Chapter 4

Prediction

Prophecy is a good line of business, but it is full of risks.
—Mark Twain, *Following the Equator*

In this chapter, we discuss prediction. Prediction is another important goal of data analysis in quantitative social science research. Our first example concerns the prediction of election outcomes using public opinion polls. We also show how to predict outcomes of interest using a linear regression model, which is one of the most basic statistical models. While many social scientists see causal inference as the ultimate goal of scholarly inquiry, prediction is often the first step towards understanding complex causal relationships that underlie human behavior. Indeed, valid causal inference requires the accurate prediction of counterfactual outcomes. Later in the chapter we discuss the connections between prediction and causal inference.

4.1 Predicting Election Outcomes

The 2008 US presidential election was historic. For the first time in American history, an African-American candidate, Barack Obama, was elected. This election was also important for the statistics community because a number of pundits accurately predicted the election outcome.

The United States's unique *Electoral College* system makes predicting election outcomes challenging. A candidate is elected to office by winning an absolute majority of electoral votes. Each of the 538 electors casts a single electoral vote. As of 2016, 535 of these votes are allocated among 50 states, corresponding to the 435 members of the House of Representatives and the 100 members of the Senate. The remaining 3 votes are given to the District of Columbia. In most cases, the electors vote for the candidate who won the plurality of votes in the state they represent, leading to a "winner-take-all" system in these states. In fact, some states have criminal penalties for voting for the candidate who did not win the plurality of votes. A winning presidential candidate must obtain at least 270 electoral votes.

Figure 4.1 shows the map of Electoral College votes for the 2008 election. See page C2 for the full-color version. Obama won 365 electoral votes (blue states), whereas

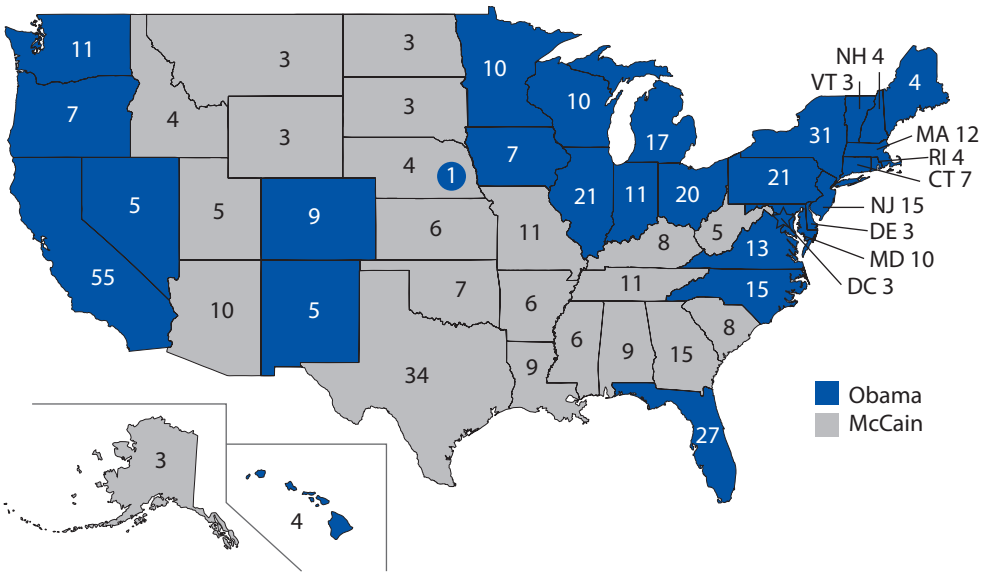


Figure 4.1. Electoral College Map of the 2008 US Presidential Election. The figure uses gray rather than red for the states won by McCain. See page C2 for the full-color version.

the Republican candidate John McCain received 173 votes (red states).[1] The Electoral College system implies that to successfully forecast the outcome of the US presidential election, we may need to accurately predict the winner of each state. Indeed, George W. Bush won the 2000 election by taking 25 electoral votes from Florida, where he defeated Al Gore by a slim margin of 537 votes after a controversial recount. As a result, Gore lost the election by the narrow margin of 5 electoral votes, even though he actually received a half million more popular votes than Bush at the national level. More recently, Donald Trump won the 2016 election even though Hillary Clinton received more votes nationally than Trump. Below, we show how to predict the election outcome using public opinion polls conducted within each state. Before we present the details of how this is done, we introduce two new programming concepts: loops and conditional statements.

4.1.1 LOOPS IN R

In many situations, we want to repeat the same operations multiple times where only small changes occur to the operations each time. For example, in order to forecast the result of the US presidential election, we must predict the election outcome within each state. This means that a similar set of computations will be performed a number of times. We would like to avoid writing nearly identical code chunks over and over again. A *loop* is a programming construct that allows us to repeatedly execute similar code chunks in a compact manner. The R syntax `for (i in X)` will create a loop, where `i` (or any other object name of your choice) is a *loop counter* that controls the

[1] Interestingly, Nebraska allocates two of its five electoral votes to the statewide winner while giving one electoral vote to the winner of each congressional district (Maine follows the same system). As a result, although McCain won a plurality of the popular vote in Nebraska, Obama received one electoral vote because he won the majority of votes in the second congressional district.

iterations of the loop, and X is the vector of values that the loop counter will successively take on. Consider the following pseudo-code.

```
for (i in X) {
    expression1
    expression2
    ...
    expressionN
}
```

Here, the collection of expressions from `expression1` through `expressionN` is repeated for each value `i` of the vector `X`. During each of these *iterations*, `i` takes on the corresponding value from the vector `X`, starting with the first element of `X` and ending with its last. Below is a simple example, which multiplies each number in a vector by 2. It is often useful to create an empty "container" vector whose elements are all `NA`s in order to store the results from computing all iterations. We use the `rep()` function to do this. Comments can be written into the loop as with any other code chunk in R. Braces `{` and `}` are used to denote the beginning and end of the body of the loop. When we start a loop (or related functions) in the RStudio text editor, the spacing will automatically indent and the closing bracket will align vertically with the `for` function. This makes the code easier to interpret and debug (i.e., identify and remove errors from the code).

```
values <- c(2, 4, 6)
n <- length(values) # number of elements in "values"
results <- rep(NA, n) # empty container vector for storing the results
## loop counter "i" will take values 1, 2, ..., n in that order
for (i in 1:n) {
    ## store the result of multiplication as the ith element of
    ## "results" vector
    results[i] <- values[i] * 2
    cat(values[i], "times 2 is equal to", results[i], "\n")
}
## 2 times 2 is equal to 4
## 4 times 2 is equal to 8
## 6 times 2 is equal to 12

results
## [1]  4  8 12
```

In each iteration of the above loop, the loop counter `i` takes an integer value, starting with `1` and ending with `n` with an increment of 1. Note that the `cat()` function, like `print()`, prints out an object on the screen. The `cat()` function combines multiple objects (character or other) into a character string, as inputs separated by commas. Without either the `cat()` or `print()` function, a loop will not print out

the `results[i]` value on the screen. Finally, recall that `\n` indicates the addition of a new line. Of course, in the above example, the loop is not strictly necessary because one can simply execute `values * 2`, which multiplies each element of the `values` vector by 2. Indeed, while loops may be conceptually easier, they are computationally intensive and so should be avoided whenever possible.

One important process is debugging code that involves a loop. Several strategies can reveal why a loop is not running properly. Since a loop simply executes the same command chunks many times, one could check whether the commands that go inside the loop can be executed without any error given a specific value of the loop counter. In the above example, one may simply try the following command before constructing the loop.

```
## check if the code runs when i = 1
i <- 1
x <- values[i] * 2
cat(values[i], "times 2 is equal to", x, "\n")

## 2 times 2 is equal to 4
```

Then, to make sure it behaves as we expect, we can change the first line to `i <- 2` or any other value that we want the loop counter `i` to take on. Another useful tip is to use the `print()` or `cat()` functions to print out the current value of the loop counter. This way, when there is an error, you always know how much of the loop succeeded. For example, if you cannot even run one iteration, there is likely something wrong with the code in the body of the loop. Alternatively, if the loop works for several iterations and then fails, perhaps something specific about the iteration that failed is causing the problem. The following example prints the iteration number to help identify the coding error. We use the `data.frame()` function to create an artificial data set with three variables, one of which is a character variable, and then attempt to compute the median of each variable using a loop.

```
## a toy data frame
data <- data.frame("a" = 1:2, "b" = c("hi", "hey"), "c" = 3:4)
## we see an error occurring at iteration 2
results <- rep(NA, 3)
for (i in 1:3) {
    cat("iteration", i, "\n")
    results[i] <- median(data[, i])
}

## iteration 1
## iteration 2

## Error in median.default(data[, i]): need numeric data

results

## [1]  1 NA NA
```

The loop was successfully executed in the first iteration but failed in the second iteration. This can be seen from the fact that an error message was printed before the printout of the `iteration 3` message. The reason for the failure is that the `median()` function takes numeric data only. As a result, the function produced an error in the second iteration, making the loop halt without computing the median for the second and third variables. This is indicated by `NA`s in the second and third elements of the `results` vector.

