Normal models

Put your name here

Put the date here

Introduction

In this assignment we will learn how to work with normal models.

Instructions

Presumably, you have already created a new project and downloaded this file into it. From the `Run` menu above, select `Run All` to run all existing code chunks.

When prompted to complete an exercise or demonstrate skills, you will see the following lines in the document:

ANSWER

These lines demarcate the region of the R Markdown document in which you are to show your work.

Sometimes you will be asked to add your own R code. That will appear in this document as a code chunk with a request for you to add your own code, like so:

```
# Add code here
```

Be sure to remove the line `# Add code here` when you have added your own code. You should run each new code chunk you create by clicking on the dark green arrow in the upper-right corner of the code chunk.

Sometimes you will be asked to type up your thoughts. That will appear in the document with the words, "Please write up your answer here." Be sure to remove the line "Please write up your answer here" when you have written up your answer. In these areas of the assignment, please use contextually meaningful full sentences/paragraphs (unless otherwise indicated) and proper spelling, grammar, punctuation, etc. This is not R code, but rather a free response section where you talk about your analysis and conclusions. You may need to use inline R code in these sections.

When you are finished with the assignment, knit to PDF and proofread the PDF file **carefully**. Do not download the PDF file from the PDF viewer; rather, you should export the PDF file to your computer by selecting the check box next to the PDF file in the Files plane, clicking the `More` menu, and then clicking `Export`. Submit your assignment according to your professor's instructions.

Load Packages

We load the standard `mosaic` package.

```
library(mosaic)
```

The Central Limit Theorem

An important aspect of all the simulations that we've done so far—assuming that we've run a large enough number of them—is that their histograms all look like bell curves. This fact is known as the "Central Limit

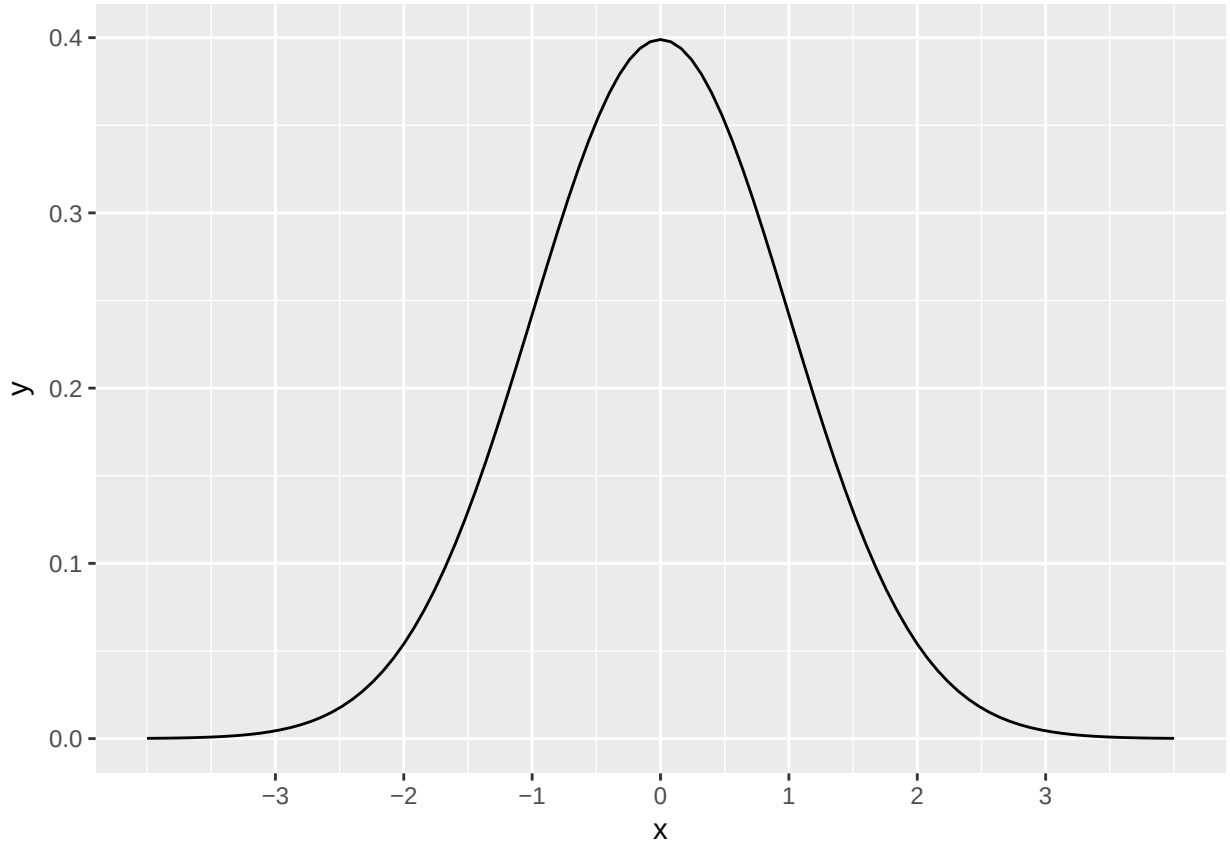
Theorem". Under some basic assumptions that we'll discuss in a later assignment, this will be typical of many of our simulated null distributions.

