Chapter 10

A Brief Note on Feature Engineering

All machine learning algorithms and statistical models depend on the concept of a **feature**. A feature is some aspect of a dataset that, the model designer believes, represents the data in a way that is relevant to the problem he/she is trying to solve.

Before any algorithm can be applied, therefore, it is necessary to decide how to represent the data: which features to include and how to extract them from the raw data. This task is called **feature engineering**.

10.1 Study Design vs. Feature Engineering

We have seen a large number of features in Chapters 1–9, but we never stopped to consider them. That's because, in many datasets, the features are chosen at the **study design** stage. The analyst (statistician, data scientist, etc.) has no say in what the features look like or which features are included.

This paradigm is changing as data science increasingly focuses on large, observational datasets, like those from electronic medical records (EMRs). In these types of studies, the raw data were not collected for the study itself, but to fulfill some other purpose. The analyst must choose how to build features

from the raw data and use them in models.

Question 10.1

The examples in Chapters 2 and 3 used the same two features. What were these features? How were they represented? What are some alternatives to this choice of features?

Question 10.2

In Chapter 7, we looked at the Wisconsin Breast Cancer Dataset, which includes 30 different imaging features relevant to predicting whether a tumor is benign or malignant. How were these features represented? What are some alternatives to this choice?

Question 10.3

In Chapters 8 and 9, we looked at two datasets that were collected for the purposes of answering particular questions. Do you agree with these study designers' choice of features? What other features could potentially have been relevant to answering each research question?

10.2 Turning Data into Numbers

A model is just a tool for learning relationships among sets of numbers. The first step in any data science problem, therefore, is deciding how to represent what is often a large, complex, noisy dataset as a set of numbers.

10.2.1 Numbers

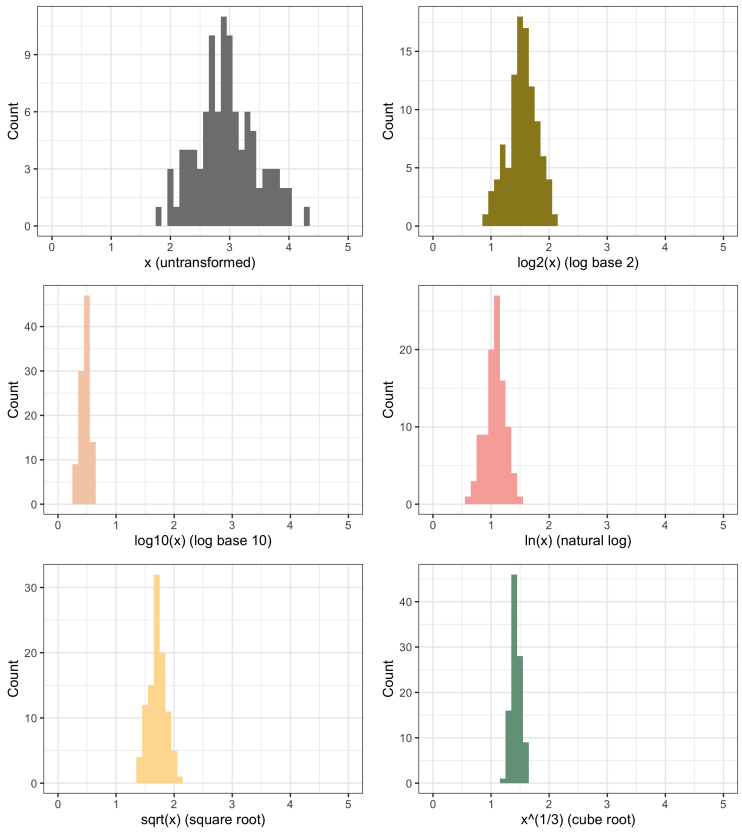
Sometimes you get lucky and the feature you need is already a number, such as a vital sign measurement, lab value, or other biomarker. In that case, more often than not, the feature enters into the model as its raw value.

In some cases, you may also choose to apply a **transformation** to the feature before it enters the model. A transformation is simply the application of a deterministic mathematical function that changes the shape of the distribution

of the feature. Transformations are often used to improve the interpretability of a model and/or to ensure that the model fulfills the assumptions of the statistical inference method(s) being used (e.g., a hypothesis test).

