CHAPTER 5

Linear regression and variable transformations I

In the previous chapter, we re-introduced linear modeling. In this chapter, we continue our treatment of linear modeling, but we begin to deal with some violations of the assumptions.

The classical linear model assumes that the error terms are distributed normally. What happens when this is not true? There are three ways of handling it. First, you can ignore it. Ignoring this violation is usually not terrible when you are dealing with interpolation. However, if the predictions are important, you should not ignore this violation.

Second, we can use high-powered regression theory, namely Generalized Linear Models and Generalized Additive Models. These will be covered in a future chapter. The strength of these methods is that they simplify the analysis. GLMs and GAMs really do what we will be doing by hand in this chapter. The primary weakness of GLMs and GAMs is that they are not as flexible as the methods we are developing here.

Finally, you can transform the dependent variable into something that is more normal than before. These transformations are very flexible. Once you get used to working in two different systems of units, you can easily use these methods to 'normalize' any restricted dependent variable.

	Estimate	Std. Error	t value	Pr(>|t|)
Intercept	19.1412	6.6557	2.88	0.0076
Years after 1998	-2.0095	0.3577	-5.62	0.0000
Includes Civil Union ban	-3.7331	1.9988	-1.87	0.0723
Percent identifying as Religious	0.9452	0.1074	8.80	0.0000

Table 5.1: *Results table for the regression of percent support of a generic ballot outlawing same-sex marriage against the three included variables. The R^2 for the model is 0.7801. The probabilities calculated are two-tailed probabilities. The hypotheses were one-tailed hypotheses. As such, all three explanatory variables are statistically significant at the standard level of significance ($\alpha = 0.05$). Note the years are now measured as post-1998 (the year of the first such ballot measure).*

5.0.1 An example

As a quick example, let us continue the analysis we started in the previous chapter. However, instead of trying to predict the proportion of people who vote in favor of the ballot measure, let us give an estimate of the measure's probability of passing. Although we predicted that 42.41% of the voters will vote in favor of banning single-sex marriage, the parameter estimates are only estimates. In Ordinary Least Squares (OLS) regression, the parameter estimates are distributed normally with the given mean and with the standard deviation equal to the standard error.

Thus, using the regression table (Table 5.1), we know that the estimate of the `yearPassed` parameter, $\beta_1 \sim \mathcal{N}(\mu = -2.0095, \sigma = 0.3577)$. Thinking of the parameter estimates in this manner emphasizes two very important aspects of those estimates.

First, the parameter estimates are estimates of the population parameter using the information in the sample. Thus, it is very important to make sure that the sample is representative of the population. It also means that these estimates are not the true value. In fact, because they are distributed normally, we know there is a 5% chance that the population parameter is more than 1.96 times the standard error away from the estimate.[1] In the case of the `yearPassed` variable, we know that there is a 5% chance that the real parameter is either lower than $-2.0095 - 1.96 \times 0.3577 = -2.711$ or greater than $-2.0095 + 1.96 \times 0.3577 = -1.308$. Thus, the real effect of passing years could be 1.0000 instead of something negative. The probability of this happening would be extremely small,

[1]The value of 1.96 is, of course, an estimate. The exact value is $\Phi^{-1}(0.05/2)$. This value arises from the fact that the parameter estimates are normally distributed.

but it is possible.[2] Keeping in mind that these parameters are just estimates helps keep us *humble*.

Second, we can leverage the fact that these are estimates to answer some different hypotheses and to predict different events. For instance, we know the predicted vote outcome for the Maine election. But, given that the parameter estimates are estimates, can we estimate the probability of it passing? The answer is yes. All we have to do is some Monte Carlo simulations.

5.0.2 Monte Carlo simulations

The purpose of Monte Carlo simulations is to make better approximations of statistics given that the underlying assumptions are not met. Thus, we could use Monte Carlo techniques to estimate the regression equation when the errors are not normally distributed. We will not, because there exist better solutions in most cases.

The steps of a typical Monte Carlo experiment consists of defining the input distributions, defining the relationship between the inputs and the outputs, and performing the experiment many times. To see how this is done, let us use Monte Carlo to predict the probability that the ballot measure will pass in Maine.