So rather than running a simulation each time we want to conduct a hypothesis test, we could also assume that the null distribution *is* a bell curve. The rest of this assignment will teach you how to work with the "normal distribution," which is just the mathematically correct term for a bell curve.

Normal models

The normal distribution looks like this:

```
# Don't worry about the syntax here.
# You won't need to know how to do this on your own.
ggplot(data.frame(x = c(-4, 4)), aes(x)) +
    stat_function(fun = dnorm) +
    scale_x_continuous(breaks = -3:3)
```



The curve pictured above is called the *standard normal distribution*. It has a mean of 0 and a standard deviation of 1. Mathematically, this is often written as

$$N(\mu = 0, \sigma = 1),$$

or sometimes just

$$N(0, 1).$$

We use this bell curve shape to model data that is unimodal, symmetric, and without outliers. A statistical "model" is a simplification or an idealization. Reality is, of course, never perfectly bell-shaped. Real data is not exactly symmetric with one clear peak in the middle. Nevertheless, an abstract model can give us good answers if used properly.

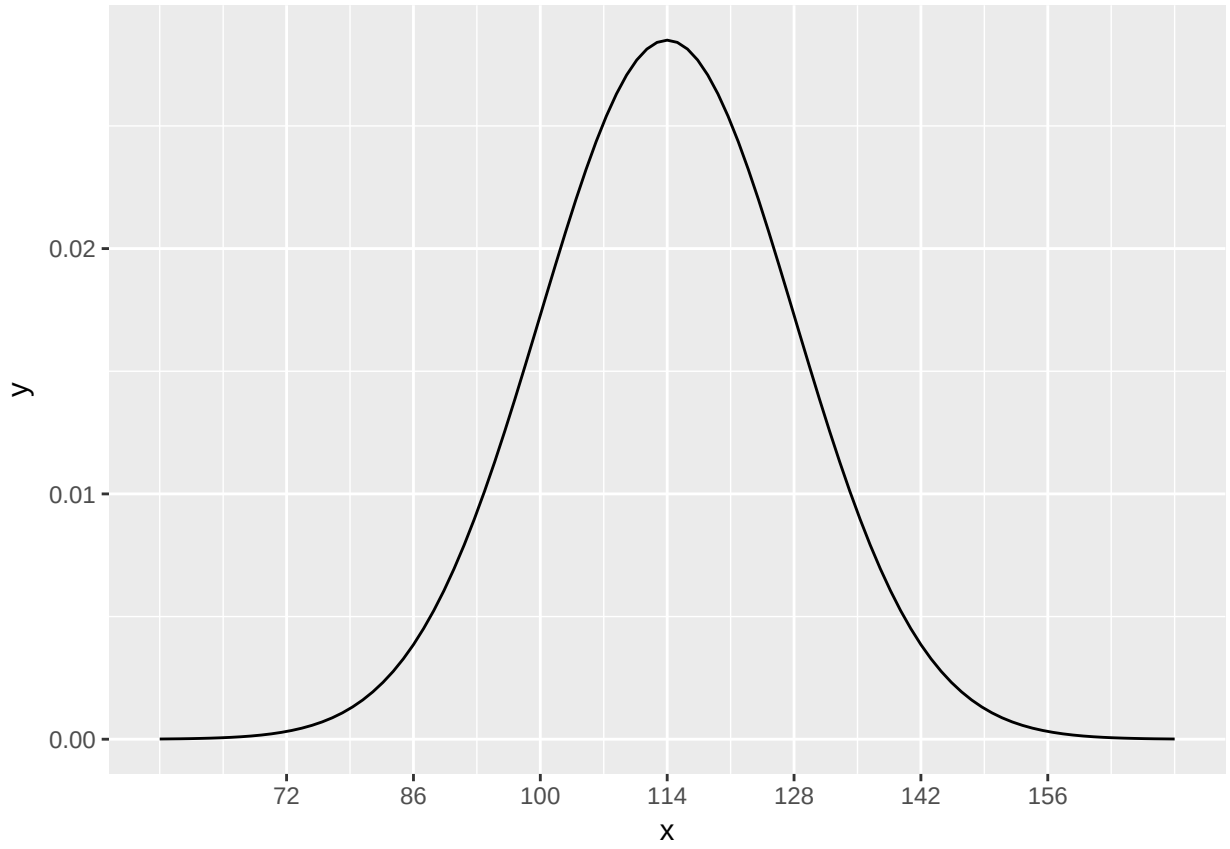
As an example of this, systolic blood pressure (SBP, measured in millimeters of mercury, or mmHg) is more-or-less normally distributed in women ages 30–44 in the U.S. and Canada, with a mean of 114 and a standard deviation of 14.[1]