4.1.2 GENERAL CONDITIONAL STATEMENTS IN R

In section 2.2.4, we introduced simple conditional statements. We used the `ifelse()` function to create a vector of values where the elements of the resulting vector depend on an input object of the logical class. The general syntax is `ifelse(X, Y, Z)`. If an element `X` in the input is evaluated as `TRUE`, the value `Y` would be returned. If `X` is evaluated as `FALSE`, then the other value, `Z`, would be returned. This function is useful when recoding variables. Now, we will consider a more powerful form of conditional statements that can implement (or not) arbitrary chunks of R code depending on a logical expression. These take the form of `if(){}` and `if(){}else{}`. The first basic syntax is as follows.

```
if (X) {
    expression1
    expression2
    ...
    expressionN
}
```

If the value of `X` is `TRUE`, the code chunk `expression1` through `expressionN` will be executed. If the value of `X` is `FALSE`, then it will skip that code chunk entirely. The following simple example illustrates this.

```
## define the operation to be executed
operation <- "add"
if (operation == "add") {
    cat("I will perform addition 4 + 4\n")
    4 + 4
}

## I will perform addition 4 + 4
## [1] 8

if (operation == "multiply") {
    cat("I will perform multiplication 4 * 4\n")
    4 * 4
}
```

In the above code, the second portion of code on multiplication was not executed because the `operation` object was set to `"add"` rather than `"multiply"`. Thus, the expression `operation == "multiply"` returned a logical value of `FALSE`, indicating that the code chunk contained in the brackets is not performed. However, if `operation` is set to `"multiply"`, then `4 * 4`, rather than `4 + 4`, will be evaluated.

The `if(){}else{}` statements allow for greater flexibility by incorporating a set of R expressions to be evaluated if the argument in the `if()` function is `FALSE`. They contrast with the `if(){}` statements, which specify only the expressions to be evaluated when the argument in the `if()` function is `TRUE`. The following code will execute the code chunk `expression1a` through `expressionNa` if `X` is `TRUE` and the code chunk `expression1b` through `expressionNb` if `X` is `FALSE`.

```
if (X) {
    expression1a
    ...
    expressionNa
} else {
    expression1b
    ...
    expressionNb
}
```

Building on the earlier example, the following code illustrates how `if(){}else{}` statements work, implementing a different operation depending on the value of an object. Specifically, if the `operation` object is set to `"add"`, then the addition is performed, but otherwise, the multiplication is executed.

```
## note that "operation" is redefined
operation <- "multiply"
if (operation == "add") {
    cat("I will perform addition 4 + 4")
    4 + 4
} else {
    cat("I will perform multiplication 4 * 4")
    4 * 4
}

## I will perform multiplication 4 * 4
## [1] 16
```

One can construct even more complicated conditional statements using the `else if(){}` statement in the following manner.

```
if (X) {
    expression1a
    ...
    expressionNa
} else if (Y) {
    expression1b
    ...
    expressionNb
} else {
    expression1c
    ...
    expressionNc
}
```

The above syntax will execute the code chunk `expression1a` through `expressionNa` if condition `X` is met. If `X` is not met, but another condition `Y` is met, then the code chunk `expression1b` through `expressionNb` will be executed. Finally, if both `X` and `Y` are not satisfied, then the code chunk `expression1c` through `expressionNc` will be executed. Note that `else if()` can be repeated many times. In addition, the order of expressions matters. For example, if condition `Y` rather than `X` is evaluated first, then the code may produce a different result. Using `else if(){}`, we can modify the above example as follows.

```
## note that "operation" is redefined
operation <- "subtract"
if (operation == "add") {
    cat("I will perform addition 4 + 4\n")
    4 + 4
} else if (operation == "multiply") {
    cat("I will perform multiplication 4 * 4\n")
    4 * 4
} else {
    cat("", operation, "" is invalid. Use either "add" or "multiply."\n",
        sep = "")
}
## "subtract" is invalid. Use either "add" or "multiply."
```

Note that the `sep` argument specifies how each object should be separated. In the above example, `sep = ""` means that no character separates these objects. A separator can be any character string, commonly a comma and space (`sep = ", "`) or a semicolon and space (`sep = "; "`). The default is `sep = " "`, which will insert a space between objects.

Finally, conditional statements can be used effectively within a loop. Suppose, for example, that we want to perform a different arithmetic operation depending on whether an integer is even or odd. The following code first checks whether the input integer value is even or not. If it is even, R adds it to itself. If it is odd, R multiplies it. A message summarizing this operation is printed out for each iteration. In R, the `%%` operator computes the remainder of a division. For example, `5 %% 2` will return `1`, which is the remainder for the division of 5 by 2. If dividing an input integer value by 2 returns the remainder of 0 rather than 1, we conclude that it is an even number.

```
values <- 1:5
n <-length(values)
results <- rep(NA, n)
for (i in 1:n) {
    ## x and r get overwritten in each iteration
    x <- values[i]
    r <- x %% 2  # remainder when divided by 2 to check whether even or odd
    if (r == 0) { # remainder is zero
        cat(x, "is even and I will perform addition",
            x, "+", x, "\n")
        results[i] <- x + x
    } else { # remainder is not zero
        cat(x, "is odd and I will perform multiplication",
            x, "*", x, "\n")
        results[i] <- x * x
    }
}

## 1 is odd and I will perform multiplication 1 * 1
## 2 is even and I will perform addition 2 + 2
## 3 is odd and I will perform multiplication 3 * 3
## 4 is even and I will perform addition 4 + 4
## 5 is odd and I will perform multiplication 5 * 5

results

## [1]  1  4  9  8 25
```

Here, the code indentation, which is done automatically in RStudio, is important, making it clear that conditional statements are nested within a loop. The use of appropriate indentation is essential for writing computer code that contains loops and conditional statements.

4.1.3 POLL PREDICTIONS

Given that we now know how to use loops and conditional statements, we undertake the task of predicting the outcome of the 2008 US presidential election. Our forecast is based on a number of public opinion polls conducted before the election. The CSV data file `pres08.csv` contains the election results by state. In addition, we have the

Table 4.1. 2008 US Presidential Election Data.

Variable	*Description*
`state`	abbreviated name of the state
`state.name`	unabbreviated name of the state
`Obama`	Obama's vote share (percentage)
`McCain`	McCain's vote share (percentage)
`EV`	number of Electoral College votes for the state

Table 4.2. 2008 US Presidential Election Polling Data.

Variable	*Description*
`state`	abbreviated name of the state in which the poll was conducted
`Obama`	predicted support for Obama (percentage)
`McCain`	predicted support for McCain (percentage)
`Pollster`	name of the organization conducting the poll
`middate`	middate of the period when the poll was conducted

CSV file `polls08.csv`, which contains many polls within each state leading up to the election.[2] The names and descriptions of the variables in these data sets are given in tables 4.1 and 4.2, respectively. We begin by creating a variable, called `margin`, in both data frames, which represents Obama's vote margin over McCain in percentage points.

```
## load election results, by state
pres08 <- read.csv("pres08.csv")
## load polling data
polls08 <- read.csv("polls08.csv")
## compute Obama's margin
polls08$margin <- polls08$Obama - polls08$McCain
pres08$margin <- pres08$Obama - pres08$McCain
```

For each state, we generate a poll prediction for Obama's margin of victory using only the latest polls from the state. That is, we compute the mean prediction of all polls taken in the state on the day closest to the election. Note that this day may differ among states and there may be multiple polls conducted on the same day (more accurately, the same middate). To do this, we first initialize or create an empty vector of length 51, called `poll.pred`, which will contain the poll prediction for each of the 50 states and the District of Columbia. In the loop, we subset the data so that each iteration contains only the polls from one state.

[2] The polling data were obtained from http://electoral-vote.com.

We then further subset to extract the polls that were conducted within the state on the day closest to Election Day. This last step requires the conversion of the `middate` variable into the `Date` class using the `as.Date()` function. The `Date` class is useful because it can easily compute the number of days between the two specific dates. Input to the `as.Date()` function is a character string of the form `year-month-date` or `year/month/date`.

```
x <- as.Date("2008-11-04")
y <- as.Date("2008/9/1")
x - y # number of days between 2008/9/1 and 11/4

## Time difference of 64 days
```

Using this operation, we create the variable, called `DaysToElection`, which represents the number of days to the election. We compute this as the difference in days between the middate and Election Day (November 4). Finally, we compute the mean of poll predictions and store it as the corresponding element of `poll.pred`. Note that we use the `unique()` function to extract the unique state names in the code chunk below.

```
## convert to a Date object
polls08$middate <- as.Date(polls08$middate)
## compute the number of days to Election Day
polls08$DaysToElection <- as.Date("2008-11-04") - polls08$middate
poll.pred <- rep(NA, 51) # initialize a vector place holder
## extract unique state names which the loop will iterate through
st.names <- unique(polls08$state)
## add state names as labels for easy interpretation later on
names(poll.pred) <- as.character(st.names)
## loop across 50 states plus DC
for (i in 1:51){
    ## subset the ith state
    state.data <- subset(polls08, subset = (state == st.names[i]))
    ## further subset the latest polls within the state
    latest <- subset(state.data, DaysToElection == min(DaysToElection))
    ## compute the mean of latest polls and store it
    poll.pred[i] <- mean(latest$margin)
}
```

To set up the loop, we use the `unique()` function to extract the set of unique state names. Within the loop, we first subset the data for the `i`th state and store it as `state.data`. For example, if `i` equals `1`, it is Alabama and hence `st.names[i]` yields `AL`. We then further subset the data by extracting only the polls taken on the day closest to Election Day, which is indicated by the minimum value of the

`DaysToElection` variable. Finally, the resulting data `latest` is used to compute the average of the predicted margins from the latest polls.

