Best Estimates and Errors

Mike Bader

June 24, 2016

In this section we will describe a very simple model, one so simple you probably learned it in middle school math: the mean. We all have a very practical sense of calculating means, we take the sum of values across observations and then divide by the number of observations. That helps us in lots of situations, but we rarely stop to think about what the mean, well, *means*. Let's think about that for a second.

The **mean** gives us our best estimate of the characteristic among a group of observations (if the characteristic follows a normal distribution). With no other information, we have a decent guess of what the height of any person in the room will be based on the mean of the group we study. In fact, we have the estimate that *minimizes* the amount of error among a group of observations. If you think about a scale, the mean sits at the fulcrum where the errors on one size total the errors on the other. That is why it represents the best estimate of data.

Let's think about the height of people in this room.

```r
class.heights <- rnorm(10,65,5)
class.height.mean <- mean(class.heights)
class.height.mean
```

```
## [1] 64.38096
```

```r
class.height.sd <- sd(class.heights)
```

For any one person, we will expect to have some error. But when we add all of the people together, the error will be zero. If, therefore, we pulled one person at a time, the errors on either side would eventually balance out.

By itself, however, the mean is not particulalry useful. Just knowing that we will – over the entire sample – be above or below doesn't give us much knowledge about how much above or how much below we will misestimate someone's height. When we combine the mean with the **standard deviation**, we can really improve our estimate. Now we can attach some level of probability to our estimate. We know that about 68% of people will be within $\pm$ 1 s.d. of the mean, about 95% of people will be within $\pm$ 2 s.d., and 99.7% will be within $\pm$ 3 s.d.

Generating Data with a Population with a Known Mean and Standard Deviation

Now we will begin, as I mentioned before, by making up our data. We don't have a particulalry large class, so it's not the best population to work from. But let's assume momentarily that we were randomly sampled from a larger population and we happened to hit the mean and standard deviation perfectly.

Formally, we will define a process that creates a population of 100 people (i.e., $N = 100$) with a mean height of 64.38 and a standard deviation of 5.23.

Write the Process & Create the Data

Let's start by indicating what a generic representation of the population based on its mean would look like. It's pretty basic:

$$x_i = \overline{x} + e_i$$

Again, this is nothing terribly exciting, but it does help us begin to see how models work in the most basic of models.[1] On the left side of the equation, we have x_i with "x" standing in for the value we care about (height). Note the subscript i there. That i means that every observation in the data has its own value of x. Let's create these first two pieces of our population:

```
N <- 100
population <- data.frame(i=c(1:N))
population$x_i <- rnorm(100,mean=class.height.mean,sd=class.height.sd)
population[1:10,] #Shows only the first ten of 100 observations in the data
```

```
##     i      x_i
## 1   1 66.89916
## 2   2 67.62866
## 3   3 65.64822
## 4   4 62.73411
## 5   5 72.50793
## 6   6 56.73845
## 7   7 54.39946
## 8   8 69.25009
## 9   9 65.21671
## 10 10 68.98724
```

The next term in the model, $\overline{x}$, represents the mean height. Notice that *no* subscript exists on that term. That means the mean is the *same for every observation* in the data. We can represent that by adding a variable in our model that represents the mean:

```
population$x_bar = mean(population$x_i)
population[1:10,]
```

```
##     i      x_i    x_bar
## 1   1 66.89916 64.23247
## 2   2 67.62866 64.23247
## 3   3 65.64822 64.23247
## 4   4 62.73411 64.23247
## 5   5 72.50793 64.23247
## 6   6 56.73845 64.23247
## 7   7 54.39946 64.23247
## 8   8 69.25009 64.23247
## 9   9 65.21671 64.23247
## 10 10 68.98724 64.23247
```

Now we have the final term in the model, e_i, representing the "error" of the estimate summarizing reality. In this case the estimate is the mean ($\overline{x}$), the error is the deviation from that estimate. The i indicates that each observation has its own value, which equals the value of the actual height minus the estimate, $x_i - \overline{x}$ (which if you notice, is just a reconfiguration of the equation above). Let's go ahead and calculate that:

[1]Another way you might see the model written would be: $x \sim \mathcal{N}(\overline{x}, \sigma^2)$, which represents the same model without the equation but includes the distribution of the model without needing to express it in words.

```r
population$e_i <- population$x_i - population$x_bar
population[1:10,]
```

```
##     i      x_i    x_bar        e_i
## 1   1 66.89916 64.23247  2.6666900
## 2   2 67.62866 64.23247  3.3961869
## 3   3 65.64822 64.23247  1.4157465
## 4   4 62.73411 64.23247 -1.4983603
## 5   5 72.50793 64.23247  8.2754510
## 6   6 56.73845 64.23247 -7.4940255
## 7   7 54.39946 64.23247 -9.8330137
## 8   8 69.25009 64.23247  5.0176169
## 9   9 65.21671 64.23247  0.9842398
## 10 10 68.98724 64.23247  4.7547673
```