If we were to plot a histogram with the SBP of every woman between the ages of 30 and 44 in the U.S. and Canada, it would have the shape of a normal distribution, but instead of being centered at 0 like the graph above, this one would be centered at 114. Mathematically, we write

$$N(\mu = 114, \sigma = 14).$$

The graph now looks like this:

```
# Again, don't worry about the syntax here.
ggplot(data.frame(x = c(58, 170)), aes(x)) +
    stat_function(fun = dnorm, args = list(mean = 114, sd = 14)) +
    scale_x_continuous(breaks = c(72, 86, 100, 114, 128, 142, 156))
```



[1]Statistics from the World Health Organization: http://www.who.int/publications/cra/chapters/volume1/0281-0390.pdf

Predictions using normal models

Using this information, we can estimate the percentage of such women who are expected to have any range of SBP without having access to all such data.

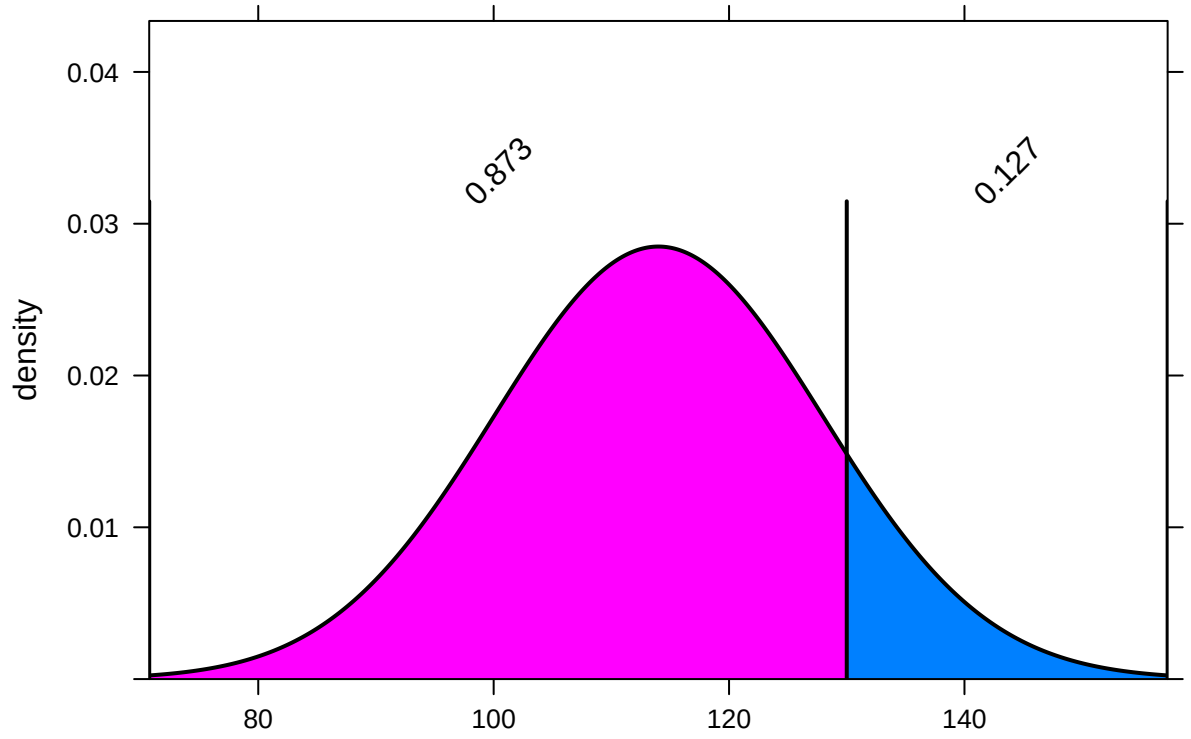
For example, what percentage of women ages 30–44 in the U.S. and Canada are expected to have SBP under 130 mmHg? The `pdist` command from the `mosaic` package will not only help you with this calculation, but it also offers a nice visual representation depending on the arguments you supply to the function. If you use `plot = FALSE`, all you get is the number:

```
pdist("norm", q = 130, mean = 114, sd = 14, plot = FALSE)
```

```
## [1] 0.873451
```

If you use `invisible = TRUE`, you get a plot instead:

```
pdist("norm", q = 130, mean = 114, sd = 14, invisible = TRUE)
```



The other pieces of the `pdist` function are pretty intuitive: `"norm"` (and it has to be in double quotes) indicates that we want a normal model, `q` is the value of interest to us, and `mean` and `sd` are self-evident. The numerical output gives the area under the curve to the left of our value of interest. In other words, 0.873451, or about 87.3% of women are expected to have SBP less than 130.

If you use this command inline, you don't need to specify `invisible` or `plot`; the pretty picture is not generated, just the value. For example, look at the following sentence (remembering that you can click anywhere inside the inline R code and hit Ctrl-Enter or Cmd-Enter):

> The model predicts that 87.3451046% of women ages 30–44 in the U.S. and Canada will have systolic blood pressure under 130 mmHg.

