Normal Distribution

2017-02-08

The probability distribution of a normal distribution is defined as

$$p(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left[-\frac{(x-\mu)^2}{2\sigma^2}\right]$$

for $-\infty \leq x \leq \infty$. Consider a normal distribution with a mean (μ) of 50, and a standard deviation (σ) of 10. We can compute the probability density ($p(x)$) for a particular x value by using this equation. For example, the probability density for $x = 65$ can be found using,

$$p(65) = \frac{1}{10\sqrt{2\pi}} \exp\left[-\frac{(65-50)^2}{2 \times 10^2}\right] = 0.01295176$$

Using R, we can carry out the computation,

```
(1/(10*sqrt(2*pi))) * exp(-(225)/200)
```

```
## [1] 0.01295176
```

There is also a more direct way to compute this using the `dnorm()` function. This function computes the density of `x` from a normal distribution with a specified `mean` and `sd`.

```
dnorm(x = 65, mean = 50, sd = 10)
```

```
## [1] 0.01295176
```

If we compute the density for several x values and plot them, we get the familiar normal shape; the graphical instantiation of the mathematical equation.

```
library(tidyverse)

data.frame(
  X = seq(from = 10, to = 90, by = 0.01)
) %>%
  rowwise() %>%
  mutate( Y = dnorm(x = X, mean = 50, sd = 10) ) %>%
  ggplot(data = ., aes(x = X, y = Y)) +
    geom_line() +
    theme_bw() +
    geom_point(x = 65, y = 0.01295176, size = 3)
```

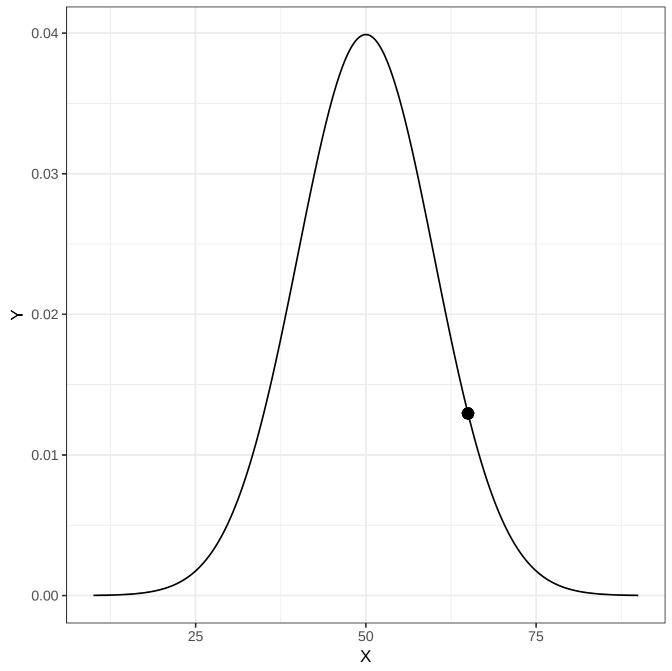


Figure 1. Plot of the probability density function (PDF) for a Normal distribution with mean of 50 and standard deviation of 10. The density value for $x = 65$, $p(65) = 0.01295176$, is also displayed on the PDF.

Other Useful R Functions for Working with Probability Distributions

There are four primary functions for working with the normal probability distribution:

- `dnorm()` : To compute the probability density (point on the curve)
- `pnorm()` : To compute the probability (area under the PDF)
- `qnorm()` : To compute the x value given a particular probability
- `rnorm()` : To draw a random observation from the distribution

Each of these requires the arguments `mean=` and `sd=`. Let's look at some of them in use.

Finding Cumulative Probability

The function `pnorm()` gives the probability x is less than or equal to some quantile value in the distribution; the cumulative probability. For example, to find the probability that $x \leq 65$ we would use,

```
pnorm(q = 65, mean = 50, sd = 10)
```

```
## [1] 0.9331928
```

This is akin to finding the proportion of the area under the normal PDF that is to the left of 65.

