our subset would have a value less than 140?

```
shadeDist(xshade = 140,
          ddist = "dnorm",
          parm1 = mu,
          parm2 = sigma,
          lower.tail = TRUE)
```

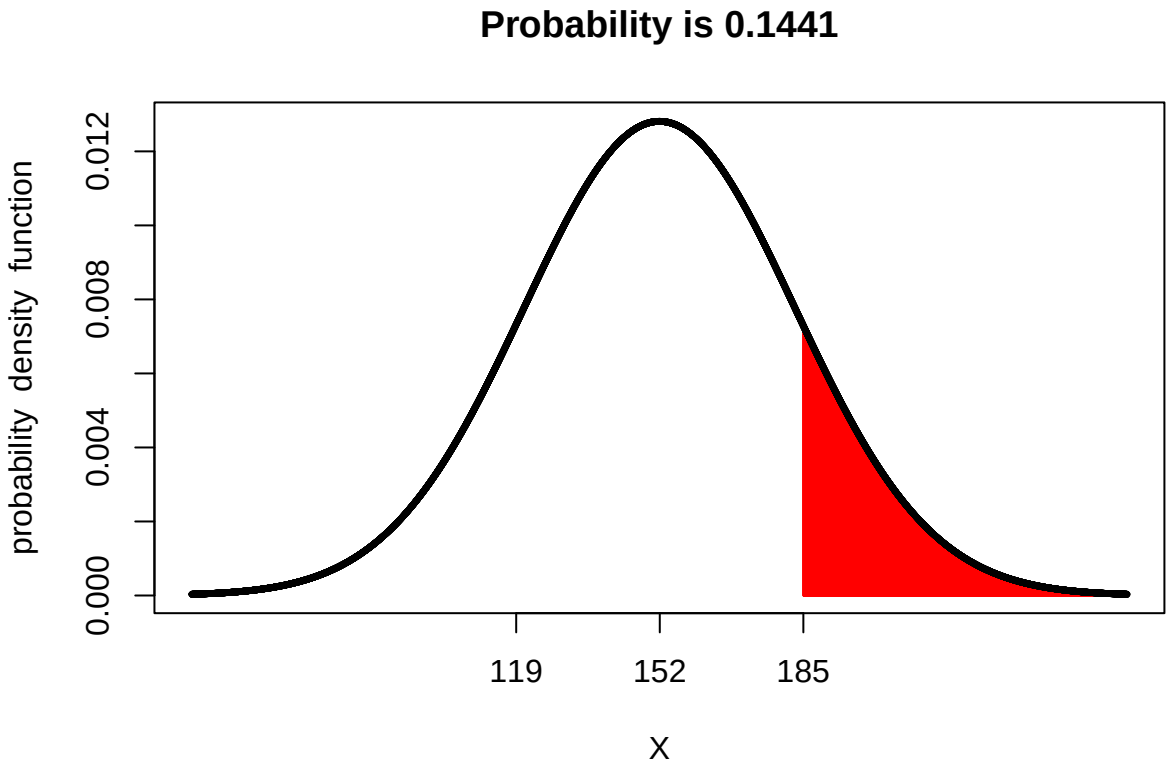


Our probability is about 0.35, or 35%.