Note that the above code multiplied the result of the `pdist` command by 100. This is important because the full sentence interpretation is meant to be read by human beings, and human beings tend to report these kinds of numbers as percentages and not decimals.[2]

Here's another question: how many women are predicted to have SBP *greater* than 130? If 87.3% of women have SBP under 130, then 12.7% must have SBP over 130. Why? Because all women have to add up to 100%!

Therefore, all we have to do to solve this problem is subtract the number we obtained in the previous question from 1. (Remember that 1 = 100%.)

> The model predicts that 12.6548954% of women ages 30–44 in the U.S. and Canada will have systolic blood pressure over 130 mmHg.

Now, here's a more complicated question: what percentage of women are predicted to have SBP between 110 mmHg and 130 mmHg?

Recall that the proportion of women predicted to have SBP less than 130 mmHg was 0.873. But this is also counting women with SBP under 110 mmHg, whom we now want to exclude. The proportion of women with SBP under 110 is found with the following code:

```
pdist("norm", q = 110, mean = 114, sd = 14,  plot = FALSE)
```

```
## [1] 0.3875485
```

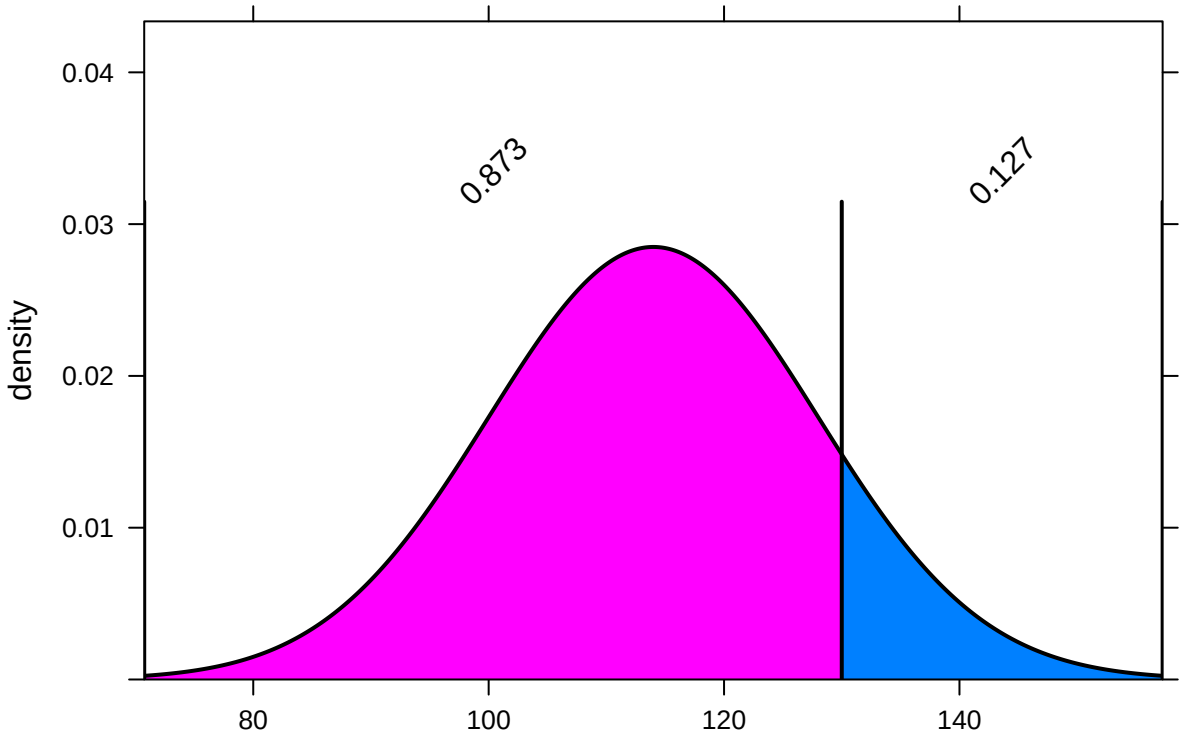
Therefore, all we have to do is calculate 0.873 minus 0.388:

