Logarithms: a gentle introduction

2025-12-08

Table of contents

1 Packages used

```r
library(tibble)
library(ggplot2)
library(moderndive)
library(kableExtra)
```

2 What are logarithms?

A logarithm is simply a way of asking: "To what power must I raise one number to get another number?"

For example, if we ask "what is the logarithm of 100 in base 10?", we're really asking: "10 raised to what power gives us 100?" The answer is 2, because $10^2 = 100$.

We write this as:

$$\log_{10}(100) = 2$$

We say this out loud as: "log base 10 of 100 equals 2"

2.1 Understanding the base of a logarithm

The **base** of a logarithm is the number we're repeatedly multiplying. In $\log_{10}(100) = 2$, the base is 10. This tells us "10 multiplied by itself 2 times equals 100."

Different bases are useful in different contexts:

- **Base 10** ($\log_{10}$ or just "log"): Common in sciences, easy to interpret because we use base-10 number system
- **Base 2** ($\log_2$): Used in computer science and information theory as computers work with binary numbers (0 and 1) and, thus, the base-2 number system
- **Base e** (ln or natural log): Most common in statistics, economics and business studies (see below)

  Convention: When someone writes $\log(x)$ without specifying a base, they usually mean the natural logarithm (base e) in statistics, although ln would be more correct. This is particular confusing as in many other fields it might also mean the base 10.

Computing logs in R is pretty straightforward:

```
log10(100)  # Base 10: "log base 10 of 100"
log2(8)     # Base 2: "log base 2 of 8" (2^3 = 8, so answer is 3)
log(100)    # Natural log (base e): "natural log of 100" or "ln of 100"
```

From the definition of the log the following special cases also follow naturally:

```
log(0)
```

```
[1] -Inf
```

and

```
log(-1) # negative logs are not defined.
```

```
[1] NaN
```

> **ℹ Business Example: Understanding Bases**
>
> Imagine you're analyzing company growth rates:
>
> - If your company **doubles** every year (2x growth), base-2 logarithms are useful: $\log_2(16) = 4$ means "we doubled 4 times to reach 16x our starting size"
> - If your company grows by **10x** at each stage (startup → local → regional → national → international), base-10 logarithms make sense sense: $\log_{10}(1000) = 3$ then means that we grow by $10x$ three times to reach $1000x$ of our starting size.

> • If your company grows at a **continuous rate** (like 15% per year compounded continuously), natural logarithms (base e) are most appropriate; this is why they dominate finance and economics.

The most common logarithm you'll encounter in statistics is the **natural logarithm** (written as "ln" or "log"), which uses a special number called e (approximately 2.718) as its base. Don't worry too much about e for now—just know that natural logarithms are the standard in statistical analysis.

```
# In R, the natural logarithm is simply log():
log(100)
```

```
[1] 4.60517
```

```
# The number e in R:
exp(1)
```

```
[1] 2.718282
```

> 💡 Business Example: Why Natural Logarithms in Finance
>
> Banks use continuous compounding, which naturally involves e. If you invest 1,000 EUR at 5% annual interest compounded continuously:
>
> • After 1 year: $1000 \times e^{(0.05 \times 1)} = 1,051.27$ EUR
> • After 10 years: $1000 \times e^{(0.05 \times 10)} = 1,648.72$ EUR
>
> Taking the natural logarithm lets you easily work backwards: if your investment grew to €2,000, you can find how long it took by calculating $\ln(2000/1000) \div 0.05$ = 13.86 years.

3 The natural logarithm and exponential functions

The natural logarithm and the exponential function e^x are mathematical inverses—they undo each other. If you apply one and then the other, you get back to where you started:

• $\ln\left(e^x\right) = x$
• $e^{\ln(x)} = x$

```
# Demonstrating that log and exp undo each other:
x <- 5
log(exp(x))  # Should give us 5
```

```
[1] 5
```

```r
exp(log(x))  # Should also give us 5
```

```
[1] 5
```

This relationship is particularly useful because many real-world processes follow exponential patterns. When something grows exponentially, it increases by a constant percentage over time (like compound interest or viral marketing reach). The exponential function captures this: if you invest 1,000 EUR at 5% annual interest, after t years you have $1000 \times e^{(0.05t)}$ EUR