Probabilities in Upper Tails

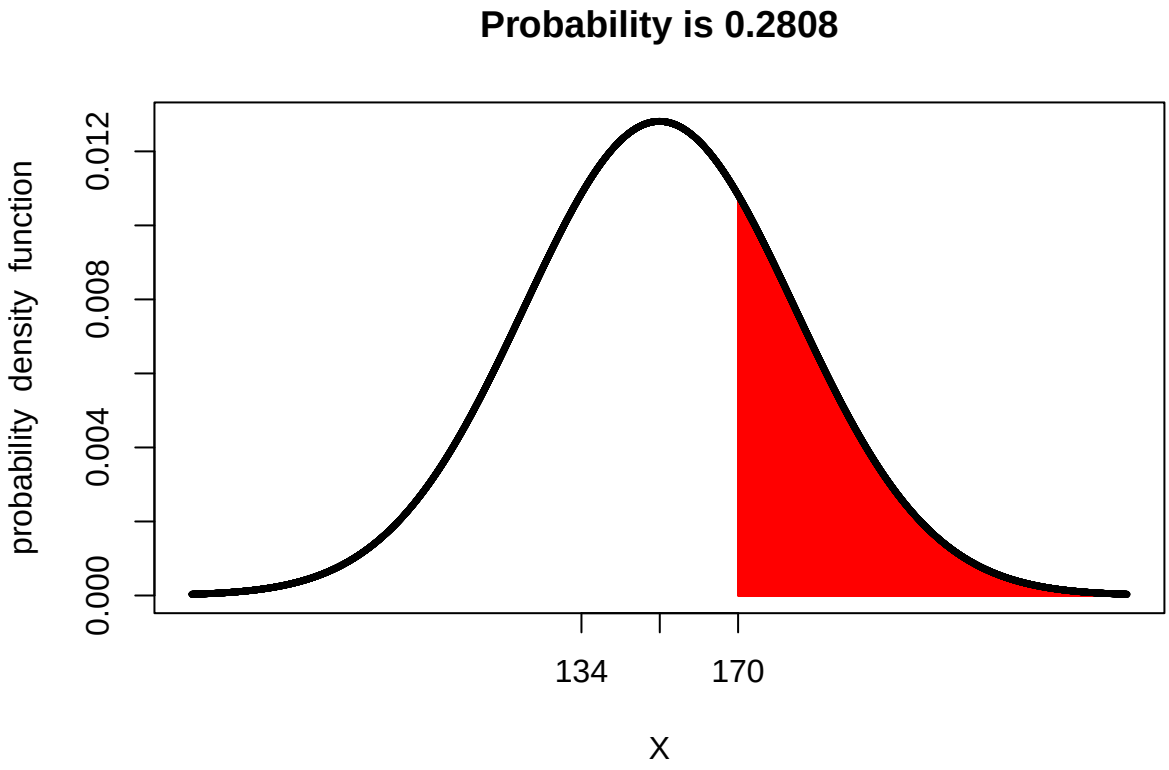
If we want to measure the probability of higher values, we just need to change the lower.tail argument to `lower.tail = FALSE`. The probability of values greater than 185 would be about 14%.

```
shadeDist(xshade = 185,
          ddist = "dnorm",
          parm1 = mu,
          parm2 = sigma,
          lower.tail = FALSE)
```



The probability of values greater than 170 would be about 28%.

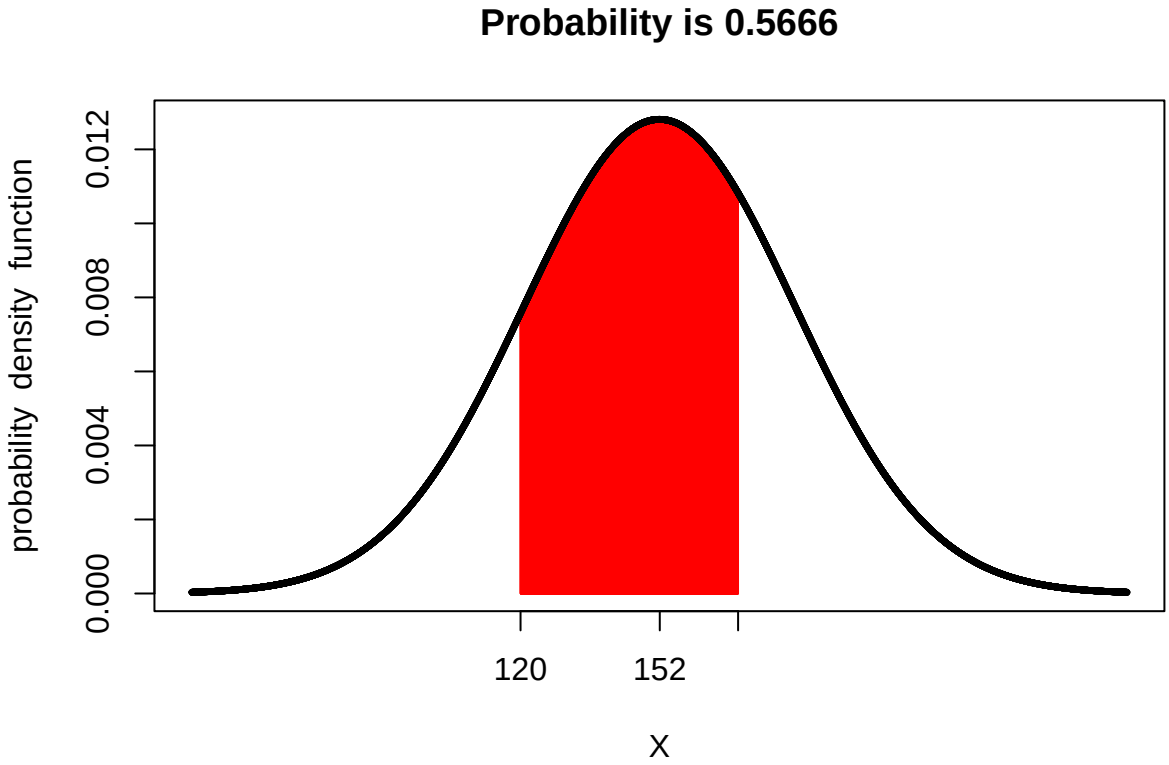
```
shadeDist(xshade = 170,
          ddist = "dnorm",
          parm1 = mu,
          parm2 = sigma,
          lower.tail = FALSE)
```