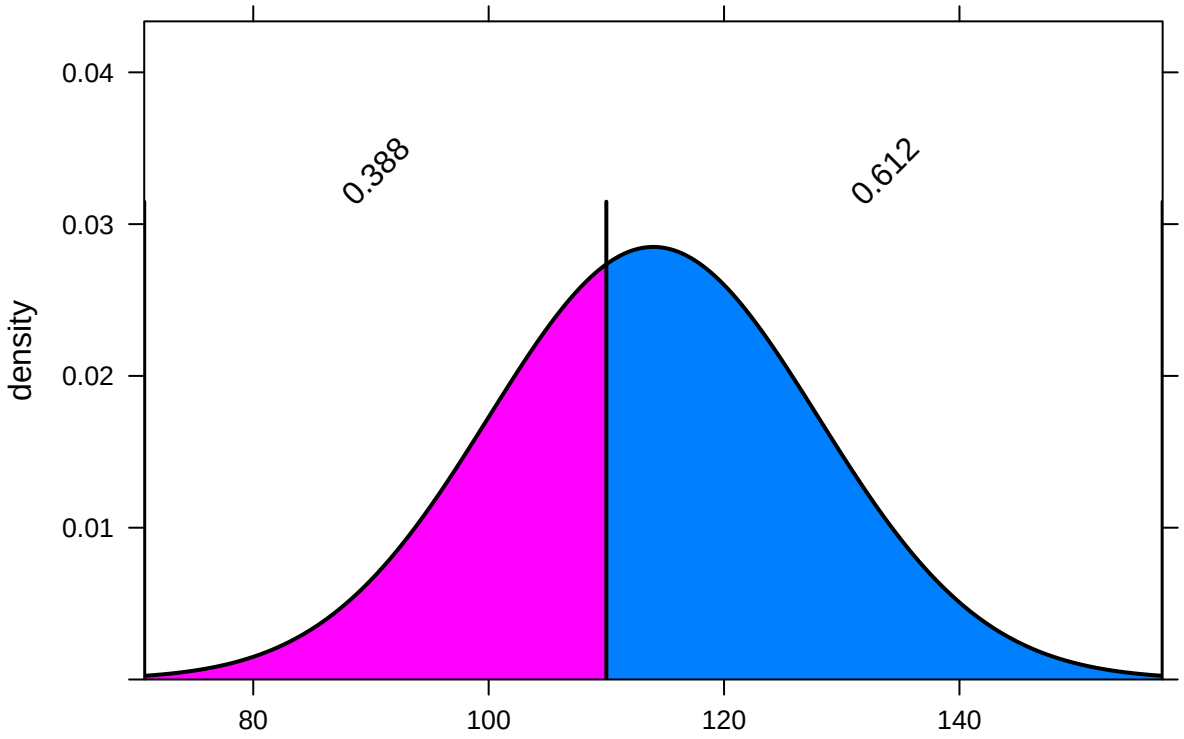
> The model predicts that 48.5902564% of women ages 30–44 in the U.S. and Canada will have systolic blood pressure between 110 mmHg and 130 mmHg.

What about the pretty picture? Unfortunately, this doesn't really work:

```
pdist("norm", q = 130, mean = 114, sd = 14, invisible = TRUE) -
    pdist("norm", q = 110, mean = 114, sd = 14, invisible = TRUE)
```

[2] When you eventually knit this to PDF, you'll see a ridiculous number of decimal places that R reports. It's a bit of a hassle to try to change it, so we'll just ignore the issue.