We investigate the accuracy of our poll prediction by subtracting it from the actual election result of each state. The difference between the actual and predicted outcome is called the *prediction error*. We compute the prediction error by comparing the actual margin of victory with the predicted margin. We then compute the mean of poll prediction errors across states. This represents the average prediction error, which we call *bias*.

```
## error of latest polls
errors <- pres08$margin - poll.pred
names(errors) <- st.names # add state names
mean(errors) # mean prediction error
```

```
## [1] 1.062092
```

The result shows that on average across all states the poll predictions are approximately *unbiased*. More precisely, the mean of poll prediction errors across states is 1.1 percentage points, representing a bias of small magnitude. The poll predictions are for some states above and for other states below the actual election results, but on average these errors appear to roughly cancel out. While the poll predictions are approximately unbiased across states, the prediction for each state may not be accurate. For some states, the poll predictions may be well above the actual margins of victory, and these positive prediction errors are offset by large negative prediction errors for other states. To investigate this possibility, we compute the *root mean square* (RMS) of prediction error (see equation (2.3) introduced in section 2.6.2) or *root-mean-squared error* (RMSE), which represents the average magnitude of prediction error.

```
sqrt(mean(errors^2))
```

```
## [1] 5.90894
```

The result indicates that the average magnitude of each poll prediction error is about 6 percentage points.

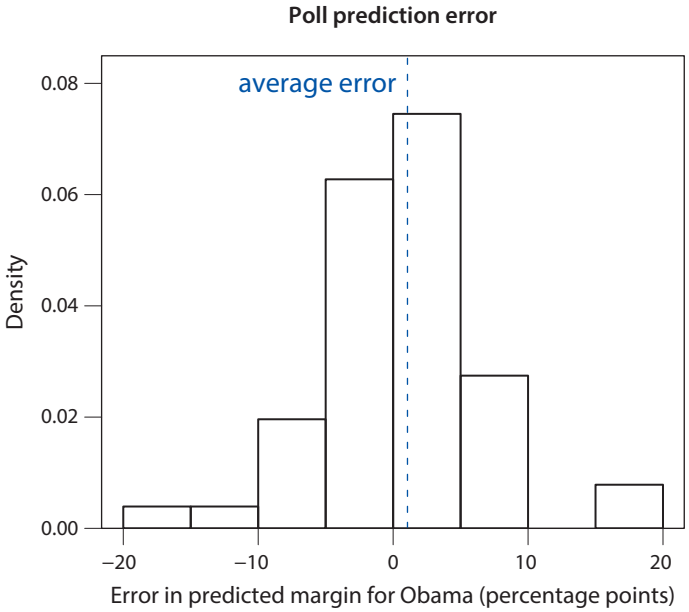
The **prediction error** is defined as

$$\text{prediction error} = \text{actual outcome} - \text{predicted outcome}.$$

The average prediction error is called **bias**, and prediction is said to be unbiased when its bias is zero. Finally, the root mean square of prediction error is called **root-mean-squared error**, representing the average magnitude of prediction error.

To obtain a more complete picture of prediction errors, we create a *histogram* using the `hist()` function (see section 3.3.2).

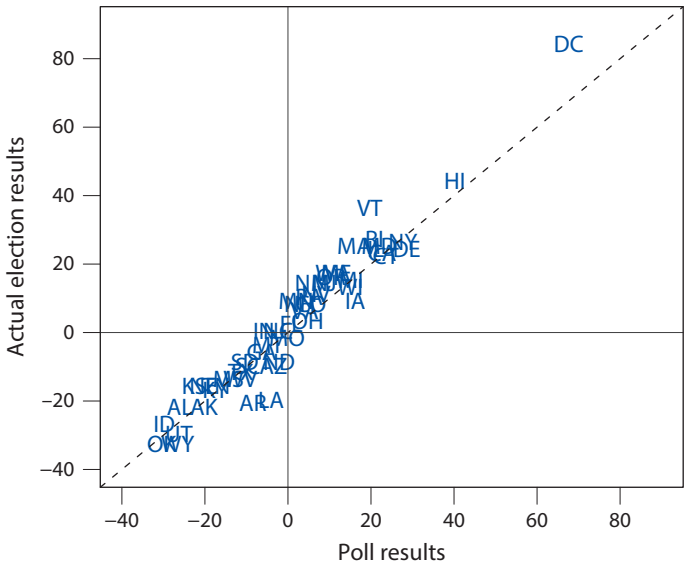
```
## histogram
hist(errors, freq = FALSE, ylim = c(0, 0.08),
     main = "Poll prediction error",
     xlab = "Error in predicted margin for Obama (percentage points)")
## add mean
abline(v = mean(errors), lty = "dashed", col = "blue")
text(x = -7, y = 0.08, "average error", col = "blue")
```



The histogram shows that the poll prediction error varies widely from one state to another. However, most errors are relatively small and larger errors are less likely to occur, yielding a *bell-shaped* distribution around zero.

We further examine the accuracy of poll predictions for each state by plotting them (horizontal axis) against the corresponding actual election results (vertical axis) using the two-letter state-name variable `state`. The states below (above) the 45-degree line indicate that the poll predictions were too favorable towards Obama (McCain). To plot text, we first create an "empty" plot by setting the `type` argument in the `plot()` function to `"n"` and then use the `text()` function to add state labels. As its first two arguments, the `text()` function takes the `x` and `y` coordinates for the location where the character string is to be plotted. The third argument of this function, `labels`, is a character vector of text labels to be plotted. In the current example, the x-coordinates and y-coordinates represent the poll predictions and Obama's actual margins.

```
## type = "n" generates "empty" plot
plot(poll.pred, pres08$margin, type = "n", main = "", xlab = "Poll results",
     xlim = c(-40, 90), ylim = c(-40, 90), ylab = "Actual election results")
## add state abbreviations
text(x = poll.pred, y = pres08$margin, labels = pres08$state, col = "blue")
## lines
abline(a = 0, b = 1, lty = "dashed") # 45-degree line
abline(v = 0)  # vertical line at 0
abline(h = 0)  # horizontal line at 0
```



Although for some states like the District of Columbia (DC) and Vermont (VT) the poll prediction is grossly inaccurate, this may not matter given that the US presidential election is essentially based on the winner-take-all system for each state. On the other hand, even when poll predictions are close to the actual election results in terms of percentage points, polls may predict the wrong candidate as the winner of a state. There are two types of prediction errors where the poll predictions chose the wrong winner. In the above plot, for the states that are plotted in the upper-left quadrant, Obama was predicted to lose (because the poll results are negative) but he actually won the states (because the actual election results are positive). Conversely, for the states in the lower-right quadrant, Obama was predicted to win but actually lost the states. The plot suggests that the poll predictions accurately chose the winner for most states. However, three states, which the poll predictions called wrongly, had a close race with the margin of victory approximately equal to 1 percentage point. We can use the `sign()` function to determine the sign of `poll.pred` and `pres08$margin` for each state. The function returns `1` if positive (Obama wins) and `-1` if negative (McCain wins) (`0` if zero, a tie).

Table 4.3. Confusion Matrix.

	Actual outcome	
	Positive	Negative
Predicted outcome		
Positive	true positive	false positive
Negative	false negative	true negative

Note: There are two types of correct classification, true positive and true negative. Similarly, false positive and false negative are two kinds of misclassification.

```
## which state polls called wrong?
pres08$state[sign(poll.pred) != sign(pres08$margin)]

## [1] IN MO NC
## 51 Levels: AK AL AR AZ CA CO CT DC DE FL GA HI IA ID ... WY

## what was the actual margin for these states?
pres08$margin[sign(poll.pred) != sign(pres08$margin)]

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

The problem of predicting the outcome category or class is called *classification*. In the current context, for each state, we would like to predict whether Obama wins or not. In a classification problem, prediction is either exactly correct or incorrect, and an incorrect prediction is called *misclassification*. In our analysis, the misclassification rate is 3/51, which is about 6 percent.

In a binary classification problem, there are two types of misclassification. We may predict Obama to be the winner for a state where he actually lost the election. Conversely, Obama may be predicted to lose a state and yet in the actual election win it. If we regard Obama's victory (rather than his loss) as the "positive" outcome, then the former type of misclassification is called *false positive* whereas the latter is *false negative*. In the current example, Missouri (`MO`) is a false positive while Illinois (`IN`) and North Carolina (`NC`) are false negatives. Table 4.3 presents a *confusion matrix* where the two types of misclassification and correct classification are shown.

Classification refers to the problem of predicting a categorical outcome. Classification is either correct or incorrect. In a binary classification problem, there are two types of **misclassification**: false positive and false negative, representing incorrectly predicted positive and negative outcomes, respectively.

Finally, we can compute the number of Electoral College votes for Obama based on the poll predictions and compare it against the actual result, which was 364 votes. Since 270 votes was the winning threshold, the results show that the polls correctly

called Obama the elected president. The predicted total number of Electoral College votes was 15 fewer than the actual election result.[3]

```
## actual results: total number of electoral votes won by Obama
sum(pres08$EV[pres08$margin > 0])
## [1] 364
## poll prediction
sum(pres08$EV[poll.pred > 0])
## [1] 349
```

While the popular vote does not determine the election outcome, we can also examine the accuracy of national polls and how public opinion changed over the course of the campaign. To do this, we analyze the national polls contained in the CSV file `pollsUS08.csv`. The names and descriptions of the variables in this data set are identical to those of the last four variables in table 4.2. For each of the last 90 days of the campaign, we compute the average of support for each candidate using all polls taken within the past week and examine how it changes as Election Day nears. This can be done with a loop, where for a given day we take all polls that were conducted within the previous 7 days and on the corresponding day. We then compare these poll-based predictions against the actual vote shares in the election, which were 52.9% and 45.7% for Obama and McCain, respectively. Using the code for state polls above as a template, we construct the following code chunk.