Probabilities Within A Range

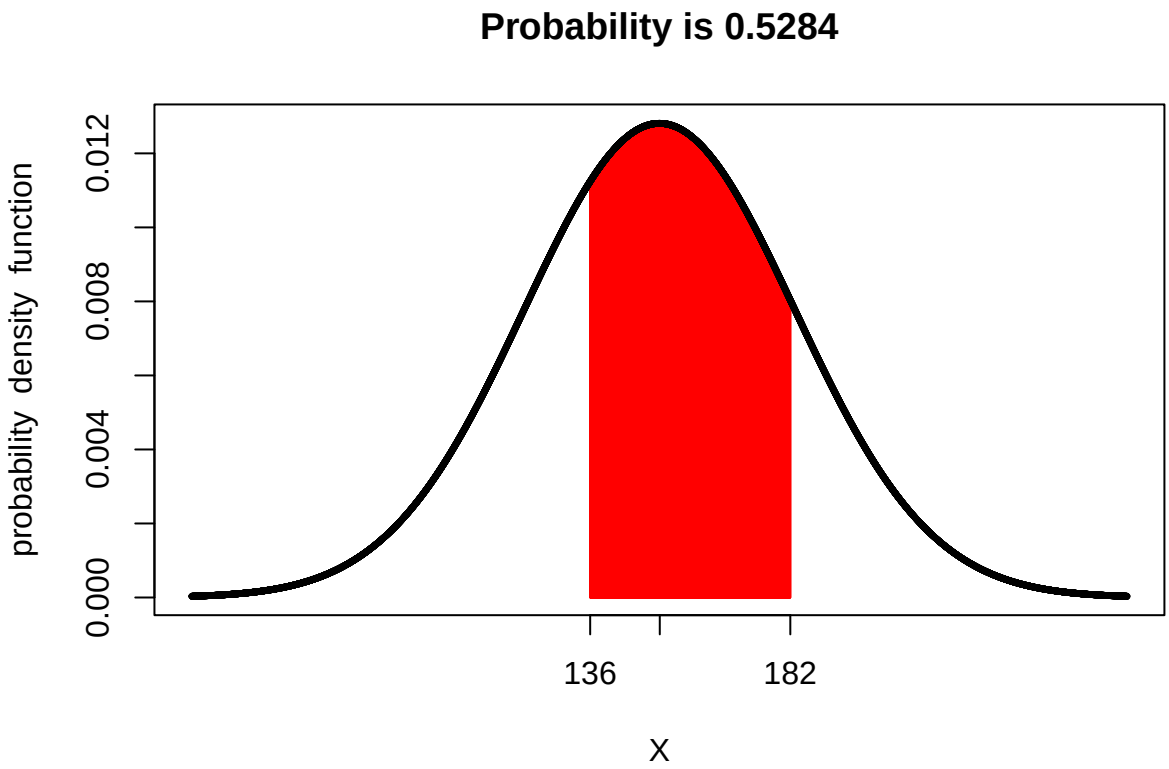
Sometimes we would like to measure the probability of a range of values in the middle of our distribution. To do this, we change our `xshade`argument to reference two values: the least and greatest values in that range. For example, to measure the probability of values between 120 and 170, we will use the argument `xshade = c(120, 170)`. We use `c()` whenever we are giving R a set of values to work with.

```
shadeDist(xshade = c(120, 170),
          ddist = "dnorm",
          parm1 = mu,
          parm2 = sigma,
          lower.tail = FALSE)
```