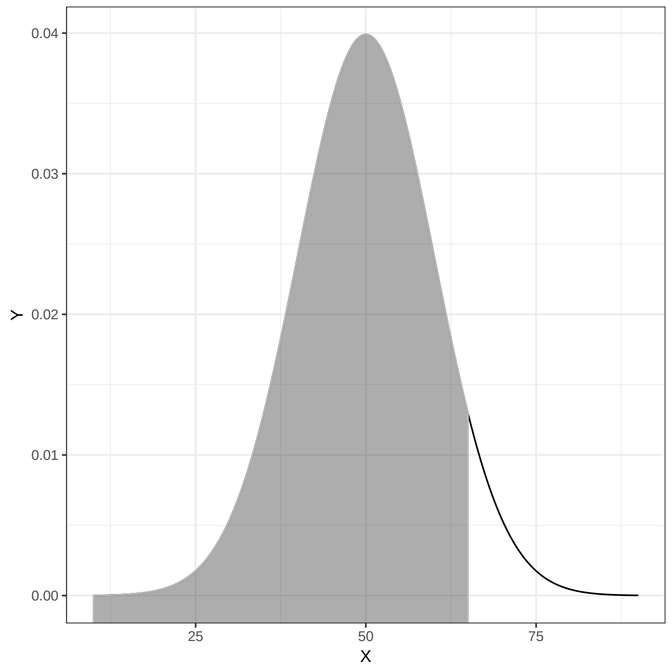


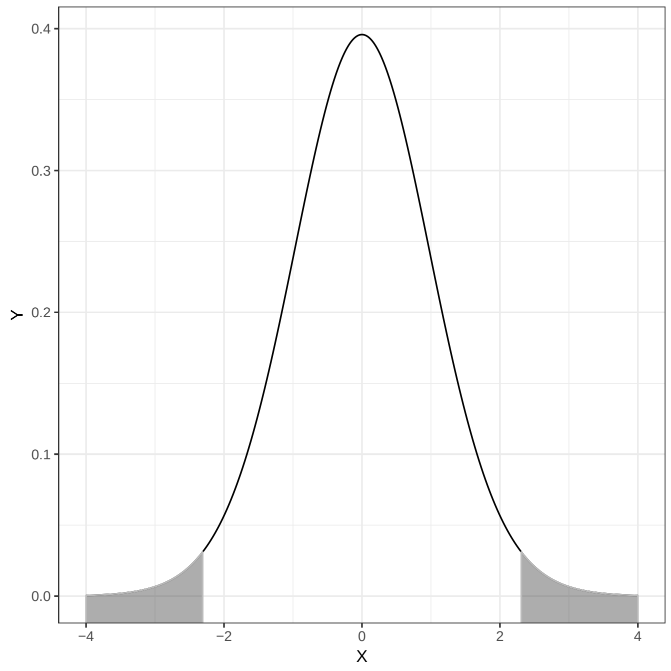
Figure 2. Plot of the PDF for a $\sim \mathcal{N}(50, 10)$ with the cumulative probability for $x \leq 65$ shaded.

For the mathematically inclined, the grey-shaded area is expressed as an integral

$$\int_{-\infty}^{65} p(x)dx$$

where $p(x)$ is the PDF for the normal distribution.

This type of computation is used most commonly to find a *p*-value. The *p*-value is just the area under the cureve that is AT LEAST as extreme as some observed value. In regression, for example, we compute the value for a slope. This is then converted to a t-value so it can be evaluated in a t-distribution. For example, say we have a t-value of 2.31. Then, the *p*-value can be graphically displayed in the t-distribution with 32 *df* as follows:



In most hypthesis tests of the slope, we just test whether the slope is equal to 0. Thus the values in the t-distribution more extreme than 2.31 encompass those values greater than 2.31 and also those values less than -2.31. (This is akin to testing a fair coin when both 8 heads OR 8 tails would provide evidence against fairness …we have to consider evidence in both directions).

This is not a normal distribution, it is a t-distribution, so we need to use `pt()` rather than `pnorm()`. The function `pt()` needs the argument `df=`. It also computes the proportion of the area under the curve TO THE LEFT of a particular value. Here we will compute the are to the left of -2.31 and then double it to produce the actual p-value.

```
2 * pt(q = -2.31, df = 32)
```

```
## [1] 0.02749139
```

Finding Quantiles

The `qnorm()` or `qt()` function is essentially the inverse of the `pnorm()` and the `pt()` functions. The `p` functions find the cumulative probability GIVEN a particular quantile. The `q` functions find the quantile GIVEN a cumulative probability. For example, in the normal distribution we defined earlier, half of the area is below the quantile value of 50 (the mean).

```
qnorm(p = 0.5, mean = 50, sd = 10)
```

```
## [1] 50
```

In the olden days, we would determine critical values to compare our t-values to in order to determine significance. For example, if we wanted to define significance as $p \leq .05$, that implied that the t-value would have to be in either the top .025 or the lowest .025 (total = .05) of the t-distribution.

```
qt(p = .025, df = 32)
```

```
## [1] -2.036933
```

In a t-distribution with 32 df this means that our t-value would have to be less than -2.04 or greater than 2.04.