Analyze the Data

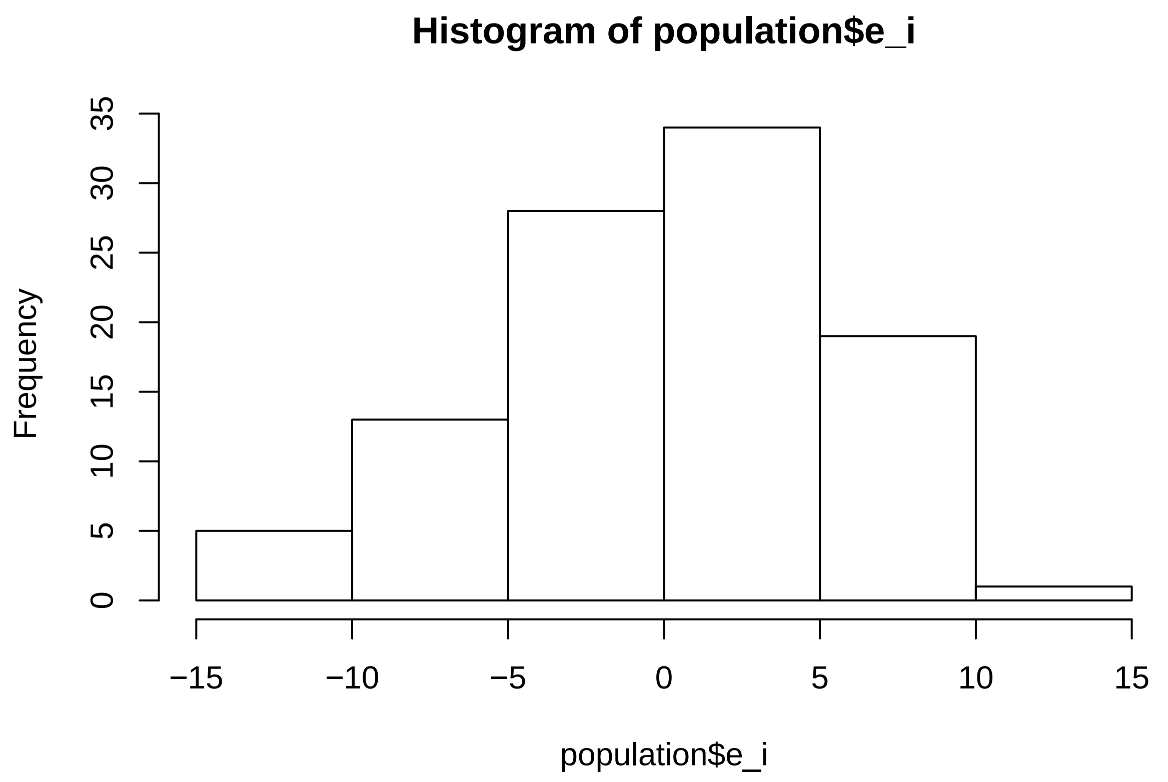
We have assumed that the heights of people in the population were normally distributed (well, really we constructed them to be so when we used the `rnorm()` command above). To express this, we would say that the errors are distributed with a mean of zero (the definition of the mean) and a standard deviation of σ^2 (which we also knew ahead of time when we generated the data). Let's see if, in fact, the errors really do sum to zero:

```r
sum(population$e_i)
```

```
## [1] -3.055334e-13
```

Yay! It's true (the tiny deviation from zero represents machine rounding error). And we can also look at the distribution of the errors to see if they look normal:

```r
hist(population$e_i)
```



Double yay! Now we can also see how we would analyze the variable if we had recorded the data without knowing the model that created the data (i.e., just the x_i column): :

```r
summary(population$x_i)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##   52.29   60.69   64.94   64.23   68.47   75.89 
```

Important note: Note above that I named our dataset `population`; this was intentional because the "error" we are talking about here represents the *deviation from the populaiton mean*. It *does not* represent any type of sampling error, this is just deviation from our single value representing our best guess of that population. When I talk about "error" initially, what I will be talking about is this deviation within the population data.

Now we have a) described a population process, b) written a model that summarizes that proces, c) created data following that process, and d) analyzed the data following that process. Not bad!

Generating a Process of Metropolitan House Price Values

Let's move on using the same steps for a new process, which we will use as an example throughout the course today. Let's think about metro-level home values and the median price per square foot of a new home in each metropolitan area. I am going to design a population of 150 metropolitan areas with a mean value of $100 per square foot. But home values, like lots of economic indicators, are not generally normally distributed. Rather, housing prices from the most expensive metropolitan area are generally a multiple higher than the second most, which are a multiple higher than the next. To fix that kind of distribution, we generally think in terms of logarithms. The natural log of 100 equals 4.61, so I will make the mean of our *normally distributed* population of metropolitan areas equal to 4.61. I imagine that a standard deviation of around 0.4 would be appropriate (meaning about 68% of metro median home values per square foot would be within about 40% of the mean).