The probability of observing individuals with yields between 120 and 170 is about 57%. What about between 136 and 182 ?

```
shadeDist(xshade = c(136, 182),
          ddist = "dnorm",
          parm1 = mu,
          parm2 = sigma,
          lower.tail = FALSE)
```

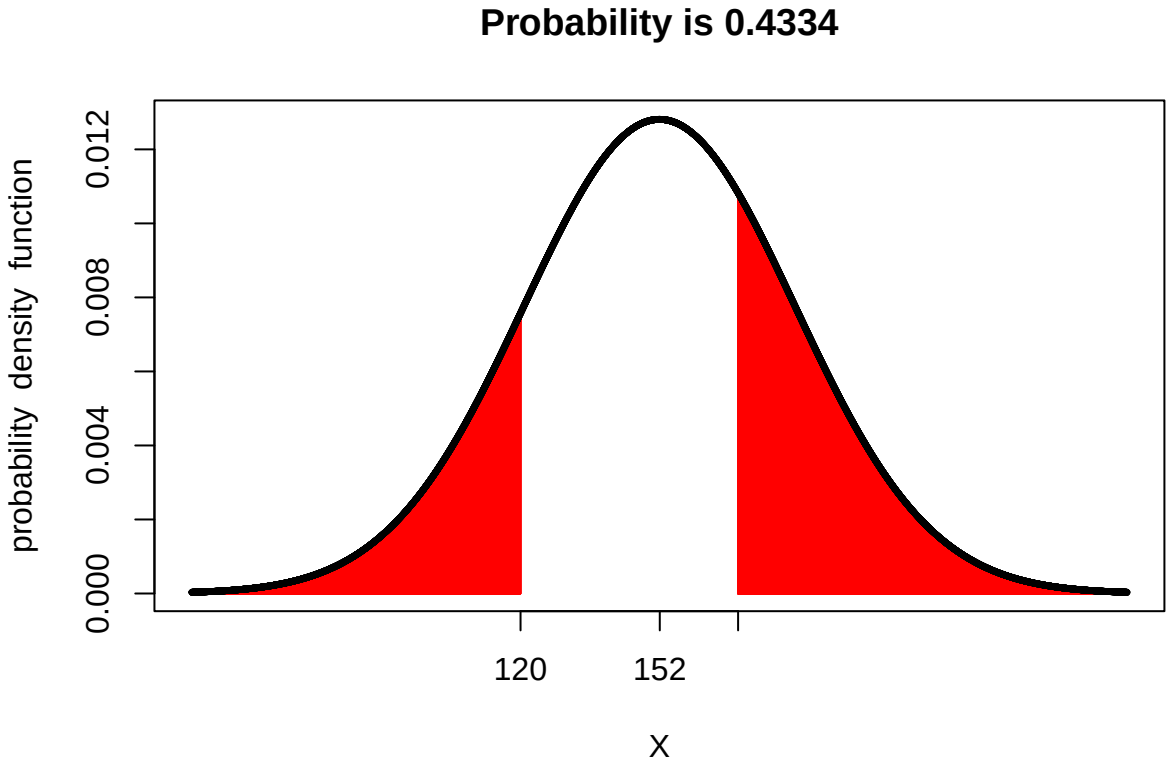


The probability is about 53%.