Back to Probability Density

Probability density is the key to understanding likelihood estimation. The likelihood of a set of data is related to the idea of *joint probability density*. Joint probability density is simply the density of x_1, x_2, …AND x_k. If we can make an assumption about INDEPENDENCE, then the joint density would be the product of the individual densities,

$$p(x_1, x_2, x_3, \ldots, x_K) = p(x_1) \times p(x_2) \times p(x_3) \times \ldots \times p(x_k)$$

Say we had three indpendent observations from our $\sim \mathcal{N}(50, 10)$ distribution, namely $x = \{60, 65, 67\}$. Then the joint density would be,

```r
dnorm(x = 60, mean = 50, sd = 10) * dnorm(x = 65, mean = 50, sd = 10) * dnorm(x = 67, mean = 50, sd = 10)
```

```
## [1] 2.947448e-06
```

We could also shortcut this computation,

```r
prod(dnorm(x = c(60, 65, 67), mean = 50, sd = 10))
```

```
## [1] 2.947448e-06
```

This value is the joint probability density.

Likelihood

In the previous example, the joint probability density indicates the probability of observing the data ($x = \{60, 65, 67\}$) GIVEN (1) they are drawn from a normal distribution and (2) the normal distribution has a mean of 50 and a standard deviation of 10. In other words, it is the probability of the data given the distribution and parameters.

Likelihood is the probability of a particualar set of parameters GIVEN (1) the data, and (2) the data are from a normal distribution. In other words, it is the probability of the parameters given the data and the distribution.

We might ask the question, given the observed data $x = \{30, 20, 24, 27\}$ come from a normal distribution, what is the likelihood (probability) that the mean is 20 and the standard deviation is 4? To answer this we compute the joint density under that particular set of parameters.

```r
prod(dnorm(x = c(30, 20, 24, 27), mean = 20, sd = 4))
```

```
## [1] 5.702554e-07
```

What is the likelihood (probability) that the mean is 25 and the standard deviation is 4?

```r
prod(dnorm(x = c(30, 20, 24, 27), mean = 25, sd = 4))
```

```
## [1] 1.774012e-05
```

Which set of parameters is more likely given the data and that they come from a normal distribution? The second set of parameters is more likely since it has a higher likelihood.

Maximum Likelihood

So now we come to the crux of Maximum Likelihood. It is to choose a set of parameters that MAXIMIZES the likelihood given the data and a distribution. In our example, given the observed data $x = \{30, 20, 24, 27\}$ come from a normal distribution, what are the values for the paramters (mean and standard deviation) that give the highest liklihood?

We can again, use a grid search,

```r
expand.grid(
  mu = seq(from = 10, to =30, by = 0.1),
  sigma = seq(from = 0, to = 10, by = 0.1)
  ) %>%
  rowwise() %>%
  mutate( lik = prod(dnorm(c(30, 20, 24, 27), mean = mu, sd = sigma)) ) %>%
  arrange(desc(lik))
```

```
## # A tibble: 20,301 × 3
##       mu sigma          lik
##    <dbl> <dbl>        <dbl>
## 1   25.2   3.7 1.829129e-05
## 2   25.3   3.7 1.829129e-05
## 3   25.3   3.8 1.824024e-05
## 4   25.2   3.8 1.824024e-05
## 5   25.4   3.7 1.823793e-05
## 6   25.1   3.7 1.823793e-05
## 7   25.2   3.6 1.823544e-05
## 8   25.3   3.6 1.823544e-05
## 9   25.1   3.8 1.818979e-05
## 10  25.4   3.8 1.818979e-05
## # ... with 20,291 more rows
```

The parameters that maximize the likelihood (in our search space) are a mean of 25.2 and a standard deviation of 3.7.