Step 1: Define the distributions

As we know that the parameter estimates are normally distributed, we have the following four input distributions:

1. $\beta_0 \sim \mathcal{N}(\mu = 19.1412, \sigma = 6.6557)$
2. $\beta_1 \sim \mathcal{N}(\mu = -2.0095, \sigma = 0.3577)$
3. $\beta_2 \sim \mathcal{N}(\mu = -3.7331, \sigma = 1.9988)$
4. $\beta_3 \sim \mathcal{N}(\mu = 0.9452, \sigma = 0.1074)$

[2] To calculate the probability that this could happen, you would use the z-score, and the normal cdf:

$$\Phi\left(\frac{-2.0095 - 1.0000}{0.3577}\right) \approx 1.70 \times 10^{-16}$$

See if you can determine where all of the numbers came from. The phi function ($\Phi()$ is the normal cumulative distribution function. It changes an x-value into a probability when that x-value represents a standardized value that is distributed normally, as in this example. In `R`, it is `dnorm()`; in Excel, `=INVNORM()`; in STATA, `norm()`; in SAS, `PROBNORM()`; and in SPSS, `CDF.NORM()`.

Step 2: Define the linking equation

The relationship between the inputs and the prediction is simply the generic model equation:

$$pctWin = \beta_0 + \beta_1 \times (yearPassed - 1998) + \beta_2 \times (civilBan) + \beta_3 \times (religPct) \tag{5.1}$$

For Maine, we know the `yearPassed` = 2009, `civilBan` = 0, and `religPct` = 48. With this information and the model equation, we know the formula to predict the vote share is

$$pctWin = \beta_0 + \beta_1 \times 11 + \beta_2 \times 0 + \beta_3 \times 48 \tag{5.2}$$

Step 3: Repeat many times

All that remains is creating several hundred thousand (or more, if your computer can handle it) draws from each of the four distribution and calculating the resulting predictions. This can even be done in Excel (the first four columns are the four distributions, and the fifth column is the prediction). In `R`, the code to accomplish this is as follows. Actually, this `R` code does the predictions, plots a histogram (Figure 5.1) of the predictions, and finds the proportion of the Monte Carlo 'elections' that produced a win for the ballot measure.

```
### Set model values
b <- 19.1412
y <- -2.0095
c <- -3.7331
r <-  0.9452

b.se <- 6.6557
y.se <- 0.3577
c.se <- 1.9988
r.se <- 0.1074

### Set values for Maine
year.me   <- 11
civil.me  <- 0
relig.me  <- 48

### Set number of trials
n <- 1000000

### Perform repeated draws
const <- rnorm(n, mean=b, sd=b.se)
year  <- rnorm(n, mean=y, sd=y.se)
civil <- rnorm(n, mean=c, sd=c.se)
relig <- rnorm(n, mean=r, sd=r.se)
```

```
### Calculate the prediction for each draw
pred.me <- year*year.me + relig*relig.me + civil*civil.me + const

### Create a histogram of the results
png("hist-model1MC.png", width=600, height=300)
hist(pred.me, breaks=151, main="", yaxt="n",
    xlab="Predicted percentage of the vote to ban gay marriage", ylab="",
    xlim=c(0,100),
    las=1)
abline(v=50, col="red")
dev.off()

### Calculate the proportion of wins
length(which(pred.me>50))/n
```

To actually use the results, we can plot a histogram (Figure 5.1), or we can count the proportion of Monte Carlo 'elections' that won the election. In Figure 5.1, all one million Monte Carlo 'elections' are displayed. Those to the right of the vertical line count as wins. Thus, we see that, under the assumptions of the model, the ballot measure has about a 20.7% chance of passing. If we want to calculate odds, we use the definition of odds: ※

$$\frac{\mathbb{P}\left[win\right]}{\mathbb{P}\left[loss\right]}$$

Thus, the odds of the ballot measure passing is approximately 3.8 to 1 against.

⋆ ⋆ ⋆

If we look carefully, we notice something interesting, and frightening: the model will sometimes predict election outcomes that do *not* make physical sense. In fact, it predicted an election to have a negative vote share. What happened? Did we do something wrong? Is there something more subtle at work here?

5.1 Restricted dependent variables

Quite often, you will come across dependent variables that simply cannot take on every real value. Examples include dichotomous, ordinal, and proportion variables. Using OLS when your dependent variable is restricted is a violation of the normality assumption. It also may be a violation of common sense: your model may predict values outside the range of possible values (as did our last example).