Probabilities Outside a Range

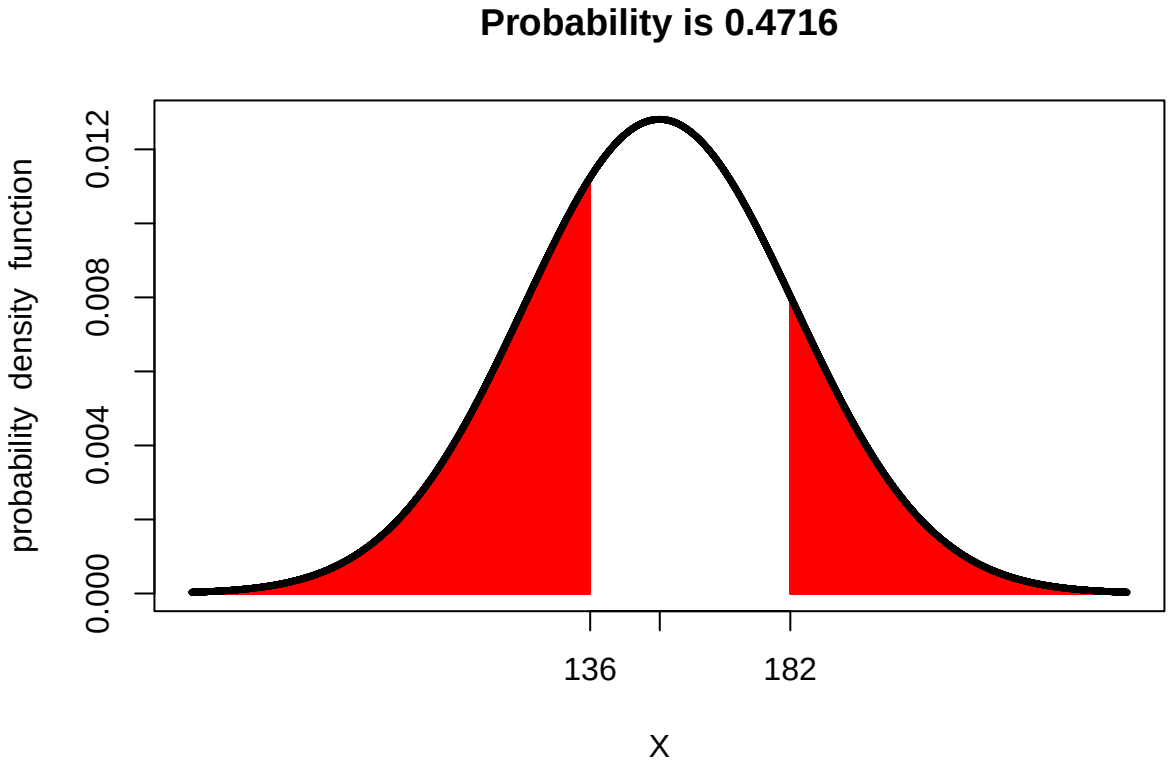
Finally, what if we wanted to measure the probability of observing a value outside the ranges in the previous section. To do this, we use the `lower.tail=TRUE` argument to the `shadeDist` function.

```
shadeDist(xshade = c(120, 170),
          ddist = "dnorm",
          parm1 = mu,
          parm2 = sigma,
          lower.tail = TRUE)
```



The probability of observing individuals with yields less than 120 or greater than 170 is about 43%. What about outside 136 and 182?

```
shadeDist(xshade = c(136, 182),
          ddist = "dnorm",
          parm1 = mu,
          parm2 = sigma,
          lower.tail = TRUE)
```



The probability of observing individuals with values less than 136 or greater than 182 is about 47%.