Write the Real Estate Process and Create the Data

I am going to let $x = \ln(\text{price})$ and generate a population of (logged) home value per square foot among metros with the following rule:

$$x_i = \mu(x) + \epsilon_i$$

Note that this is the same model that I have above, except that I now use $\mu(x)$ to represent the mean and ϵ_i to represent the error, following standard conventions that we use Greek letters to represent parameters. Since I'm making a population, then all of my values will be parameters.

Let's go ahead and create the data:

```
N       <- 150
mu      <- 4.61
sigma   <- 0.4

metros <- data.frame(i=c(1:N))
metros$price_i <- rlnorm(N, meanlog=mu, sdlog=sigma)
metros$x_i <- log(metros$price_i)
```

First, we should check to make sure the population looks like it should. Let's check out the price:

```
hist(metros$price_i,breaks=15)
```

It looks log-normally distributed. Now let's check out the logged value of the price:

And it looks pretty normally distributed.

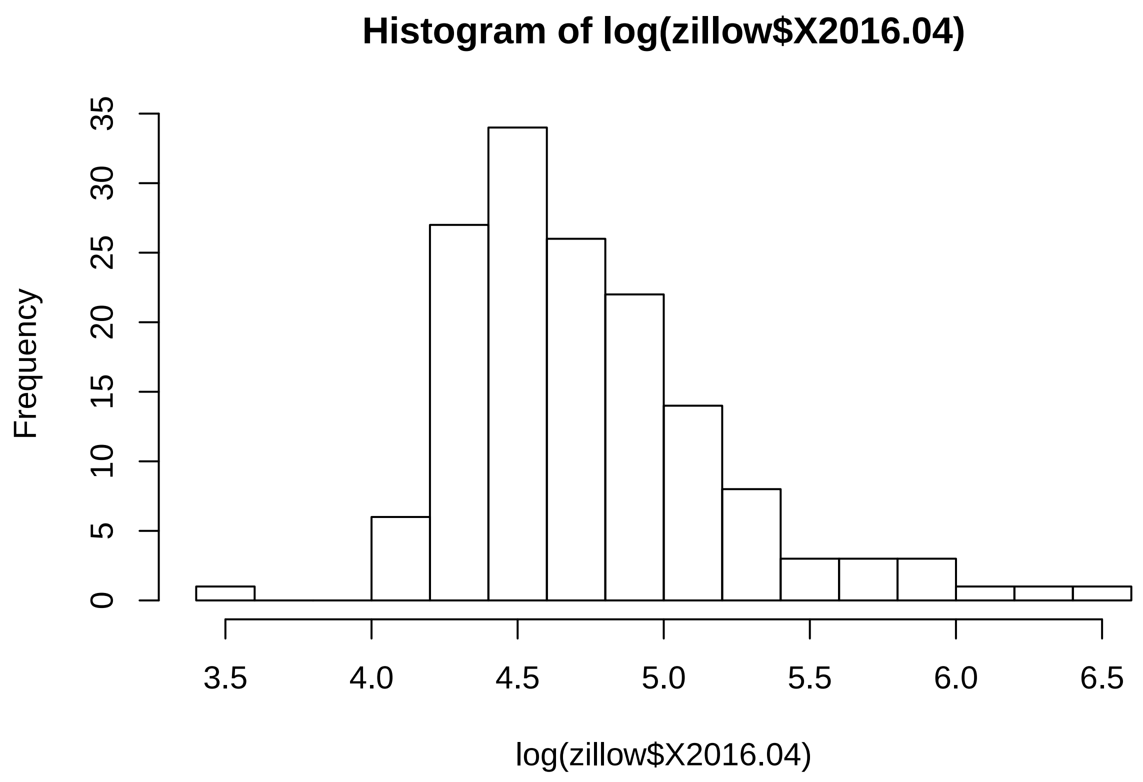
Analyze the Generated Metropolitan House Price Values

Let's look really quickly at the first several data lines of the dataset that we generated:

```
head(metros)
```

Rather, they tend to be logarithmically distributed, so rather than thinking about a

```
zillow <- read.csv('http://files.zillowstatic.com/research/public/Metro/Metro_MedianValuePerSqft_AllHom
hist(log(zillow$X2016.04),breaks=15)
```


Histogram of log(zillow$X2016.04)
Frequency
0
5
10
15
20
25
30
35
3.5
4.0
4.5
5.0
5.5
6.0
6.5
log(zillow$X2016.04)