Log-Likelihood

The likelihood values are quite small since we are multiplying several probabilities together. We could take the natural logarithm of the likelihood to alleviate this issue. So in our example, $\mathcal{L} = .00001829129$ and the log-likelihood would be

```r
log(.00001829129)
```

```
## [1] -10.90909
```

Going back to how we compute the likelihood,

$$
\begin{aligned}
\ln\Big(\mathcal{L}(\text{parameters}|\text{data})\Big) &= \ln\Big(p(\epsilon_1) \times p(\epsilon_2) \times \ldots \times p(\epsilon_n)\Big) \\
&= \ln\Big(p(\epsilon_1)\Big) + \ln\Big(p(\epsilon_2)\Big) + \ldots + \ln\Big(p(\epsilon_n)\Big)
\end{aligned}
$$

The log-likelihood is the sum of the log-transformed densities. This means we could also write our function to compute the log-likelihood.

```r
expand.grid(
  mu = seq(from = 10, to =30, by = 0.1),
  sigma = seq(from = 0, to = 10, by = 0.1)
  ) %>%
  rowwise() %>%
  mutate( log_lik = sum(log(dnorm(c(30, 20, 24, 27), mean = mu, sd = sigma))) ) %>%
  arrange(desc(log_lik))
```

```
## # A tibble: 20,301 × 3
##       mu sigma   log_lik
##    <dbl> <dbl>     <dbl>
## 1   25.2   3.7 -10.90909
## 2   25.3   3.7 -10.90909
## 3   25.2   3.8 -10.91188
## 4   25.3   3.8 -10.91188
## 5   25.1   3.7 -10.91201
## 6   25.4   3.7 -10.91201
## 7   25.3   3.6 -10.91214
## 8   25.2   3.6 -10.91214
## 9   25.1   3.8 -10.91465
## 10  25.4   3.8 -10.91465
## # ... with 20,291 more rows
```

Maximizing the log-likelihood gives the same parameter values as maximizing the likelihood.Remember that the log computation keeps the same ordination of values as the original data, so maximizing the log-likelihood is the same as maximizing the likelihood.

Maximum Likelihood Estimation for Regression

In model fitting, the components we care about are the residuals. Those are the things we put distributional assumptions on (e.g., normality, homogeneity of variance, independent). So we want to determine the parameter values (β_0, β_1) that maximize the likelihood for a given set of residuals that come from a normal distribution.

To understand this, let's go back to our toy example using the 10 X and Y values from the OLS notes.

```
# Enter data into vectors
x = c(4, 0, 3, 4, 7, 0, 0, 3, 0, 2)
y = c(53, 56, 37, 55, 50, 36, 22, 75, 37, 42)
```

We will write a function to compute the likelihood (or log-likelihood) of the residuals given a particular `b0` and `b1` estimate that will be inputted to the function. One issue is that in using the `dnorm()` function we need to specify the mean and standard deviation. The regression assumptions help with this.

The conditional mean residual value is 0. So we will set the mean value to 0. The asumption about the standard deviation is that the conditional distributions all have the same SD, but it doesn't speicfy what that is. However, the SD of the errors seems like a reasonable value, so let's use that.

Now we read in our function.

```
likelihood = function(b0, b1){
  errors = y - b0 - b1*x
  sigma = sd(errors)
  lik = prod(dnorm(errors, mean = 0, sd = sigma))
  return(lik)
}
```

Now we can use our function in a grid search.

```
expand.grid(
  b0 = seq(from = 30, to = 50, by = 0.1),
  b1 = seq(from = -5, to = 5, by = 0.1)
) %>%
  rowwise() %>%
  mutate( lik = likelihood(b0 = b0, b1 = b1) ) %>%
  arrange(desc(lik))
```

```
## # A tibble: 20,301 × 3
##       b0    b1          lik
##    <dbl> <dbl>        <dbl>
## 1   40.1   2.7 7.135597e-18
## 2   40.0   2.7 7.133956e-18
## 3   40.2   2.7 7.133136e-18
## 4   39.9   2.8 7.132213e-18
## 5   39.8   2.8 7.131803e-18
## 6   40.0   2.8 7.128523e-18
## 7   39.9   2.7 7.128215e-18
## 8   39.7   2.8 7.127293e-18
## 9   40.3   2.7 7.126575e-18
## 10  40.1   2.8 7.120739e-18
## # ... with 20,291 more rows
```

Here the parameter values that maximize the likelihood are $b0 = 40.1$ and $b1 = 2.7$. We can also compute what the standard deviation value was using the estimated parameter values.

```
errors = y - 40.1 - 2.7*x
sd(errors)
```

```
## [1] 13.18665
```

This value is an estimate of the RMSE. In practice, there are a couple subtle differences, namely that the estimate for the SD value we use in `dnorm()` is slightly different. This generally does not have an effect on the coefficient estimates, but does impact the estimate of the RMSE. We will talk more about this when we talk about *Restricted Maximum Likelihood Estimation* (REML).