Probability and SD

We learned during the lesson that about 68% of individuals would be expected to have values with one standard deviation of the mean. We can use `shadeDist()` to verify that. First we need to determine the values for the mean minus one standard deviation and the mean plus one standard deviation. Given that we assigned the mean and standard deviation to the variables mu and sigma above, this is easy.

```
mu_minus_sigma = mu-sigma
mu_plus_sigma = mu+sigma
print(mu_minus_sigma)
```

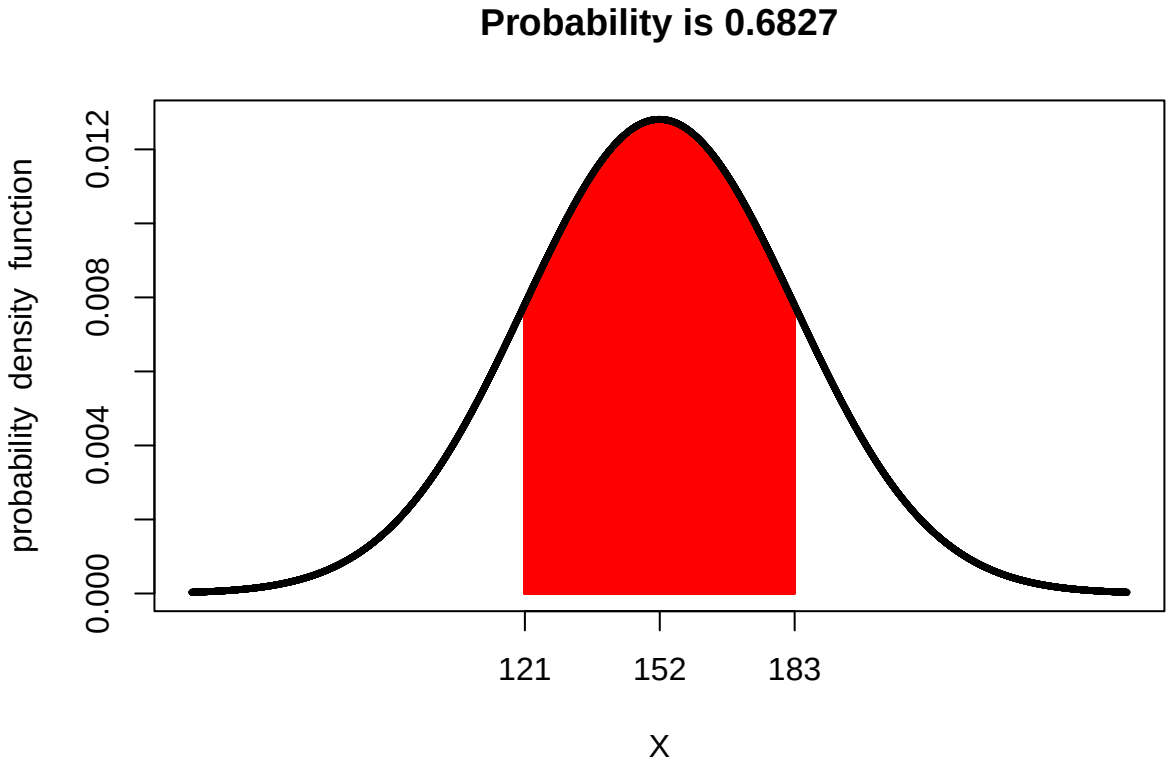
```
## [1] 120.7797
```

```
print(mu_plus_sigma)
```

```
## [1] 183.0603
```

We can see the population mean minus one standard deviation is about 120.8, and the mean plus one standard deviation is about 183.1 since we have assigned these variables to `mu_minus_sigma`, and `mu_plus_sigma`, we can pass them directly to the shadeDist function. Finally, we need to use "lower.tail=FALSE" to tell shadeDist to calculate the percentage of values between these two numbers.

```
shadeDist(xshade = c(mu_minus_sigma, mu_plus_sigma),
          ddist = "dnorm",
          parm1 = mu,
          parm2 = sigma,
          lower.tail = FALSE)
```



We confirm the percentage of observations between the population mean minus one standard deviation and the population mean plus one standard deviation is about 68%.