```
## load the data
pollsUS08 <- read.csv("pollsUS08.csv")
## compute number of days to the election as before
pollsUS08$middate <- as.Date(pollsUS08$middate)
pollsUS08$DaysToElection <- as.Date("2008-11-04") - pollsUS08$middate
## empty vectors to store predictions
Obama.pred <- McCain.pred <- rep(NA, 90)
for (i in 1:90) {
    ## take all polls conducted within the past 7 days
    week.data <- subset(pollsUS08, subset = ((DaysToElection <= (90 - i + 7))
                                & (DaysToElection > (90 - i))))
    ## compute support for each candidate using the average
    Obama.pred[i] <- mean(week.data$Obama)
    McCain.pred[i] <- mean(week.data$McCain)
}
```

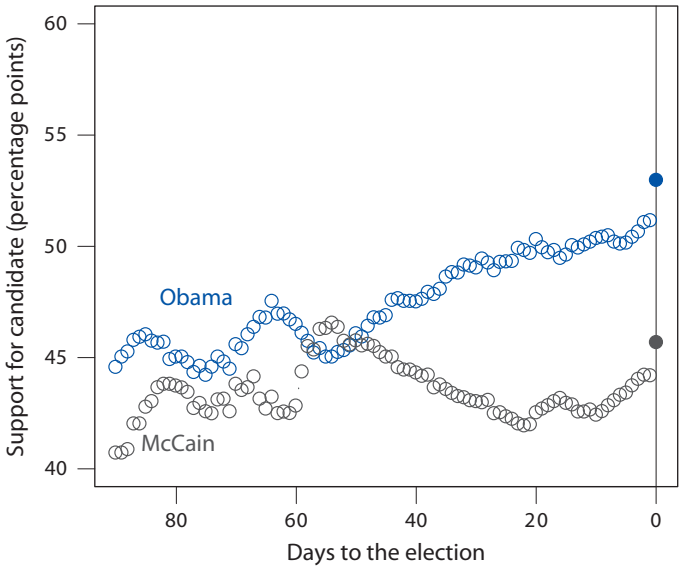
Note that in the above code we utilize shortcut syntax to assign the same value to multiple objects. Specifically, we use the single expression `x <- y <- z` rather than

[3] As noted earlier, Obama received one vote from Nebraska even though he lost the statewide vote.

two separate expressions, `x <- z` and `y <- z`, in order to assign the same value `z` to two objects, `x` and `y`. Furthermore, within the loop, we subset the data `pollsUS08` so that the resulting data contain only the polls conducted within the past 7 days and the day itself. For example, when the loop starts (i.e., `i` is equal to `1`), we subset the polls for which the `DaysToElection` variable is less than or equal to 96 ($= 90 - 1 + 7$) and greater than 89 ($= 90 - 1$). In the final iteration (i.e., `i` is equal to 90), this variable for the subsetted data takes a value less than or equal to 7 ($= 90 - 90 + 7$) and greater than 0 ($= 90 - 90$).

We now display the results using a *time-series plot*. We define the horizontal axis such that its leftmost value is 90 days prior to Election Day and its rightmost value is Election Day. This can be done by specifying the `xlim` argument to be `c(90, 0)` instead of `c(0, 90)`. The plot at the bottom uses gray circles rather than red for the states won by McCain. See page C3 for the full-color version.

```
## plot going from 90 days to 1 day before the election
plot(90:1, Obama.pred, type = "b", xlim = c(90, 0), ylim = c(40, 60),
     col = "blue", xlab = "Days to the election",
     ylab = "Support for candidate (percentage points)")
## type = "b" gives plot that includes both points and lines
lines(90:1, McCain.pred, type = "b", col = "red")
## actual election results: pch = 19 gives solid circles
points(0, 52.93, pch = 19, col = "blue")
points(0, 45.65, pch = 19, col = "red")
## line indicating Election Day
abline(v = 0)
## labeling candidates
text(80, 48, "Obama", col = "blue")
text(80, 41, "McCain", col = "red")
```



Which person is the most competent?

Figure 4.2. Example Pictures of Candidates Used in the Experiment. Source: A. Todorov et al. (2005) *Science*, vol. 308, no. 10 (June), pp. 1623–1626.

The resulting figure demonstrates the reasonable accuracy of preelection polls in terms of margin. Indeed, the Election Day margin (the difference between two solid circles) almost coincides with the predicted margin based on the polls taken within a week prior to the election. It is also interesting that public opinion shifts quite a bit during the course of campaign. Two months before the election, support for Obama was roughly tied with that for McCain. However, as Election Day approached, Obama's margin over McCain gradually increased. On Election Day, it was more than 7 percentage points. It is also worth noting that the proportion of other voters who were either undecided or supported third-party candidates declined.

4.2 Linear Regression

In the previous section, we used polling data to predict election outcomes. When doing so, we simply used the average of poll predictions. An alternative method of prediction is based on a statistical model. In this section, we introduce one of the most basic statistical models, called *linear regression*.

4.2.1 FACIAL APPEARANCE AND ELECTION OUTCOMES

Several psychologists have reported the intriguing result of an experiment showing that facial appearance predicts election outcomes better than chance.[4] In their experiment, the researchers briefly showed student subjects the black-and-white head shots of two candidates from a US congressional election (winner and runner-up). Figure 4.2 shows example pictures of the candidates from the 2004 Wisconsin Senate race. Russ Feingold of the Democratic Party (left) was the actual winner, and Tim Michels of the Republican Party (right) was the runner-up. The exposure of subjects to facial pictures lasted less than a second, and the subjects were then asked to evaluate the two candidates in terms of their perceived competence.

[4] This section is based on Alexander Todorov, Anesu N. Mandisodza, Amir Goren, and Crystal C. Hall (2005) "Inferences of competence from faces predict election outcomes." *Science*, vol. 308, no. 10 (June), pp. 1623–1626.

Table 4.4. Facial Appearance Experiment Data.

Variable	*Description*
`congress`	session of Congress
`year`	year of the election
`state`	state of the election
`winner`	name of the winner
`loser`	name of the runner-up
`w.party`	party of the winner
`l.party`	party of the loser
`d.votes`	number of votes for the Democratic candidate
`r.votes`	number of votes for the Republican candidate
`d.comp`	competence measure for the Democratic candidate
`r.comp`	competence measure for the Republican candidate

The researchers used these competence measures to predict election outcomes. The key hypothesis is whether or not a within-a-second evaluation of facial appearance can predict election outcomes. The CSV data set, `face.csv`, contains the data from the experiment. Table 4.4 presents the names and descriptions of the variables in this data set. Note that we include data only from subjects who did not know the candidates' political parties, their policies, or even which candidate was the incumbent or challenger. They were simply making snap judgments about which candidate appeared more competent based on their facial expression alone.

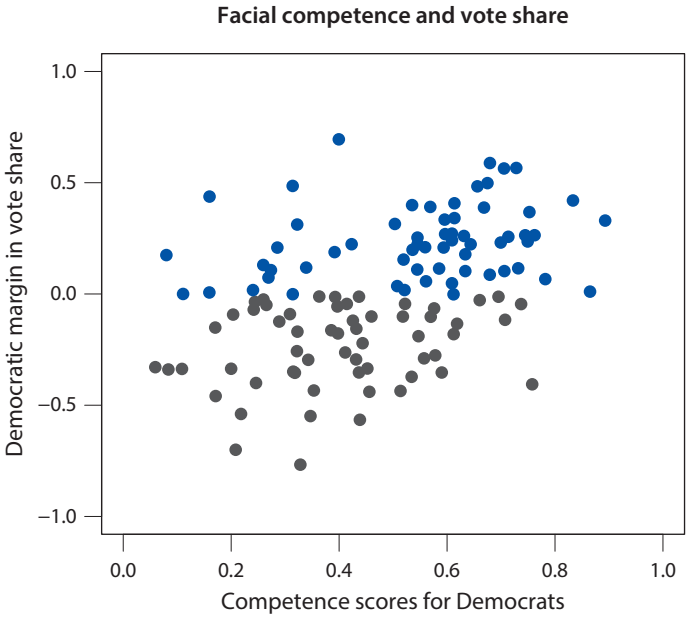
We begin our analysis of the facial appearance experiment data by creating a *scatter plot* of the competence measure against election outcomes. To do this, we create the win margins for Democratic candidates as the difference in two-party vote shares for Democratic and Republican candidates. Positive win margins favor Democrats. A two-party vote share is the number of votes each candidate receives out of just those votes cast for a major party candidate (not out of all votes cast).

```
## load the data
face <- read.csv("face.csv")
## two-party vote share for Democrats and Republicans
face$d.share <- face$d.votes / (face$d.votes + face$r.votes)
face$r.share <- face$r.votes / (face$d.votes + face$r.votes)
face$diff.share <- face$d.share - face$r.share
```

Next, we use the `plot()` function to generate a scatter plot. To make the symbols more informative, we can change them based on variables in our data set. The argument `pch` for the `plot()` function can specify the type of points to be plotted (see section 3.6). We use the `ifelse()` function when specifying the `col` argument so that red dots are used for the races with Republican winners and blue dots are used for those with Democratic winners. The plot shows a mild upward trend in the Democratic margin as the competence score for Democrats increases.

```
plot(face$d.comp, face$diff.share, pch = 16,
     col = ifelse(face$w.party == "R", "red", "blue"),
     xlim = c(0, 1), ylim = c(-1, 1),
     xlab = "Competence scores for Democrats",
     ylab = "Democratic margin in vote share",
     main = "Facial competence and vote share")
```

The plot below uses gray circles instead of red circles for Republican winners. See page C3 for the full-color version.



4.2.2 CORRELATION AND SCATTER PLOTS

We learned in section 3.6.2 that correlation represents the degree to which one variable is associated with another. A positive (negative) value of correlation means that one variable is more (less) likely to be above (below) its mean when the other variable is above its own mean. The upwards-sloping data cloud in the above scatter plot shows a positive *correlation* between perceived competence and vote share differential. To compute the correlation coefficient, we use the function `cor()`.