By taking the natural logarithm of exponentially growing data, we transform that curved, accelerating relationship into a straight line, as shown in Figure 1. This is why this kind of transformation is particularly helpful in the context of linear regression.

```r
library(ggplot2)
library(patchwork)

# Generate exponential data
time <- 0:20
value <- 10 * exp(0.15 * time)

# Create data frame
df <- data.frame(time = time, value = value, log_value = log(value))

# Plot 1: Original exponential relationship
p1 <- ggplot(df, aes(x = time, y = value)) +
  geom_point(size = 2, color = "steelblue") +
  geom_line(color = "steelblue", linewidth = 1) +
  labs(title = "Original exponential growth",
       x = "Time",
       y = "Value") +
  theme_minimal(base_size = 12)

# Plot 2: Log-transformed relationship (now linear)
p2 <- ggplot(df, aes(x = time, y = log_value)) +
  geom_point(size = 2, color = "darkred") +
  geom_line(color = "darkred", linewidth = 1) +
  labs(title = "After log transformation (linear!)",
       x = "Time",
       y = "ln(Value)") +
  theme_minimal(base_size = 12)

# Combine plots
p1 + p2
```
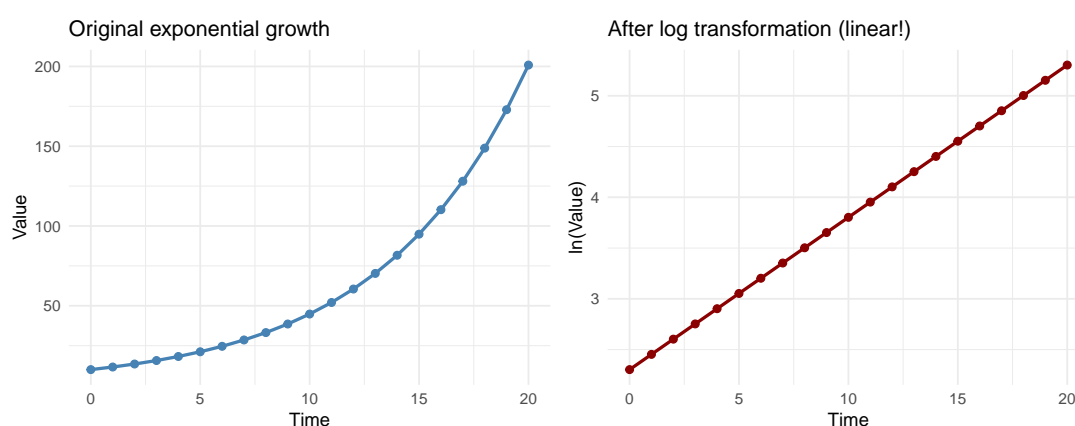
Figure 1: Transforming exponential growth into a linear relationship using the natural logarithm

4 When and why do we use logarithms?

Logarithms are useful in statistics and research for a number of reasons:

Making skewed data more symmetric. Many real-world variables—like income, property prices, or company revenues—have a few very large values and many small values. Taking the logarithm compresses those large values and spreads out the small ones, making the data easier to analyze with standard statistical methods, as illustrated in Figure 2.

```
# Generate skewed income data (lognormal distribution)
set.seed(123)
income <- rlnorm(1000, meanlog = 10, sdlog = 0.8)

df_income <- data.frame(
  income = income,
  log_income = log(income)
)

# Plot original skewed distribution
p1 <- ggplot(df_income, aes(x = income)) +
  geom_histogram(bins = 50, fill = "steelblue", color = "white") +
  labs(title = "Original income distribution (highly skewed)",
       x = "Income (€)",
       y = "Frequency") +
  theme_minimal(base_size = 12)

# Plot log-transformed distribution
p2 <- ggplot(df_income, aes(x = log_income)) +
  geom_histogram(bins = 50, fill = "darkred", color = "white") +
  labs(title = "Log-transformed income (more symmetric)",
```

```
        x = "ln(Income)",
        y = "Frequency") +
  theme_minimal(base_size = 12)

p1 + p2
```
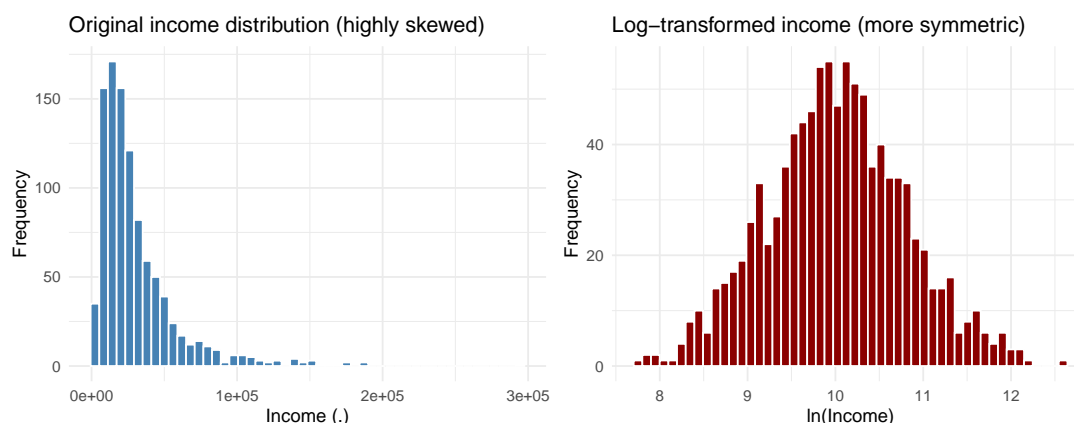