We also learned that about 95% of the population occurs between 1.96 standard deviations below the mean and 1.96 standard deviations above the mean. We can calculate these values as above.

```
mu_minus_1.96_sigma = mu-(1*sigma)
mu_plus_1.96_sigma = mu+(1*sigma)
print(mu_minus_1.96_sigma)
```

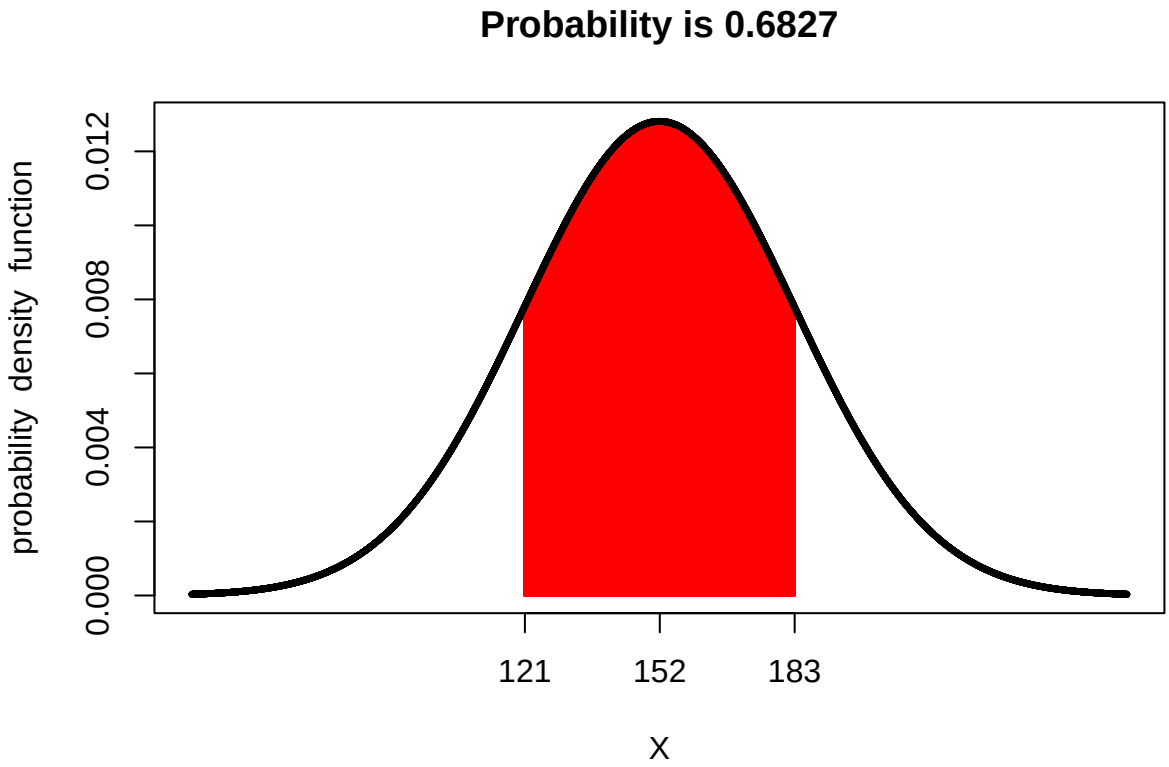
```
## [1] 120.7797
```

```
print(mu_plus_1.96_sigma)
```

```
## [1] 183.0603
```

We can see the population mean minus one standard deviation is about 90.9, and the mean plus one standard deviation is about 213.0.

```
shadeDist(xshade = c(mu_minus_1.96_sigma, mu_plus_1.96_sigma),
          ddist = "dnorm",
          parm1 = mu,
          parm2 = sigma,
          lower.tail = FALSE)
```



We have confirmed 95% of individuals are between 1.96 standard deviations below and above the mean.

