# Unofficial Kahn Academy R Supplement: Z-Scores and Normal Distributions

There is no R function for calculating the z-score for a specific observation, largely because R primarily deals with variables and dataframes. However, it is relatively easy to calculate the z-score by hand. Once again we draw on the **mtcars** dataset. Here the Toyota Carolla has the highest fuel efficiency at a mpg of 33.9. To convert this into a z-score, we take the value (33.9), subtract the average mpg from that number, then divide that result by the standard deviation, or spread. Once again, you can either do this all at once in one line

```
(33.9 - mean(mtcars$mpg)) / sd(mtcars$mpg)
```

```
## [1] 2.291272
```

Or you can do it in pieces, save the individual results as objects, and do it sequentially

```
mean_mpg<-mean(mtcars$mpg)
sd_mpg<-sd(mtcars$mpg)
z= ((33.9-mean_mpg)/sd(mtcars$mpg))
z
```

```
## [1] 2.291272
```

At this point, of course, we'd go to a z-score table to look up the how much of the curve is to the left or right of our z-score. Of course, what use is statistical software if it doesn't save you from having to attend to onerous details like converting z-scores to probabilities! R has a command called *pnorm* that generates the probability of having a value below the value of the observation. Of course, if we want to find the probability of having a value higher than the X value, we simply subtract one from that number. We can directly convert the already-calculated z-score, or we can create the probability from the dataset itself.

```
# Here we are simply finding the probability given the z-score calculated above.
pnorm(z)
```

```
## [1] 0.9890261
```

```
#This is the probability that we'll find a value above the x-value.
1-pnorm(z)
```

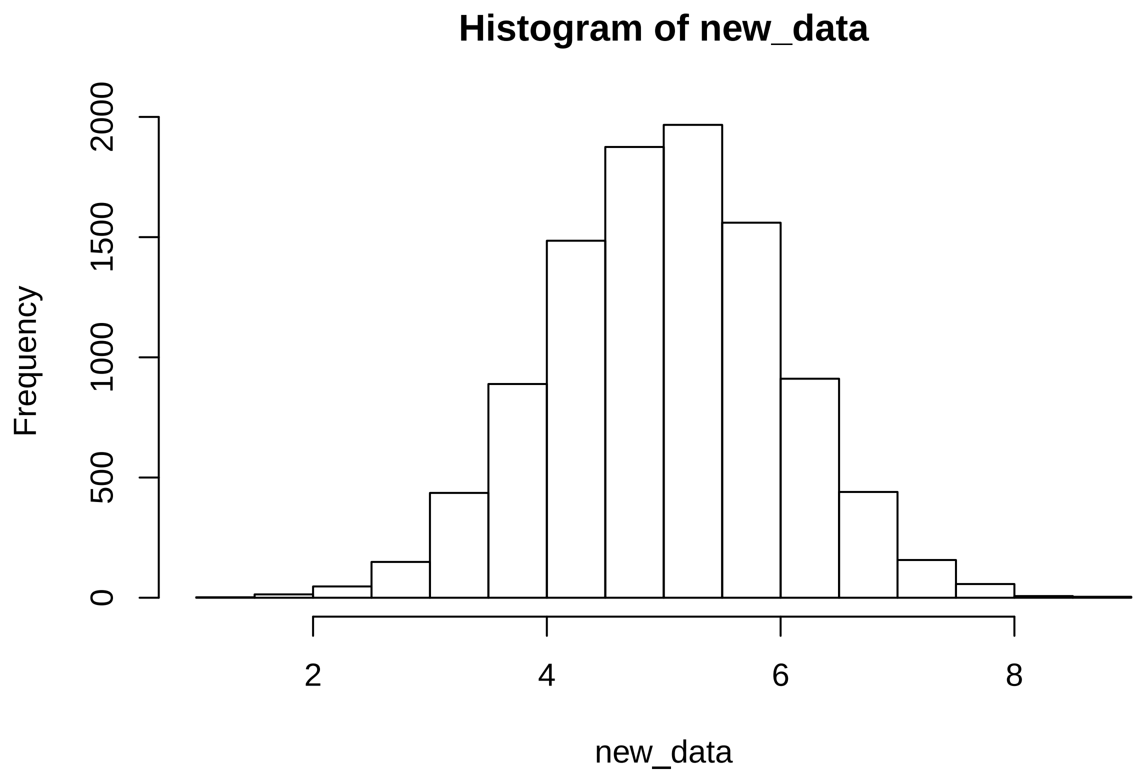
```
## [1] 0.01097385
```

```
#This is referring to the data itself so that we don't have to calculate the mean and sd separately.
pnorm(33.9, mean(mtcars$mpg), sd(mtcars$mpg))
```

```
## [1] 0.9890261
```

As noted in KA, the z-statistic is based on a normal distribution. There are other distributions that will be discussed on KA in the future, but the normal distribution is by far the most commonly used. There are various statistical tests for determining whether data are normally distributed (e.g. the Shapiro-Wilks test). One way to grasp normal distributions is to simulate your own dataset using R. the *rnorm* command allows you to create your own randomly-generated dataset given a specific mean, standard deviation, and number of observations. The *rnorm* command at its most basic has the form *rnorm(n, mean, sd)*, so in the case of a dataset of 10000 observations with a mean of five and a standard deviation of one, we have

```
new_data<-rnorm(10000, 5, 1)
hist(new_data)
```



As you can see, our randomly generated, normally distributed dataset looks, well, normally distributed, with a nice peak in the middle and tails on either side. One other note on things that R does that include randomness: because randomness (well, pseudo-randomness, but the explanation for that is further down your statistics education) is used, the numbers are going to be different every time. This can be a problem when you want your results to be replicable (somebody else can put in your code and get your same exact results). To get the best of both worlds, you can do what is called "setting the seed." Basically, the "random" process can run the exact same way if you set an initial starting value to be a certain way. For example, if I ran the above code multiple times, the data would look different every time, but if I used the *set.seed()* command, then it would have the same results every time. It really doesn't matter what your seed is. Generally, I always set my seed to one (so *set.seed(1)*), to make it clear that I'm not changing my seed until I get the answers I want, but you can choose some other way to select your starting seed.