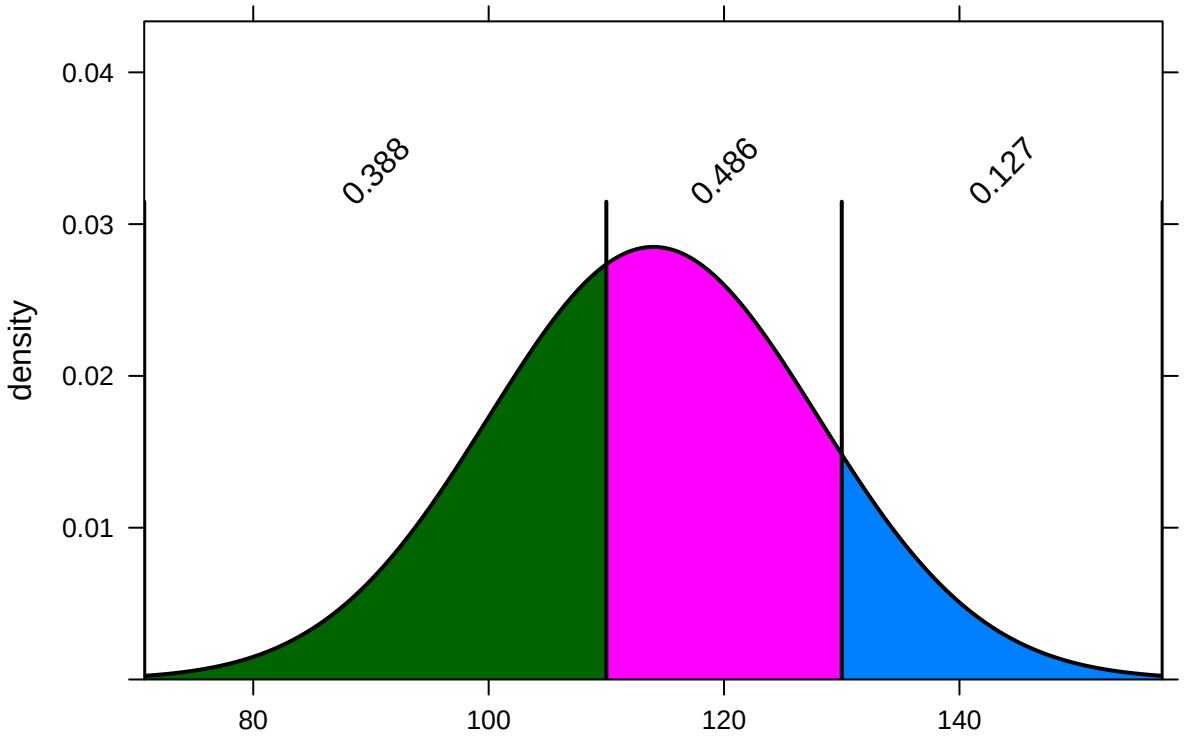

0.04
0.03
0.02
0.01
density
0.873
0.127
80
100
120
140




```
## [1] 0.4859026
```

Conceptually, this makes sense, but the R code is getting a little bulky, and it also prints both plots separately, making the output messy. Instead, let's observe that the `pdist` command can include both values (110 and 130) using the vector notation `c`:

```
pdist("norm", q = c(110, 130), mean = 114, sd = 14, invisible = TRUE)
```



Now the picture looks great and you can see the proportion you desire in the area between the two lines at 110 and 130.

This doesn't work so well for the numerical output though. Observe:

```
pdist("norm", q = c(110, 130), mean = 114, sd = 14,  plot = FALSE)
```

```
## [1] 0.3875485 0.8734510
```

There are two numbers shown, but neither is the correct answer. This command shows the percentages below 110 and below 130, respectively, but not the area in between 110 and 130. We still have to subtract. However, R can do this for us easily with the `diff` command that we've seen before:

```
diff(pdist("norm", q = c(110, 130), mean = 114, sd = 14, plot = FALSE))
```

```
## [1] 0.4859026
```

Again, for inline R code, you don't need to specify `plot = FALSE`:

> The model predicts that 48.5902564% of women ages 30–44 in the U.S. and Canada will have systolic blood pressure between 110 mmHg and 130 mmHg.

Exercises

IQ scores are standardized so that they have a mean of 100 and a standard deviation of 16.

For each of the following questions, use the `pdist` command with `invisible = TRUE` to draw the right picture and then state your answer in a contextually meaningful full sentence using inline R code. Don't forget to use the phrase "The model predicts..." and report numbers as percentages, not decimals.

What percentage of people would you expect to have IQ scores over 80?

ANSWER

```
# Add code here to draw the model.
```

Please write up your answer here.

What percentage of people would you expect to have IQ scores under 90?

ANSWER

```
# Add code here to draw the model.
```

Please write up your answer here.

What percentage of people would you expect to have IQ scores between 112 and 132?

ANSWER

```
# Add code here to draw the model.
```

Please write up your answer here.

Percentiles

Often, the question is reversed: instead of getting a value and being asked what percentage of the population falls above or below it, we are given a percentile and asked about the value to which it corresponds.

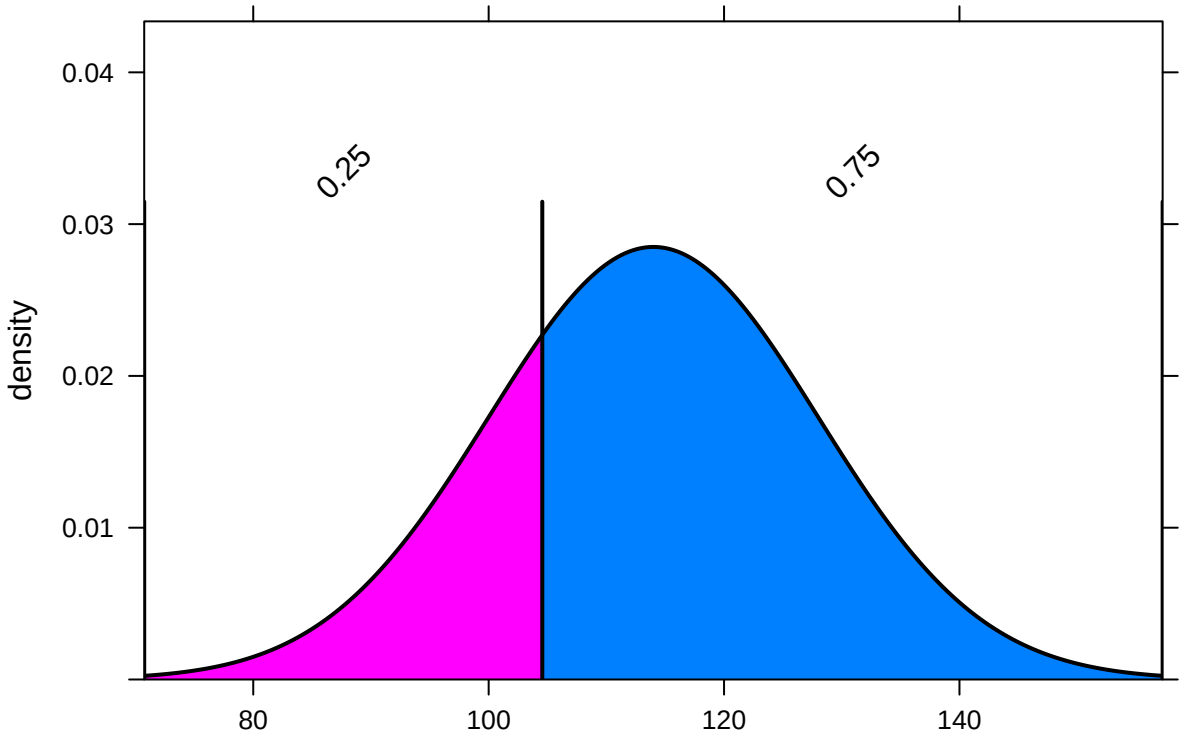
Here is an example using systolic blood pressure: what is the cutoff value of SBP for the lowest 25% of women ages 30–44 in the U.S. and Canada? In other words, what is the 25th percentile of SBP for this group of women?