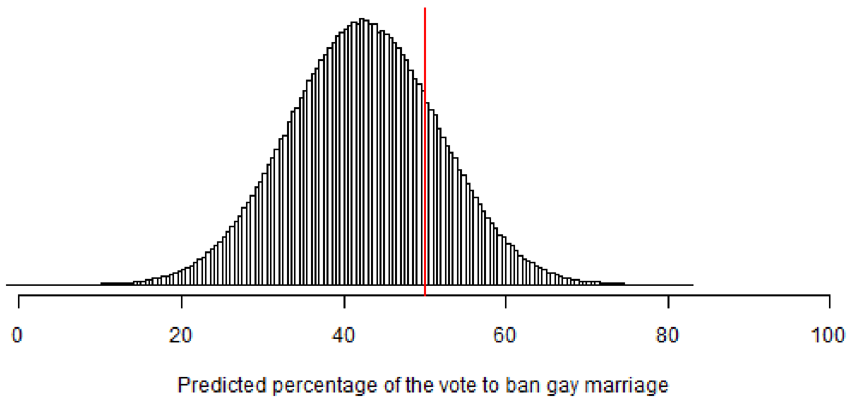

Figure 5.1: *A histogram of the outcomes of the one million Monte Carlo 'elections.' Those predictions to the right of the vertical line indicate a win; to the left, a lose. The total proportion to the right is 0.206651.*

As an example of this, let us return to the model in the previous chapter. Table 5.1, reproduced here, is the result of regressing the percent of vote in the state in favor of banning single-sex marriage. In the last chapter, we tested the assumptions of OLS and decided that the model does not violate those assumptions. We forgot, however, to check if the dependent variable can take on all values. This is a consequence of the assumption that $e_i \overset{\text{iid}}{\sim} \mathcal{N}(0, \sigma^2)$. As the errors are allowed to take on all real values, so must the dependent variable. However, the vote share variable is bounded below by zero and above by 100. As such, the residuals cannot be truly distributed normal.

In the model, let us predict the vote share for the year 1990 on a ballot measure that does not ban civil unions, in a state with a religiosity level of 85. All of these values are legitimate values; the religiosity level in Mississippi is 85%. Substituting the values into our regression equation, we get

$$pctWin = 4034.16 - 2.01(1990) - 3.73(0) + 0.95(85) \tag{5.3}$$

Using our calculator, we find the predicted vote share in this scenario is 115.01. That is, this model predicts that the vote in favor of banning single-sex marriage is 115.01% of those who voted. This prediction makes no sense (unless we are in Chicago). So, what do we do?

Y	(level units)
$\downarrow$	transform with $\tilde{y} = f(y)$
$\tilde{Y}$	(transformed units)
$\downarrow$	un-transform with $y = f^{-1}(\tilde{y})$
Y	(level units)

Table 5.2: *Schematic of the variable transformation procedure.*

5.1.1 Variable transformations

One solution, and perhaps the most obvious, is to change the dependent variable so that all values make sense. This is done through a process of variable transformation. There are three steps: First, transform the variable from a restricted range to an unrestricted range. Second, perform the linear regression on this transformed variable. Finally, un-transform the variable (and results) into the original range.

The overview of this plan is shown in Table 5.2. The key is the transformation. It must change the range of Y from its current limits to an unlimited version, denoted $\tilde{Y}$. Luckily, there are a few popular transformations, which I will cover in the next sections.

These are just the frequently used transformations. If your particular restriction is not listed, you can create your own transformation. The only two requirements are that it works and that it, and its inverse, are functions.

5.2 Proportions

Let us assume that your dependent variable is a proportion; that is, $0 < Y < 1$. The function that transforms this range into $\mathbb{R}$ is called the logit function:

$$\tilde{Y} = f(y) = \log\left(\frac{y}{1-y}\right) \tag{5.4}$$

Its inverse, which transforms it from logit units into level units is called the logistic function:

$$Y = f^{-1}(\tilde{y}) = \frac{1}{1+\exp(-\tilde{y})} = \frac{\exp(\tilde{y})}{1+\exp(\tilde{y})} \tag{5.5}$$

A careful reader will note that the range of Y includes neither 0 or 1. This is because there is no way of transforming a (semi-) closed interval into an open interval such as $\mathbb{R}$, while ensuring that the inverse is also a function. This is a provable fact of mathematics.