```
cor(face$d.comp, face$diff.share)
## [1] 0.4327743
```

This correlation of about 0.4 tells us that there is a moderately positive relationship between a candidate's perceived competence and his or her actual margin of victory on Election Day. That is, candidates who appear more competent than their

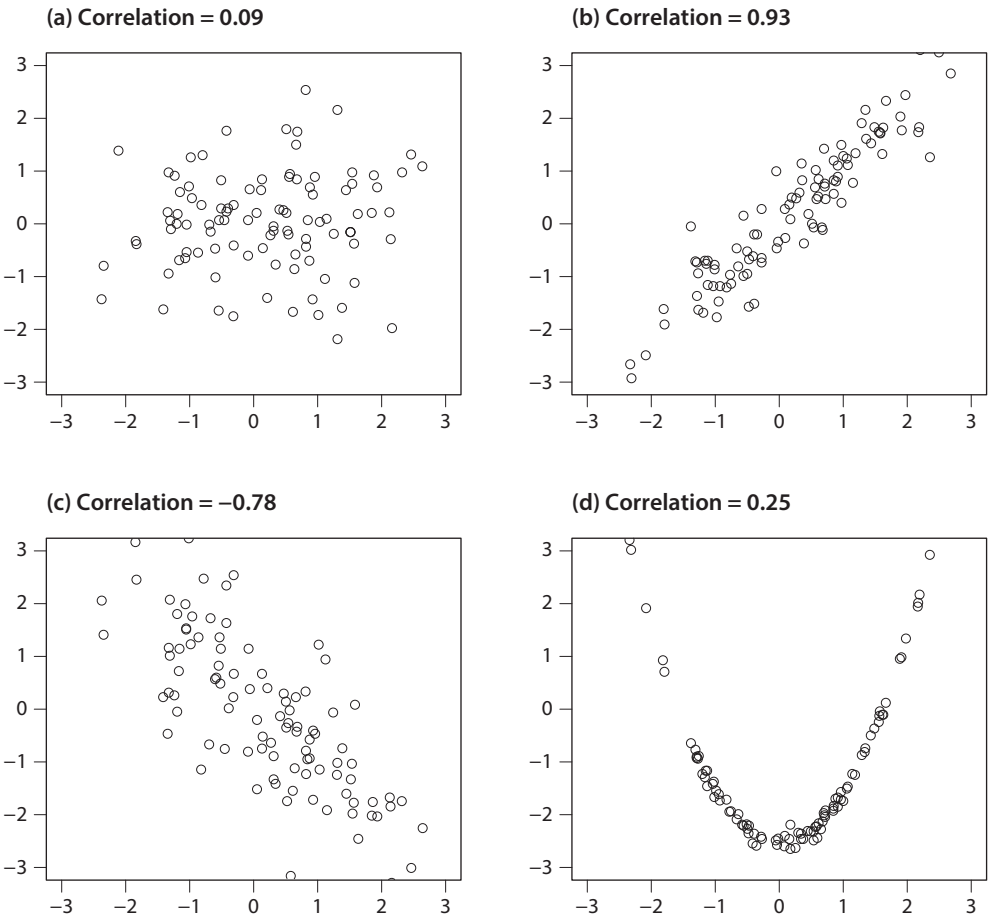


Figure 4.3. Correlation Coefficients and Patterns of the Data Cloud in Scatter Plots.

opponents—as rapidly judged by uninformed voters who don't recognize the candidates—are likely to win a higher share of the votes cast.

To get a better sense of the relationship between correlation coefficients and data cloud shapes, figure 4.3 presents four artificial data sets with various degrees of correlation. We observe that a positive (negative) correlation corresponds to an upwards (downwards) trend in the data cloud, and a greater magnitude of the correlation coefficient indicates a stronger linear relationship. Indeed, correlation represents a *linear relationship* between two variables. Perfect positive (negative) correlation, i.e., correlation of 1 (−1), would mean the two variables have a perfect linear relationship with data points located on a single line.

Thus, it is important for us to note that a lack of correlation does not necessarily imply a lack of a relationship. In panel (d), the correlation between the two variables is low but there is a clear *nonlinear relationship*, which in this case is a quadratic function.

The **correlation coefficient** quantifies the linear relationship between two variables. An upwards trend in the data cloud in a scatter plot implies a positive correlation, whereas a downwards trend in the data cloud represents a negative correlation. Correlation is often not suitable for representing a nonlinear relationship.

4.2.3 LEAST SQUARES

As shown above, correlation describes a linear relationship between two variables. However, such a relationship is best characterized using the following *linear model*:

$$Y = \underbrace{\alpha}_{\text{intercept}} + \underbrace{\beta}_{\text{slope}} X + \underbrace{\epsilon}_{\text{error term}} . \tag{4.1}$$

In the model, Y is the outcome or response variable and X is the predictor or independent (explanatory) variable. In the current application, we will use the perceived competence measure as the predictor and the difference in two-party vote share as the outcome. Recall that any line can be defined by the *intercept* α and the *slope* parameter β. The intercept α represents the average value of Y when X is zero. The slope β measures the average increase in Y when X increases by one unit. The intercept and slope parameters are together called *coefficients*. The *error* (or *disturbance*) term, ϵ, allows an observation to deviate from a perfect linear relationship.

We use a model like this under the assumption that it approximates the *data-generating process* well. However, as well-known statistician George Box has stated, we must recognize that "all models are wrong, but some are useful." Even if the data are not generated according to the linear model specified in equation (4.1), the model can be a useful tool to predict the outcome of interest.

Since the values of α and β in equation (4.1) are unknown to researchers, they must be estimated from the data. In statistics, the estimates of parameters are indicated by "hats," where $\hat{\alpha}$ and $\hat{\beta}$ represent the estimates of α and β, respectively. Once we obtain the estimated values of coefficients α and β, then we have the so-called *regression line*. We can use this line to predict the value of the outcome variable given that of a predictor. Specifically, given a particular value of the predictor, $X = x$, we compute the *predicted value* (or *fitted value*) of the outcome variable, denoted by $\widehat{Y}$, using the regression function

$$\widehat{Y} = \hat{\alpha} + \hat{\beta} x. \tag{4.2}$$

Most likely, the predicted value will not equal the observed value. The difference between the observed outcome and its predicted value is called the *residual* or *prediction error*. Formally, we can write the residual as

$$\hat{\epsilon} = Y - \widehat{Y}. \tag{4.3}$$

Notice that the residual is represented by ϵ with a hat. Since the error term ϵ in equation (4.1) is unobserved, the residual represents an estimate of this error term.

The **linear regression model** is defined as

$$Y = \alpha + \beta X + \epsilon,$$

where Y is the outcome (or response) variable, X is the predictor or the independent (or explanatory) variable, ϵ is the error (or disturbance) term, and (α, β) are the coefficients. The slope parameter β represents the increase in the average outcome associated with a one-unit increase in the predictor. Once the estimates of the coefficients $(\hat{\alpha}, \hat{\beta})$ are obtained from the data, we can predict the outcome, using a given value of the predictor $X = x$, as $\widehat{Y} = \hat{\alpha} + \hat{\beta}x$. The difference between the observed outcome and this fitted or predicted value $\widehat{Y}$ is called the residual and is denoted by $\hat{\epsilon} = Y - \widehat{Y}$.

To fit a linear regression model in R, we use the `lm()` function. This function takes a formula of the form `Y ~ X` as the main argument where the outcome variable is `Y` and the predictor is `X`, taken from a data frame specified as the `data` argument. Note that an intercept will be automatically added to the regression model.

We now obtain the regression line for the facial appearance experiment data. We use the Democratic margin in the two-party vote share as the response variable and the perceived competence for Democratic candidates as the predictor.

```
fit <- lm(diff.share ~ d.comp, data = face) # fit the model
fit
```

```
##
## Call:
## lm(formula = diff.share ~ d.comp, data = face)
##
## Coefficients:
## (Intercept)       d.comp
##     -0.3122       0.6604
```

The output shows that the estimated intercept is -0.3122 whereas the estimated slope is 0.6604. That is, when no experimental subject thinks a Democratic candidate is more competent than a Republican counterpart, the predicted Democratic margin of two-party vote share is approximately -31.2 percentage points. If the perceived competence score increases by 10 percentage points, then the outcome variable is predicted to increase on average by 6.6 ($= 0.6604 \times 10$) percentage points.

There is an alternative way of fitting the same model without the `data` argument. This requires specifying the entire names of objects for the outcome variable and the predictor as follows.

```
lm(face$diff.share ~ face$d.comp)
```

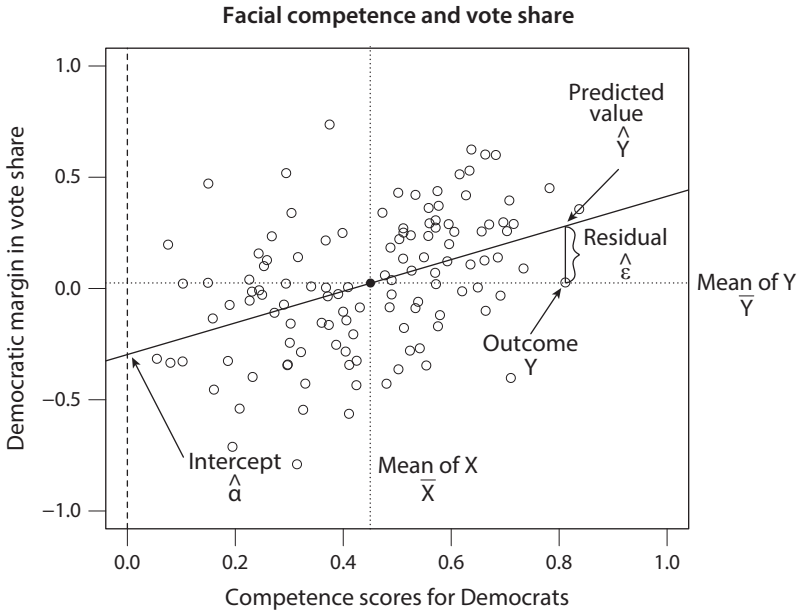
In general, this is not recommended because it unnecessarily complicates the syntax and may cause confusion. However, it may be useful when the variables we wish to use for regression exist as separate objects in the workspace.

In addition, to directly obtain the estimated coefficients ($\hat{\alpha}$, $\hat{\beta}$) and the predicted or fitted values $\widehat{Y}$, we can use the `coef()` and `fitted()` functions, respectively.