Figure 2: How logarithmic transformation makes skewed data more symmetric

Interpreting percentage changes. When we use logarithms in regression models, our results tell us about proportional or percentage changes rather than absolute changes. For instance, if you're studying how education affects income, using log(income) helps you say "an extra year of education is associated with a 10% increase in income" rather than "an extra year increases income by EUR 5,000" — which is more meaningful since a 5,000 EUR increase means very different things to someone earning 20,000 EUR versus 200,000 EUR.

---

i The Mathematics Behind Percentage Interpretation

When you use log(Y) as your dependent variable in regression, the coefficient tells you about **proportional changes** rather than absolute changes. Here's why:
If your regression model is: $\ln(Y) = \beta_0 + \beta_1 X$
Then taking the exponential of both sides:

$$Y = e^{(\beta_0 + \beta_1 X)} = e^{\beta_0} \times e^{(\beta_1 X)}$$

When X increases by 1 unit, Y gets multiplied by e^{β_1}. For small values of β_1 (say, less than 0.15), we can use the approximation:
Percentage change in $Y \approx \beta_1 \times 100$
For example, if $\beta_1 = 0.1$, then a one-unit increase in X is associated with approximately a 10% increase in Y.
Why this matters: In the education-income example, if ln(Income) = \$10.5 + 0.10 ×\$Education: - Someone with 12 years of education: predicted income = $e^(10.5 + 0.10 \times 12) = 73,700$ EUR - Someone with 13 years of education: predicted income = $e^(10.5 + 0.10 \times 13) = 81,450$ EUR - The increase is EUR 7,750, which is

---

> 10.5% of EUR 73,700
> Notice that the absolute increase (EUR 7,750) changes depending on the starting income, but the percentage increase (≈ 10) stays constant. This is much more realistic for most economic relationships.

Analyzing multiplicative relationships. Some phenomena grow or shrink by multiplication rather than addition—like compound interest, population growth, or market dynamics. Logarithms transform these multiplicative relationships into additive ones, which are much simpler to work with statistically.

ℹ Example: Multiplicative vs. Additive Relationships

Multiplicative relationship (harder to model): Suppose a product's market share grows by 20% each quarter. Starting at 5%:

- Quarter 0: 5
- Quarter 1: 5
- Quarter 2: 6
- Quarter 3: 7.2

The relationship is: Market Share(t) = 5
This is **multiplicative** - we multiply by 1.20 each period. Linear regression struggles with this curved, accelerating pattern.
After log transformation (additive and linear): Taking the natural logarithm:

$$\ln(MarketShare) = \ln(5\%) + t \times \ln(1.20)$$

Now the relationship is **additive** - we add $\ln(1.20) \approx 0.182$ each period:

- Quarter 0: ln(5
- Quarter 1: $-3.00 + 0.182 = -2.82$
- Quarter 2: $-2.82 + 0.182 = -2.64$
- Quarter 3: $-2.64 + 0.182 = -2.45$

This is now a simple straight line that standard linear regression can easily handle! The coefficient (0.182) directly tells us the quarterly growth rate.
Key insight: Multiplication in the original data becomes addition after log transformation, making complex exponential patterns manageable with basic statistical tools.

Handling exponential relationships. Consider a company analyzing how advertising spending affects product awareness. In the early stages, each 1,000 EUR spent might dramatically increase awareness, but as awareness grows, it takes progressively more spending to achieve the same absolute increase—a classic exponential pattern.