Practice: Cotton Dataset

Let's load the cotton data.

```
cotton = read.csv("data/cotton_uniformity.csv")
```

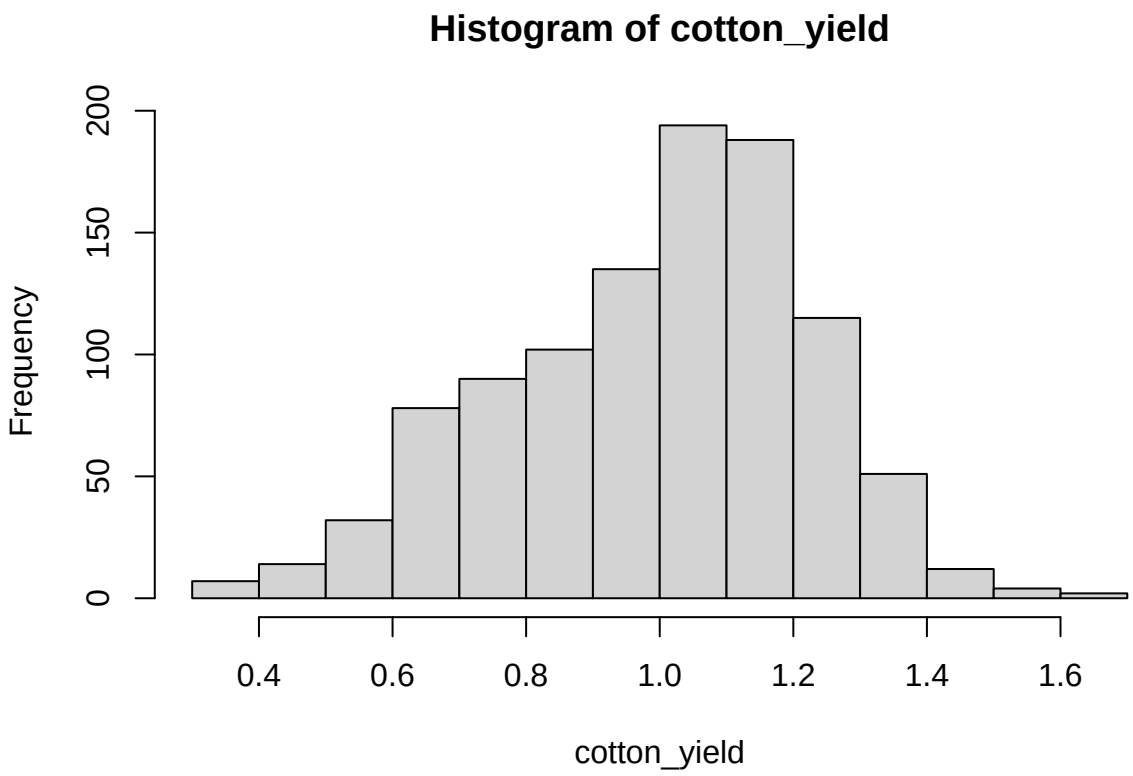
And examine its top six rows.

```
head(cotton)
```

```
##   col row yield block
## 1   1   1  1.23    B1
## 2   2   1  0.90    B1
## 3   3   1  0.97    B1
## 4   4   1  1.10    B1
## 5   5   1  0.99    B1
## 6   6   1  1.43    B1
```

The data we want are in the yield column. Let's isolate those and plot them in a histogram.

```
cotton_yield = cotton$yield
hist(cotton_yield)
```



Let's calculate our cotton mean and standard deviation.

```
cotton_mu = mean(cotton_yield)
cotton_sigma = sd(cotton_yield)
print(cotton_mu)
```

```
## [1] 1.001309
```

```
print(cotton_sigma)
```

```
## [1] 0.2287059
```

Now let's explore:

1. What is the probability an individual in the cotton popualation has a value less than 0.7? (You should get a value of about 0.09, or 9%.)
2. What is the probability an individual has a value greater than 1.4? (You should get a value around 0.4, or 4%.)
3. What is the probability of an individual having a value between 0.7 and 1.4? (Answer: about 87%)
4. What is the probability of an individual having a value less than 0.7 or greater than 1.4? (Answer: about 13%)

Practice: Tomato Dataset

Analyze the tomato_uniformity.csv dataset on your own:

1. What is the probability an individual will have a value less than 40? (Answer: about 16%)
2. What is the probability an individual will have a value greater than 60? (Answer: about 18%)

3. What is the probability an individual will have a value greater than 40 but less than 60? (Answer: about 66%)
4. What is the probability an individual will have a value less than 40 but greater than 60? (Answer: about 34%)