The command we need is `qdist`. It looks a lot like `pdist`. Observe:

```
qdist("norm", p = 0.25, mean = 114, sd = 14,  plot = FALSE)
```

```
## [1] 104.5571
```

```
qdist("norm", p = 0.25, mean = 114, sd = 14,  invisible = TRUE)
```



The only change here is that one of the arguments is `p` instead of `q`, and the value of `p` is a proportion (between 0 and 1) instead of a value of SBP. The *output* is now an SBP value.

Here it is inline:

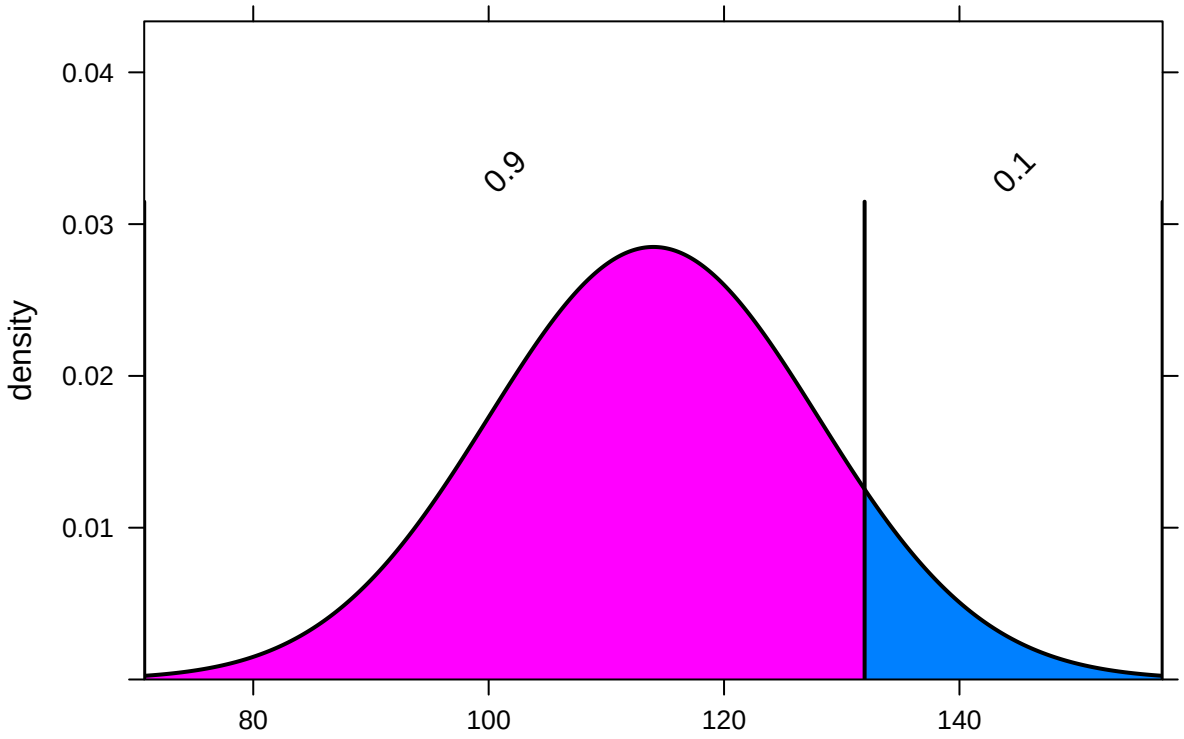
> The model predicts that the 25th percentile for SBP in women ages 30–44 in the U.S. and Canada is 104.5571435 mmHg.