If awareness follows the relationship:

$$Awareness = 10 \times e^{(0.0005 \times Spending)}$$

trying to fit a linear regression directly would fail badly. But if we take the natural logarithm of both sides:

$$\ln\left(Awareness\right) = \ln\left(10\right) + 0.0005 \times Spending$$

we now have a linear relationship that's perfect for regression analysis. The regression coefficient (0.0005) tells us how spending affects the growth rate of awareness.

You can see this in action with a concrete example in Figure 3.

```r
# Generate advertising data following exponential relationship
set.seed(456)
spending <- seq(0, 10000, by = 500)
# True relationship: Awareness = 10 * exp(0.0005 * Spending) + noise
awareness <- 10 * exp(0.0005 * spending) * exp(rnorm(length(spending), 0, 0.1))

df_ad <- data.frame(
  spending = spending,
  awareness = awareness,
  log_awareness = log(awareness)
)

# Fit linear model on log-transformed data
model <- lm(log_awareness ~ spending, data = df_ad)

# Plot 1: Original scale with exponential curve
p1 <- ggplot(df_ad, aes(x = spending, y = awareness)) +
  geom_point(size = 2, color = "steelblue") +
  geom_line(aes(y = exp(predict(model))),
            color = "darkred", linewidth = 1, linetype = "dashed") +
  labs(title = "Original scale: Exponential relationship",
       x = "Advertising spending (€)",
       y = "Product awareness (%)") +
  theme_minimal(base_size = 12)

# Plot 2: Log scale showing linear relationship
p2 <- ggplot(df_ad, aes(x = spending, y = log_awareness)) +
  geom_point(size = 2, color = "steelblue") +
  geom_smooth(method = "lm", se = TRUE, color = "darkred", fill = "pink") +
  labs(title = "Log scale: Linear relationship",
       x = "Advertising spending (€)",
       y = "ln(Awareness)") +
  theme_minimal(base_size = 12)

p1 + p2
```
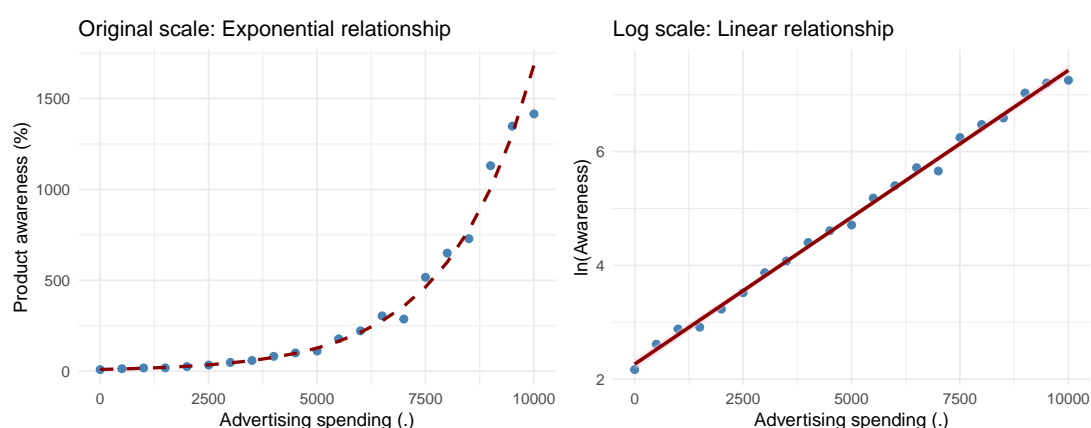
Figure 3: Linear regression on log-transformed data captures the exponential relationship between advertising spending and product awareness

```
get_regression_table(model) |>
  kable()
```

Table 1: Regression results.

term	estimate	std_error	statistic	p_value	lower_ci	upper_ci
intercept	2.264	0.044	51.017	0	2.171	2.357
spending	0.001	0.000	68.046	0	0.001	0.001

In this example, the coefficient for spending (approximately 0.0005, see Table 1) tells us that for each additional euro spent on advertising, there is an associated awareness increase by about 0.05%. This multiplicative effect is much more realistic than assuming the same absolute increase in awareness regardless of current spending levels.

5 Summary

You don't need to calculate logarithms by hand—statistical software handles this automatically. What matters is understanding when and why we use them in our analyses:

- Use logarithms when your data is highly skewed
- Use logarithms when you're interested in percentage changes rather than absolute changes
- Use logarithms when relationships are exponential or multiplicative rather than linear
- Remember that `log()` in R gives you the natural logarithm