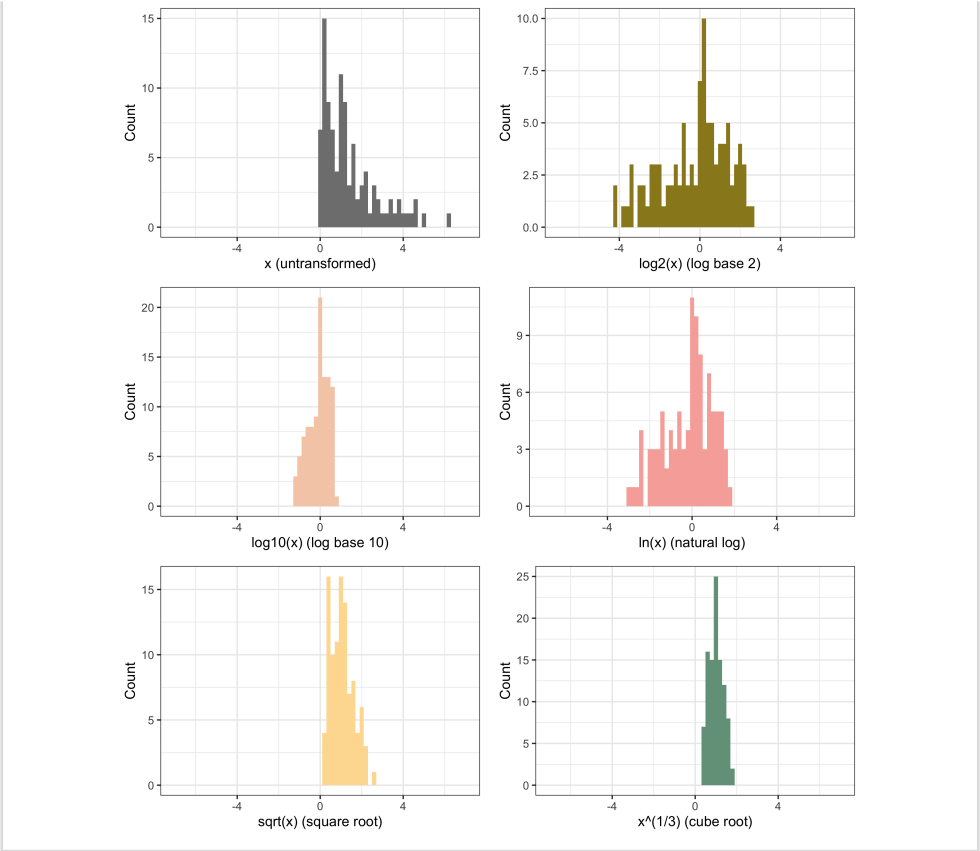
Question 10.4

Here are 100 random samples from a normal distribution with $\mu = 3.0$ and $\sigma = 0.5$ and five different transformations of those samples. What do you notice about the shape and position of the data under the different transformations?



Question 10.5

Here are 100 random samples from an exponential (see Section 4.7) distribution with $\lambda = 0.8$ and the same five transformations of those samples. What do you notice about the shape and position of the data under the different transformations?



Economics, the social sciences, and related disciplines, which are heavily dependent on the use of regression models and hypothesis tests, rely extensively on transformations. In my experience, machine learning folks spend almost no time on them because their primary concern is predictive accuracy, not model interpretation. Machine learning practitioners, however, very frequently **scale and center** their predictors (see footnote in Section 12.5), which is another type of transformation. We will get into more detail on transformations as we continue to learn about regression models.

10.2.2 Binary Variables

For features which are yes/no (e.g., presence/absence of a disease, symptom, physical attribute, etc.) the most common coding scheme is to use "1" for "yes" and "0" for "no". This is useful for interpretation, particularly in regression

models. In a linear regression model using this coding scheme, for example, the model coefficient will be the shift in the mean of the normal distribution representing the outcome, y, when the feature is present.

10.2.3 Categories

Categorical features with $k > 2$ categories are generally represented using **indicator variables**. If a feature, x, has k **levels**, we can use $k - 1$ yes/no indicator variables to represent that feature. For example, assume $k = 3$ and the possible levels of our feature, x, are A, B, and C. We set:

$$x_1 = \begin{cases} 1 & \text{if } x = A \\ 0 & \text{otherwise} \end{cases}$$

$$x_2 = \begin{cases} 1 & \text{if } x = B \\ 0 & \text{otherwise} \end{cases}$$

If the value of x is A, $x_1 = 1$ and $x_2 = 0$. If it's B, $x_1 = 0$ and $x_2 = 1$. The value C is called our **reference category** and has $x_1 = 0$ and $x_2 = 0$. In this way, information about all three categories is captured using only two variables. Creating indicator variables is just another way of transforming the value of a feature.

Question 10.6

In Section 9.3, we saw an example of a model that predicts whether or not a mother will give birth to a low birthweight baby. One of the factors considered in that model is the mother's race, which was coded (crudely and probably inaccurately, I might add) as `1 = white, 2 = Black, 3 = other`. You can tell how the feature `RACE` was coded by examining the model output. How many indicator variables were used? Which level of the feature was used as the reference category?