What if we asked about the highest 10% of women? All you have to do is remember that the top 10% is actually the 90th percentile.

```
qdist("norm", p = 0.9, mean = 114, sd = 14, plot = FALSE)
```

```
## [1] 131.9417
```

```
qdist("norm", p = 0.9, mean = 114, sd = 14, invisible = TRUE)
```



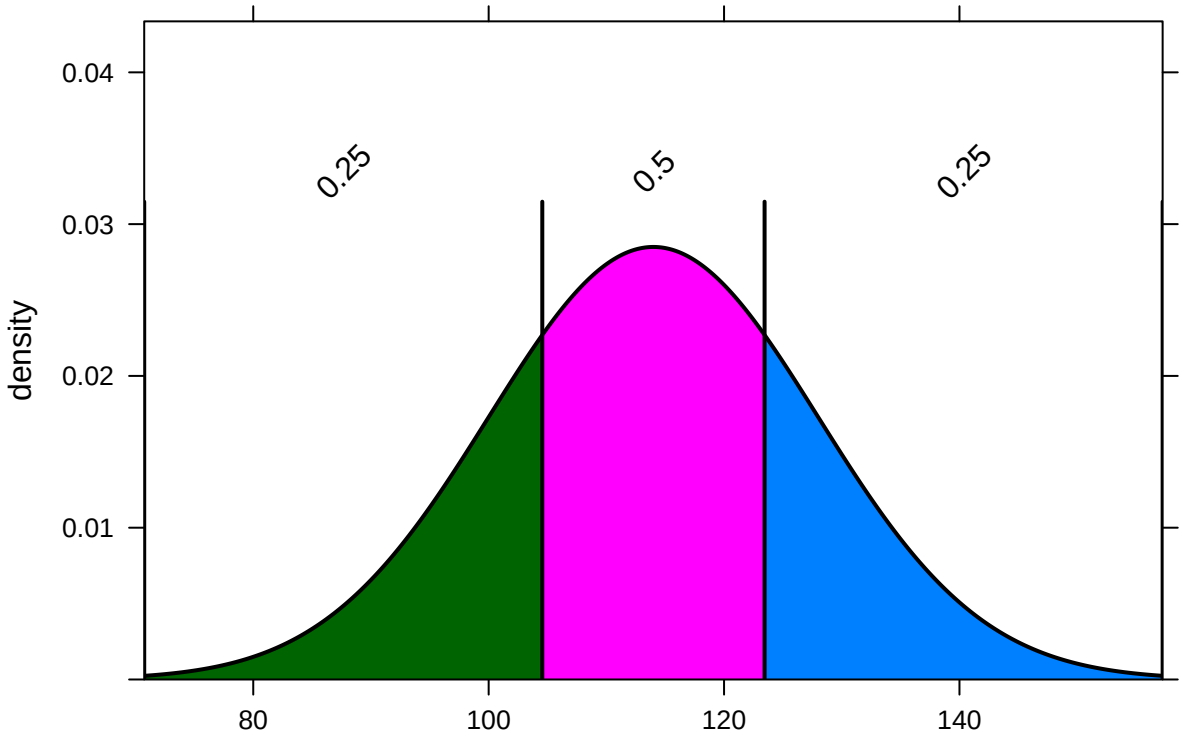
The model predicts that the top 10% of SBP in women ages 30–44 in the U.S. and Canada have SBP higher than 131.9417219 mmHg.

Finally, what if we want the middle 50%? This is trickier. The middle 50% lies between the 25th percentile and the 75th percentile. Observe the syntax below:

```
qdist("norm", p = c(0.25, 0.75), mean = 114, sd = 14, plot = FALSE)
```

```
## [1] 104.5571 123.4429
```

```
qdist("norm", p = c(0.25, 0.75), mean = 114, sd = 14, invisible = TRUE)
```



Therefore, the model predicts that the middle 50% of SBP for women ages 30–44 in the U.S. and Canada lies between 104.5571435 mmHg and 123.4428565 mmHg.

We did something tricky in the inline code above. Because the `qdist` command produces two values (one at the 25th percentile and one at the 75th percentile), we can grab each value separately by appending `[1]` or `[2]` to the end of the command.

Exercises

We'll continue to use IQ scores (mean of 100 and standard deviation of 16).

For each of the following questions, use the `qdist` command to draw the right picture and then state your answer in a contextually meaningful full sentence. Don't forget to use the phrase "The model predicts. . . "

What cutoff value bounds the highest 5% of IQ scores?

ANSWER

```
# Add code here to draw the model.
```

Please write up your answer here.

What cutoff value bounds the lowest 30% of IQ scores?

ANSWER

```
# Add code here to draw the model.
```

Please write up your answer here.

What cutoff values bound the middle 80% of IQ scores?

ANSWER

```
# Add code here to draw the model.
```

Please write up your answer here.

Conclusion

The normal model is ubiquitous in statistics, so understanding how to use it to make predictions is critical. When certain assumptions are met (that will be discussed in a future assignment), we can use the normal model to make predictions. Sometimes we have a value and we're interested in calculating a percentage of the population above or below that value. Other times, we want to calculate the value corresponding to a certain percentile. The `pdist` and `qdist` functions are very simple ways of visualizing and calculating using normal models.