But, what do we do if there are y-values that are zero (one)? One solution is to add (subtract) an extremely small number, ϵ, to the zero (one). A second solution is to completely drop those data from the analysis. A third solution is to change the proportion into a count (which is covered later). None of these solutions is perfect. The best answer is to do all three and see how much ※ your answer changes. A general rule of thumb is that, if your underlying map is correct, the results should not vary wildly based on similar models. That is, if we know Y depends on X_1 and X_2, then all closely appropriate modeling techniques should give approximately the same answers. If they do not, then there is something seriously wrong with your assumptions about the underlying relationships.

Statistical programs will add a very small value to zeroes and subtract a very small value from ones to avoid the issues discussed here.

Example

What if our data is a percent, $0 < Y < 100$? The first step is to transform it into a proportion by dividing by 100. Then, use the above transformations to extend the range. Just remember to multiply by 100 at the end.

With this in mind, let us revisit the `ssm1.csv` data and the model from last chapter and from Section 5.0.1. Recall that the `ssm1` data consists of five variables: nominal `stateId`, continuous `yearPassed`, percent `pctWin` (percent vote in favor of the measure), dichotomous `civilBan` (whether the measure also contained a ban on civil unions), and percent `religPct` (the percent of people in the state agreeing that religion is an important part of their lives).

Our basic model is that $pctWin \sim yearPassed + civilBan + religPct$. The model we ran in last chapter was a basic OLS linear model. Our regression equation ended up being $pctWin = 4034.16 - 2.01(yearPassed) - 3.73(civilBan) + 0.95(religPct)$. This is a straight-forward linear model. An increase of one year reduces the expected vote share by 2.01 percentage points. As such, it is easily interpretable. However, in Sections 5.0.1 and 5.1, we discovered that the model equation produced impossible predictions. This was due to the limited aspect of the dependent variable. Thus, let us transform it into an unbounded variable using the above transformation.

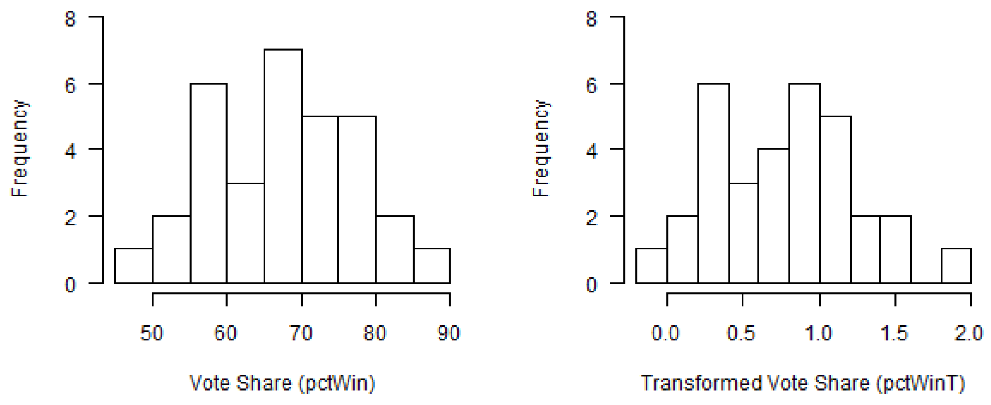

Figure 5.2: *Histograms of the original pctWin variable and of the transformed variable.*

Thus, let us define a new variable, $pctWinT$, as the transformed variable. Then,

$$pctWinT := \log\left[\frac{pctWin/100}{1 - pctWin/100}\right]$$

We can check the range of the new variable, and compare it to the histogram of the un-transformed variable (see Figure 5.2). However, there is not much you can tell by looking at the histograms.

The next step is to perform the OLS regression on this transformed variable. Thus, we will get a regression equation of the form $pctWinT = \beta_0 + yearPassed\beta_1 + civilBan\beta_2 + religPct\beta_3$. Note that this regression equation is just as easily interpretable as the original equation. The only difference ※ is that you are *no longer* predicting the vote share, you are predicting the *logit* of the vote share. Thus, to transform these predictions into vote share predictions you must take the logistic of the prediction.[3]

The results of the OLS regression show that the new model is a *better* fit to the data than was the old model. Note the smaller p-values in all of the variables of interest. Also note the increased R^2 value. Both of these facts suggest that this transformed model is superior to the original.

I will skip the testing of the assumptions for this model and leave it as an exercise for you. The

[3]Recall from above (Eqns 5.4 and 5.5) that the logistic function is the inverse of the logit function.