```
coef(fit)  # get estimated coefficients

## (Intercept)      d.comp
##  -0.3122259   0.6603815

head(fitted(fit))  # get fitted or predicted values

##           1           2           3           4           5
##  0.06060411 -0.08643340  0.09217061  0.04539236 0.13698690
##           6
##  -0.10057206
```

It is straightforward to add the regression line to the scatter plot using the `abline()` function which takes the output object from the `lm()` function as its input. The plot also shows the estimated intercept $\hat{\alpha}$ as well as the observed outcome Y, the predicted or fitted value $\hat{Y}$, and the residual $\hat{\epsilon}$ for one of the observations.

```
plot(face$d.comp, face$diff.share, xlim = c(0, 1.05), ylim = c(-1,1),
     xlab = "Competence scores for Democrats",
     ylab = "Democratic margin in vote share",
     main = "Facial competence and vote share")
abline(fit) # add regression line
abline(v = 0, lty = "dashed")
```



This regression line is the "line of best fit" because it minimizes the magnitude of prediction error. To estimate the line's intercept and slope parameters, a commonly used method is that of *least squares*. The idea is to choose $\hat{\alpha}$ and $\hat{\beta}$ such that together they minimize the *sum of squared residuals* (SSR), which is defined as

$$\text{SSR} = \sum_{i=1}^{n} \hat{\epsilon}_i^2 = \sum_{i=1}^{n} (Y_i - \widehat{Y}_i)^2 = \sum_{i=1}^{n} (Y_i - \hat{\alpha} - \hat{\beta} X_i)^2. \tag{4.4}$$

In the equation, Y_i, X_i, and $\hat{\epsilon}_i$ represent the outcome variable, the predictor, and the residual, respectively, for the ith observation, and n is the sample size. The second and third equalities follow from the definition of the residual given in equations (4.3) and (4.2), respectively. The value of SSR is difficult to interpret. However, we can use the idea of *root mean square* (RMS) introduced in section 2.6.2 and applied earlier. Specifically, we can compute the *root-mean-squared error* (RMSE) as

$$\text{RMSE} = \sqrt{\frac{1}{n}\text{SSR}} = \sqrt{\frac{1}{n}\sum_{i=1}^{n} \hat{\epsilon}_i^2}. \tag{4.5}$$

Therefore, RMSE represents the average magnitude of the prediction error for the regression, and this is what the method of least squares minimizes.

In R, RMSE can be easily calculated by first obtaining the residuals from the `resid()` function.

```
epsilon.hat <- resid(fit)   # residuals
sqrt(mean(epsilon.hat^2))   # RMSE

## [1] 0.2642361
```

The result implies that while the perceived competence score does predict the election outcome, the prediction is not very accurate, yielding on average a prediction error of 26 percentage points.

The least squares estimates of intercept and slope parameters are given by

$$\hat{\alpha} = \overline{Y} - \hat{\beta}\overline{X}, \tag{4.6}$$

$$\hat{\beta} = \frac{\sum_{i=1}^{n}(Y_i - \overline{Y})(X_i - \overline{X})}{\sum_{i=1}^{n}(X_i - \overline{X})^2}. \tag{4.7}$$

Recall that the sample means of Y and X are given by $\overline{Y} = \frac{1}{n}\sum_{i=1}^{n} Y_i$ and $\overline{X} = \frac{1}{n}\sum_{i=1}^{n} X_i$, respectively. The results imply that the regression line always goes through the center of the data $(\overline{X}, \overline{Y})$. This is so because substituting $x = \overline{X}$ into equation (4.2) and using the expression for $\hat{\alpha}$ in equation (4.6) yields $\widehat{Y} = \overline{Y}$:

$$\widehat{Y} = \underbrace{(\overline{Y} - \hat{\beta}\overline{X})}_{\hat{\alpha}} + \hat{\beta}\overline{X} = \overline{Y}.$$

In the above plot, we observe that this is indeed the case. The regression line runs through the intersection of the vertical and horizontal dotted lines, which represent the means of X and Y, respectively.

In addition, when the method of least squares is used to estimate the coefficients, the predictions based on the fitted regression line are accurate on average. More precisely, the mean of residual $\hat{\epsilon}$ is zero, as the following algebraic manipulation shows:

$$\text{mean of } \hat{\epsilon} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{\alpha} - \hat{\beta}X_i) = \overline{Y} - \hat{\alpha} - \hat{\beta}\overline{X} = 0.$$

In this equation, the first equality is due to the definition of the residual, the next equality is obtained by applying the summation for each term in the parentheses, and the final equality follows from equation (4.6). We emphasize that this is an algebraic equality and holds for *any* data set. In other words, a linear regression model always has zero average prediction error across all data points in the sample, but this does not necessarily mean that the linear regression model accurately represents the actual data-generating process.

A common method of estimating the coefficients of the linear regression model is the method of **least squares**, which minimizes the sum of squared residuals,

$$\text{SSR} = \textstyle\sum_{i=1}^{n}\hat{\epsilon}_i^2 = \sum_{i=1}^{n}(Y_i - \hat{\alpha} - \hat{\beta}X_i)^2.$$

The mean of residuals is always zero, and the regression line always goes through the center of data $(\overline{X}, \overline{Y})$ where $\overline{X}$ and $\overline{Y}$ are the sample means of X and Y, respectively.

It is also important to understand the relationship between the estimated slope of the regression and the correlation coefficient introduced in section 3.6.2:

$$\hat{\beta} = \frac{1}{n}\sum_{i=1}^{n}\frac{(Y_i - \overline{Y})(X_i - \overline{X})}{\sqrt{\frac{1}{n}\sum_{i=1}^{n}(Y_i - \overline{Y})^2}\sqrt{\frac{1}{n}\sum_{i=1}^{n}(X_i - \overline{X})^2}} \times \frac{\sqrt{\frac{1}{n}\sum_{i=1}^{n}(Y_i - \overline{Y})^2}}{\sqrt{\frac{1}{n}\sum_{i=1}^{n}(X_i - \overline{X})^2}}$$

$$= \text{correlation of } X \text{ and } Y \times \frac{\text{standard deviation of } Y}{\text{standard deviation of } X}. \tag{4.8}$$

The first equality holds because we divide and multiply the right hand side of equation (4.7) by the standard deviation of Y, i.e., $\sqrt{\frac{1}{n}\sum_{i=1}^{n}(Y_i - \overline{Y})^2}$, whereas the second equality follows from the definitions of correlation and standard deviation (see equations (3.2) and (2.4), respectively).

The expression for the estimated slope parameter in equation (4.8) has two important implications. First, a positive (negative) correlation corresponds to a positive (negative) slope because standard deviations never take a negative value. Second, each increase of 1 standard deviation in X is associated with an average increase of ρ standard deviations in Y, where ρ is the correlation between X and Y. For example, if the correlation is 0.5, then a 1 standard deviation increase in X would result in a 0.5 standard deviation increase in Y. In the current example, the correlation between the

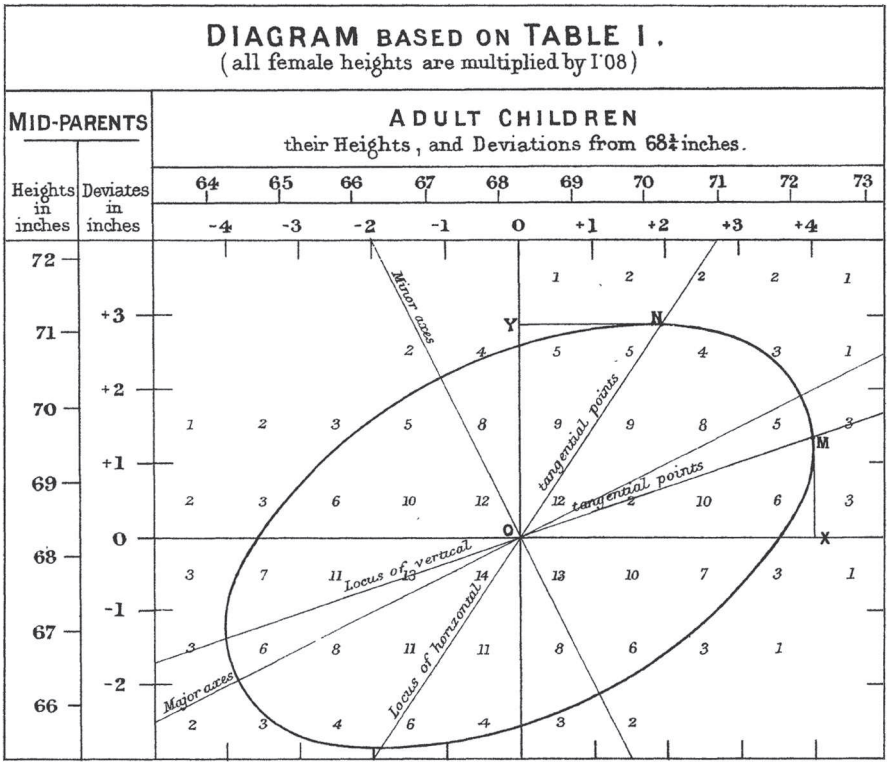


Figure 4.4. Galton's Regression Towards Mediocrity. Source: Francis Galton (1886) "Regression towards mediocrity in hereditary stature." *Journal of the Anthropological Institute of Great Britain and Ireland*, vol. 15, pp. 246–263.

perceived competence score and the two-party vote share differential is 0.43, whereas the standard deviations of X and Y are 0.19 and 0.29, respectively. Thus, an increase in the perceived competence score of 0.19 is associated with an average increase in the two-party vote share differential of roughly 13 percentage points ($\approx 0.43 \times 0.29$).

> The estimated **slope coefficient** from a linear regression model equals the ρ standard deviation unit increase in the outcome variable that is associated with an increase of 1 standard deviation in the predictor, where ρ is the correlation between the two variables.

4.2.4 REGRESSION TOWARDS THE MEAN

In his 1886 paper entitled "Regression towards mediocrity in hereditary stature," a British scholar, Sir Francis Galton, conducted one of the first regression analyses. He studied human hereditary stature by examining the relationship between the height of adult children and the average of their parents' heights, which Galton called the "mid-parents' height." Galton was the first to present an example of the phenomenon called *regression towards the mean*. He summarized this as "When Mid Parents are shorter (taller) than mediocrity, their Children tend to be taller (shorter) than they."

Figure 4.4 is taken from the original paper. In this figure, the values indicate the number of observations and the ellipse represents the data cloud. The "locus of

vertical tangential points" represents a regression line where the outcome variable is adult children's height (horizontal axis) and the predictor is their mid-parents' height (vertical axis). Note that the outcome variable is measured on the horizontal axis while the predictor is on the vertical axis, which is the exact opposite of the current practice of plotting the outcome variable on the vertical axis. Galton also regressed mid-parents' heights on the heights of adult children. This regression line is denoted by the "locus of horizontal tangential points." The angle of the slope of this regression line, which Galton calculated to be 2/3, represents the rate of regression from mid-parents to children.

To demonstrate the regression effect numerically, consider the observations that have mid-parents' heights of approximately 71 inches. As we can see from figure 4.4, there are 24 such observations, represented by those in the second row from the top. Out of these observations, only 8, or 33% of them, have children who are at least as tall as their mid-parents. In contrast, focus on the observations whose mid-parents are about 67 inches and hence shorter than the average height (they are in the second row from the bottom). Out of 57 such observations, 40 observations, or 70%, have children whose heights are at least their mid-parents' height. Galton called this pattern the "regression towards mediocrity." Note, however, that as indicated by the positive slope of the regression line, children whose parents are taller also tend to be taller on average. We emphasize that as shown in chapter 6 this empirical phenomenon can be explained by chance alone. Thus, regression towards the mean does not imply that human heights are converging and everyone will have an identical height in the future!

Regression towards the mean is observed in other contexts as well. Below, we show another example of this phenomenon, demonstrating that Obama tended to gain fewer votes in 2012 than in 2008 for the states in which he did well in 2008. Other examples include test scores where students who perform well in the midterm exam tend not to do as well in the final exam. An important point is that this decline in performance may have arisen due to chance rather than to a lack of Obama's or the students' efforts.

Regression towards the mean represents an empirical phenomenon where an observation with a value of the predictor further away from the distribution's mean tends to have a value of an outcome variable closer to that mean. This tendency can be explained by chance alone.

4.2.5 MERGING DATA SETS IN R

We will examine whether or not the US presidential election data exhibit the regression towards the mean phenomenon. To do this, we use Obama's vote share in the 2008 election to predict his vote share in his 2012 reelection. We *merge* the 2012 election result data set, `pres12.csv`, into the 2008 election data set. The variable names and descriptions of the 2012 election result data set are given in table 4.5.

Merging two data sets can be done in R using the `merge()` function. The function takes three main arguments, `x`, `y`, and `by`, where the `x` and `y` arguments represent two data frames to be merged and the `by` argument indicates the variable name(s) used for merging. Let's first look at two data sets we would like to merge.

Table 4.5. 2012 US Presidential Election Data.

Variable	*Description*
`state`	abbreviated name of the state
`Obama`	Obama's vote share (percentage)
`Romney`	Romney's vote share (percentage)
`EV`	number of Electoral College votes for the state