Using R to Directly Compute the Likelihood and Log-Likelihood

We can use R to directly compute the log-likelihood after we fit a model using the `lm()` function. To do this, we use the `logLik()` function.

```
lm.1 = lm(y ~ 1 + x)
logLik(lm.1)
```

```
## 'log Lik.' -39.45442 (df=3)
```

To compute the likelihood, we can use the `exp()` function to back-transform the log-likelihood to the likelihood (although generally we will work with the log-likelihood).

```
exp(-39.45442)
```

```
## [1] 7.330998e-18
```

Way, Way, Way too Much Mathematics

Let's also expound on this more mathematically. Going back to the mathematical notation, we can specify the liklihood as,

$$\mathcal{L}(\beta_0, \beta_1 | \text{data}) = p(\epsilon_1) \times p(\epsilon_2) \times \ldots \times p(\epsilon_n)$$

where

$$p(\epsilon_i) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left[-\frac{(\epsilon_i - \mu)^2}{2\sigma^2}\right]$$

The regression assumptions specify that each conditional error distribution is normal distributed with a mean of 0 and some variance (that is the same for all conditional error distributions), we can call it σ^2. Substituting these things in to the density function, we get,

$$
\begin{aligned}
p(\epsilon_i) &= \frac{1}{\sigma\sqrt{2\pi}} \exp\left[-\frac{(\epsilon_i - 0)^2}{2\sigma^2}\right] \\
&= \frac{1}{\sigma\sqrt{2\pi}} \exp\left[-\frac{(\epsilon_i)^2}{2\sigma^2}\right]
\end{aligned}
$$

Now we use this expression for each of the $p(\epsilon_i)$ values in the liklihood computation.

$$
\begin{aligned}
\mathcal{L}(\beta_0, \beta_1 | \text{data}) &= p(\epsilon_1) \times p(\epsilon_2) \times \ldots \times p(\epsilon_n) \\
&= \frac{1}{\sigma\sqrt{2\pi}} \exp\left[-\frac{\epsilon_1^2}{2\sigma^2}\right] \times \frac{1}{\sigma\sqrt{2\pi}} \exp\left[-\frac{\epsilon_2^2}{2\sigma^2}\right] \times \ldots \times \frac{1}{\sigma\sqrt{2\pi}} \exp\left[-\frac{\epsilon_n^2}{2\sigma^2}\right]
\end{aligned}
$$

We can simplify this

$$
\mathcal{L}(\beta_0, \beta_1 | \text{data}) = \left[\frac{1}{\sigma\sqrt{2\pi}}\right]^n \times \exp\left[-\frac{\epsilon_1^2}{2\sigma^2}\right] \times \exp\left[-\frac{\epsilon_2^2}{2\sigma^2}\right] \times \ldots \times \exp\left[-\frac{\epsilon_n^2}{2\sigma^2}\right]
$$

Now we will take the natural logarithm of both sides of the expression

$$
\ln\Big(\mathcal{L}(\beta_0, \beta_1 | \text{data})\Big) = \ln\left(\left[\frac{1}{\sigma\sqrt{2\pi}}\right]^n \times \exp\left[-\frac{\epsilon_1^2}{2\sigma^2}\right] \times \exp\left[-\frac{\epsilon_2^2}{2\sigma^2}\right] \times \ldots \times \exp\left[-\frac{\epsilon_n^2}{2\sigma^2}\right]\right)
$$

Using our rules for logarithms and re-arranging gives,

$$
\updownarrow(\beta_0, \beta_1 | \text{data}) = -\frac{n}{2} \times \ln(2\pi\sigma^2) - \frac{1}{2\sigma^2} \times \sum \epsilon_i^2
$$

The log-liklihood is a function of n, σ^2 and the SSE. The data define n (the sample size) and the other two come from the error terms which are a function of the parameters and the data.

Once we have this function, calculus can be used to find the analytic maximum. Typically before we do this, we replace ϵ_i with $Y_i - \beta_0 - \beta_1(X_i)$. Then the partial derivatives w.r.t. β_0 and β_1 are both compued, set equal to zero, and solved for the respective parameter.