	Estimate	Std. Error	t value	Pr(>|t|)
Intercept	-1.7138	0.29250	5.57	0.0000
Years after 1998	-0.0886	0.0157	-5.63	0.0000
Includes a Civil Union ban	-0.2318	0.0878	-2.64	0.0134
Percent identifying as Religious	0.0475	0.0047	10.06	0.0000

Table 5.3: *Results table for the regression of the transformed percent support of a generic ballot outlawing same-sex marriage against the three included variables. The R^2 for the model is 0.8119. The probabilities calculated are two-tailed probabilities. The hypotheses were one-tailed hypotheses. As such, all three explanatory variables are statistically significant at the standard level of significance ($\alpha = 0.05$). Also note that I transformed the* `yearPassed` *variable (by subtracting 1998) to make it not so overbearing on the remaining variables. This only affected the constant term.*

steps and tests are the same as discussed in the previous chapter.

The regression equation from this model is

$$pctWinT = -1.7138 - 0.089(yearPassed - 1998) - 0.232(civilBan) + 0.048(religPct) \tag{5.6}$$

Using our data about Maine and Eqn 5.6, we get an expected logit of vote share of -0.409. To determine what that value is in level units, we take the logistic of it:

$$pctWin = 100 \times \frac{1}{1 + \exp(0.409)}$$

With this, the model predicts the outcome of the vote is that 39.9% of the people vote in favor of the ballot measure.

Simulation

Now, let us continue this line of thought by determining the probability that the ballot measure still passes. We will use the simulation methods from Section 5.0.1.

Recall that these parameter estimates are *estimates*. Thus, there is uncertainty in their true value (population value). According to this sample, we calculated the best estimate, but this is a sample, and we want to draw conclusions on the population. As this is OLS regression, we know that the parameter estimates are normally distributed with the give mean and with standard deviation equal to the calculated standard error. Thus, the effect of the `yearPassed` variable, $\beta_1 \sim \mathcal{N}(\mu = -0.08857, \sigma = 0.01572)$. First, we draw a Monte Carlo sample from this distribution

Column	Excel Formula
A	=NORMINV(RAND(),-1.713775,0.29250)
B	=NORMINV(RAND(),-0.08857,0.01572)
C	=NORMINV(RAND(),-0.23177,0.08784)
D	=NORMINV(RAND(),0.04748,0.00472)
E	=A1 + A2*11 + A3*0 + A4*48
F	=1/(1 + exp(-E1))

Table 5.4: *Excel formulae to determine the probability that the ballot measure passes. Recall that the values for Maine are `yearPassed` = 2009, `civilBan` = 0, and `religPct` = 48.*

(and from the distributions of all of the other variables). Second, we use them to calculate the expected vote share. Third, we count the proportion of those predictions above 0.500. This will give us the probability that the ballot measure passes.

We can actually do this in Excel (see Table 5.4). If the first four columns represent the parameter estimates for the constant and for the three variables, then the fifth column will be the predicted logit vote share. We can then make Column F the predicted vote share. It is then just a matter of calculating the proportion of Column F which is above 0.500.

The first step is to type all of this into the first row in Excel. Then, copy and paste the row down to 100,000 (or more). Then calculate how many of Column F are greater than 0.500. Finally, divide this number by the 100,000. This will give you the probability that the ballot passes.

Other statistical packages have other ways of doing this. In `R`, the procedure is a straightforward application of the `rnorm()` function. I divided the script into three sections. The first section defined the variables using the results from the regression. The second section performed the prediction and transformed it to level units. The third section plotted the histogram and calculated the proportion of predictions over 0.500. It is very similar to `R` listing in Section 5.0.2. The differences are due to the different model estimations and to the need to back-transform the predictions from logit units to level units.

```
### Set model values
b <- -1.713775
y <- -0.08856685
c <- -0.2317658
r <-  0.04747807

b.se <- 0.29250
y.se <- 0.01572
c.se <- 0.08784
r.se <- 0.00472
```

```
### Set values for Maine
year.me  <- 11
civil.me <-  0
relig.me <- 48

### Set number of trials
n <- 1000000

### Perform repeated draws
const <- rnorm(n, mean=b, sd=b.se)
year  <- rnorm(n, mean=y, sd=y.se)
civil <- rnorm(n, mean=c, sd=c.se)
relig <- rnorm(n, mean=r, sd=r.se)

### Calculate the prediction for each draw
pred.me <- year*year.me + relig*relig.me + civil*civil.me + const

### Back-transform into level units using the logistic function
pred.me <- 100*(1 + exp(-pred.me))^(-1)

### Create a histogram of the results
png("hist-model2MC.png", width=600, height=300)
hist(pred.me, breaks=151, main="", yaxt="n",
    xlab="Predicted percentage of the vote to ban gay marriage", ylab="",
    xlim=c(0,100), las=1)
abline(v=50, col="red")
dev.off()

### Calculate the proportion of wins
length(which(pred.me>50))/n
```

Figure 5.3 is a histogram of the predicted outcomes. The vertical line is at $pctWin = 0.500$. Thus, those to the right of the line are elections where the ballot passed, and those to the left are elections where the ballot did not pass.