```
pres12 <- read.csv("pres12.csv")  # load 2012 data
## quick look at two data sets
head(pres08)

##   state.name state Obama McCain EV margin
## 1    Alabama    AL    39     60  9    -21
## 2     Alaska    AK    38     59  3    -21
## 3    Arizona    AZ    45     54 10     -9
## 4   Arkansas    AR    39     59  6    -20
## 5 California    CA    61     37 55     24
## 6   Colorado    CO    54     45  9      9

head(pres12)

##   state Obama Romney EV
## 1    AL    38     61  9
## 2    AK    41     55  3
## 3    AZ    45     54 11
## 4    AR    37     61  6
## 5    CA    60     37 55
## 6    CO    51     46  9
```

We will use the state name variable `state`, which is contained in both data sets, to merge the two data frames.

```
## merge two data frames
pres <- merge(pres08, pres12, by = "state")
## summarize the merged data frame
summary(pres)

##      state          state.name    Obama.x
##  AK     : 1   Alabama    : 1   Min.   :33.00
##  AL     : 1   Alaska     : 1   1st Qu.:43.00
##  AR     : 1   Arizona    : 1   Median :51.00
##  AZ     : 1   Arkansas   : 1   Mean   :51.37
##  CA     : 1   California: 1   3rd Qu.:57.50
```

```
##  CO     : 1   Colorado  : 1   Max.   :92.00
##  (Other):45   (Other)   :45
##      McCain           EV.x           margin
##  Min.   : 7.00   Min.   : 3.00   Min.   :-32.000
##  1st Qu.:40.00   1st Qu.: 4.50   1st Qu.:-13.000
##  Median :47.00   Median : 8.00   Median :  4.000
##  Mean   :47.06   Mean   :10.55   Mean   :  4.314
##  3rd Qu.:56.00   3rd Qu.:11.50   3rd Qu.: 17.500
##  Max.   :66.00   Max.   :55.00   Max.   : 85.000
##
##     Obama.y          Romney           EV.y
##  Min.   :25.00   Min.   : 7.00   Min.   : 3.00
##  1st Qu.:40.50   1st Qu.:41.00   1st Qu.: 4.50
##  Median :51.00   Median :48.00   Median : 8.00
##  Mean   :49.06   Mean   :49.04   Mean   :10.55
##  3rd Qu.:56.00   3rd Qu.:58.00   3rd Qu.:11.50
##  Max.   :91.00   Max.   :73.00   Max.   :55.00
##
```

Note that if the data frames have variables with identical names, i.e., `Obama` and `EV`, then the merged data frame will append `.x` and `.y` to each name, thereby attributing each variable to its original data frame. The variable used for merging must exist in both data frames. This variable may have the same name in both data frames, as in the above code chunk, but if the variable happens to have different names, then we can use the `by.x` and `by.y` arguments to specify the exact variable names used in each data frame. By default, the merged data frame keeps the name of the variable from data frame `x`, which is specified by the `by.x` argument. An example code chunk is given here.

```
## change the variable name for illustration
names(pres12)[1] <- "state.abb"
## merging data sets using the variables of different names
pres <- merge(pres08, pres12, by.x = "state", by.y = "state.abb")
summary(pres)
```

```
##      state          state.name    Obama.x
##  AK     : 1   Alabama   : 1   Min.   :33.00
##  AL     : 1   Alaska    : 1   1st Qu.:43.00
##  AR     : 1   Arizona   : 1   Median :51.00
##  AZ     : 1   Arkansas  : 1   Mean   :51.37
##  CA     : 1   California: 1   3rd Qu.:57.50
##  CO     : 1   Colorado  : 1   Max.   :92.00
##  (Other):45   (Other)   :45
##      McCain           EV.x           margin
##  Min.   : 7.00   Min.   : 3.00   Min.   :-32.000
```

```
##  1st Qu.:40.00   1st Qu.: 4.50   1st Qu.:-13.000
##  Median :47.00   Median : 8.00   Median :  4.000
##  Mean   :47.06   Mean   :10.55   Mean   :  4.314
##  3rd Qu.:56.00   3rd Qu.:11.50   3rd Qu.: 17.500
##  Max.   :66.00   Max.   :55.00   Max.   : 85.000
##
##     Obama.y          Romney           EV.y
##  Min.   :25.00   Min.   : 7.00   Min.   : 3.00
##  1st Qu.:40.50   1st Qu.:41.00   1st Qu.: 4.50
##  Median :51.00   Median :48.00   Median : 8.00
##  Mean   :49.06   Mean   :49.04   Mean   :10.55
##  3rd Qu.:56.00   3rd Qu.:58.00   3rd Qu.:11.50
##  Max.   :91.00   Max.   :73.00   Max.   :55.00
##
```

An alternative way of combining two data frames is the `cbind()` function, which enables column-binding of multiple data frames. (As a side note, the `rbind()` function performs row-binding of multiple data frames by stacking one below another.) But sometimes problematically, the `cbind()` function assumes the proper sorting of data frames such that corresponding observations appear in the same row of the data frames. In our current application, each state must appear in the same row of the two data frames. The `merge()` function, on the other hand, appropriately sorts the data frames according to the variable used for merging. Another disadvantage of the `cbind()` function is that it preserves all columns in both data frames even when they represent the same variable, containing identical information.

The code chunk below illustrates these two problems. The resulting merged data frame keeps all variables from both data frames, and more importantly, the merged data frame has incorrect information for the District of Columbia (`DC`) and Delaware (`DE`) because their order is different in the two original data frames. In contrast, the `merge()` function will sort the second data frame, `pres12`, appropriately to match with the first data frame, `pres08`.