Also note the shape of the histogram. The predictions are *no longer* normally distributed, as it was previously. It is now skew-right due to the logistic transformation. Before the logistic transformation, the predictions *were* normally distributed, as expected. However, the transformation modified the shape.

Since the predictions are no longer normally distributed, all of the statistics based on normality no longer apply to the results. This means that confidence intervals need to be determined based on Monte Carlo methods. I will leave this to a future chapter.

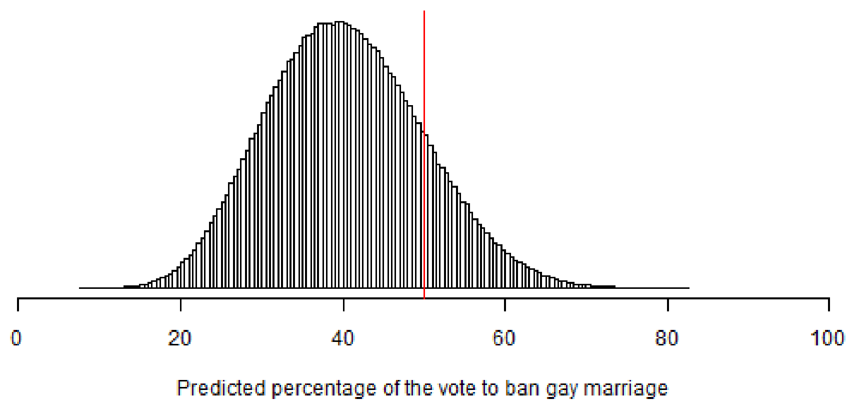

Figure 5.3: *A histogram of the Monte Carlo results. Note even with the expected vote being 39.9%, there is still a 16% chance of this ballot measure passing.*

5.3 Conclusion

In this chapter, we focused on two items: Monte Carlo and transformations. The first we can consider a statistical experiment repeated many, many, many times. It can be used to answer types of hypotheses and to estimate statistics when the underlying distribution is not standard (such as the standard error). The Monte Carlo procedure is essentially three steps: Set up the distributions, connect the distributions with the statistic, and repeat many times. We will see Monte Carlo again in the upcoming chapters in much the same way it was used in this chapter. We will also see it in the guise of bootstrapping (both parametric and non-parametric).

In addition to Monte Carlo, we discussed the need to transform your dependent variable and how to do it in the case of when the dependent variable is a proportion (or a percent). In the following chapters, we will see other transformation examples. For a listing of applicable transformations, see Appendix xxxSomewherexxx.

5.4 Extension

Here are a few things you can do to practice (and extend) what you read in this chapter.

1. Determine if the assumptions of OLS are met in Model 2 (Section 5.2).
2. The actual vote share for Maine was 52.8%. Explain why Models 1 and 2 both failed so much in predicting the vote outcome. What can be done to improve the predictions?

5.5 `R` commands

In this chapter, we saw the following new `R` commands:

dnorm(x) This function is the cumulative distribution function (cdf) for the normal distribution. It returns a probability that a normally-distributed variable will be less than or equal to `x`. This function has two additional parameters that remove the requirement that `x` has undergone the z-transformation.

rnorm(n, m, s) This function returns `n` draws from a normal distribution centered at `m` and with a standard deviation `s`. This function is the cornerstone of Monte Carlo analysis.

abline() This is an extremely handy line-generating function. It paints a line on the current plot (or returns an error if no plot exists). The line may be horizontal (`h=`), vertical (`v=`), affine (`a=, b=`), or the results of a linear model (`model`).

log(x, b) This returns the log of `x`, with a base of `b`. If you omit the `b`, this function returns the natural logarithm of `x`. To calculate the common logarithm, set `b=10`.

exp(x) This function returns the exponential of the argument, `x`; that is, it returns e^x.