```
## cbinding two data frames
pres1 <- cbind(pres08, pres12)
## this shows all variables are kept
summary(pres1)
```

```
##       state.name     state        Obama
##  Alabama    : 1   AK     : 1   Min.   :33.00
##  Alaska     : 1   AL     : 1   1st Qu.:43.00
##  Arizona    : 1   AR     : 1   Median :51.00
##  Arkansas   : 1   AZ     : 1   Mean   :51.37
##  California: 1   CA     : 1   3rd Qu.:57.50
##  Colorado   : 1   CO     : 1   Max.   :92.00
##  (Other)    :45   (Other):45
```

```
##      McCain             EV            margin
##  Min.   : 7.00   Min.   : 3.00   Min.   :-32.000
##  1st Qu.:40.00   1st Qu.: 4.50   1st Qu.:-13.000
##  Median :47.00   Median : 8.00   Median :  4.000
##  Mean   :47.06   Mean   :10.55   Mean   :  4.314
##  3rd Qu.:56.00   3rd Qu.:11.50   3rd Qu.: 17.500
##  Max.   :66.00   Max.   :55.00   Max.   : 85.000
##
##    state.abb      Obama           Romney
##  AK      : 1   Min.   :25.00   Min.   : 7.00
##  AL      : 1   1st Qu.:40.50   1st Qu.:41.00
##  AR      : 1   Median :51.00   Median :48.00
##  AZ      : 1   Mean   :49.06   Mean   :49.04
##  CA      : 1   3rd Qu.:56.00   3rd Qu.:58.00
##  CO      : 1   Max.   :91.00   Max.   :73.00
##  (Other):45
##        EV
##  Min.   : 3.00
##  1st Qu.: 4.50
##  Median : 8.00
##  Mean   :10.55
##  3rd Qu.:11.50
##  Max.   :55.00
##
## DC and DE are flipped in this alternative approach
pres1[8:9, ]
##   state.name state Obama McCain EV margin state.abb Obama
## 8       D.C.    DC    92      7  3     85        DE    59
## 9   Delaware    DE    62     37  3     25        DC    91
##   Romney EV
## 8     40  3
## 9      7  3
## merge() does not have this problem
pres[8:9, ]
##   state state.name Obama.x McCain EV.x margin Obama.y
## 8    DC       D.C.      92      7    3     85      91
## 9    DE   Delaware      62     37    3     25      59
##   Romney EV.y
## 8      7    3
## 9     40    3
```

Using the merged data frame, we investigate whether or not the regression towards the mean phenomenon exists in the US presidential election data. Given the recent

trend of increasing polarization in American politics (see section 3.5), we standardize vote shares across elections by computing their *z-scores* so that we can measure Obama's electoral performance in each state relative to his average performance of that year (see section 3.6.2). That is, we subtract the mean from Obama's vote share in each election and then divide it by the standard deviation. This can be done easily by using the `scale()` function. We perform this transformation because technically, the regression towards the mean phenomenon holds when both the outcome and explanatory variables are standardized.

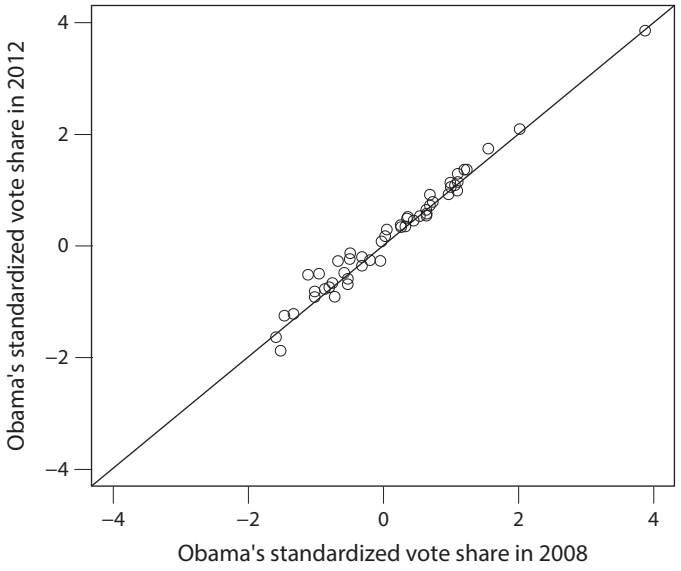
```
pres$Obama2008.z <- scale(pres$Obama.x)
pres$Obama2012.z <- scale(pres$Obama.y)
```

We regress Obama's 2012 standardized vote share on his 2008 standardized vote share. As expected, we observe a strong positive linear relationship between the two. Obama tended to receive more votes in 2012 from states that gave him more votes in 2008. Note that when we standardize both the outcome variable and the predictor, the estimated intercept becomes zero. This is because the estimated intercept is given by $\hat{\alpha} = \overline{Y} - \hat{\beta}\overline{X}$ (see equation (4.6)) and after standardizing, the sample means of both variables, $\overline{Y}$ and $\overline{X}$, are zero. As shown below, in this case, R estimates the intercept to be essentially zero. It is also possible to fit the model without an intercept by including `-1` in the formula.

```
## intercept is estimated as essentially zero
fit1 <- lm(Obama2012.z ~ Obama2008.z, data = pres)
fit1

##
## Call:
## lm(formula = Obama2012.z ~ Obama2008.z, data = pres)
##
## Coefficients:
## (Intercept)  Obama2008.z
##  -3.521e-17    9.834e-01

## regression without an intercept; estimated slope is identical
fit1 <- lm(Obama2012.z ~ -1 + Obama2008.z, data = pres)
fit1

##
## Call:
## lm(formula = Obama2012.z ~ -1 + Obama2008.z, data = pres)
##
## Coefficients:
## Obama2008.z
##      0.9834
```

Here, we plot the fitted regression line as well as the data points where we observe a strong linear relationship.

```
plot(pres$Obama2008.z, pres$Obama2012.z, xlim = c(-4, 4), ylim = c(-4, 4),
     xlab = "Obama's standardized vote share in 2008",
     ylab = "Obama's standardized vote share in 2012")
abline(fit1) # draw a regression line
```



Now we compute the proportion of states where Obama received a greater share of standardized votes in 2012 than he did in 2008. We do so using first the bottom quartile of Obama's 2008 (standardized) vote share, then the top quartile. If the regression towards the mean phenomenon exists, then this proportion should be greater for the states in the bottom quartile than those in the top quartile.

```
## bottom quartile
mean((pres$Obama2012.z >
          pres$Obama2008.z)[pres$Obama2008.z
                            <= quantile(pres$Obama2008.z, 0.25)])
## [1] 0.5714286
## top quartile
mean((pres$Obama2012.z >
          pres$Obama2008.z)[pres$Obama2008.z
                            >= quantile(pres$Obama2008.z, 0.75)])
## [1] 0.4615385
```

In the above code, we use the `quantile()` function to compute the top and bottom quartiles. Then, a logical vector where `TRUE` (`FALSE`) indicates Obama's 2012 vote share being greater than (less than or equal to) his 2008 vote share is subsetted by another logical vector. This second logical vector, inside the square brackets, indicates whether Obama's 2008 vote share for a state is in the bottom or top quartile. The result clearly shows the regression towards the mean phenomenon. Obama fared better in 2012 than in 2008 in 57% of bottom quartile states, where he failed most in 2008. In contrast, Obama fared better in 2012 only among 46% of the top quartile states, where he succeeded most in 2008.

4.2.6 MODEL FIT

Model fit measures how well the model fits the data, i.e., how accurately the model predicts observations. We can assess model fit by looking at the *coefficient of determination*, or R^2, which represents the proportion of total variation in the outcome variable explained by the model. To define R^2, we first introduce the *total sum of squares* or TSS, which is defined as

$$\text{TSS} = \sum_{i=1}^{n} (Y_i - \overline{Y})^2.$$

The TSS represents the total variation of the outcome variable based on the square distance from its mean. Now, we can define R^2 as the proportion of TSS explained by the predictor X:

$$R^2 = \frac{\text{TSS} - \text{SSR}}{\text{TSS}} = 1 - \frac{\text{SSR}}{\text{TSS}}.$$

The SSR or sum of squared residuals is defined in equation (4.4) and represents the residual variation of Y left unexplained by X. The value of R^2 ranges from 0 (when the correlation between the outcome and the predictor is 0) to 1 (when the correlation is 1), indicating how well the linear model fits the data at hand.

> The **coefficient of determination** is a measure of model fit and represents the proportion of variation in the outcome variable explained by the predictor. It is defined as one minus the ratio of the sum of squared residuals (SSR) to the total sum of squares (TSS).

As an illustrative example, consider the problem of predicting the 2000 US election results in Florida using the 1996 US election results from the same state at the county level. In Florida, there are 68 counties, and the CSV file `florida.csv` contains the number of votes cast for each candidate in those two elections. Table 4.6 displays the names and descriptions of variables in this data file. We focus on libertarian candidates

Table 4.6. 1996 and 2000 US Presidential Election Data for Florida Counties.

Variable	*Description*
`county`	county name
`Clinton96`	Clinton's votes in 1996
`Dole96`	Dole's votes in 1996
`Perot96`	Perot's votes in 1996
`Bush00`	Bush's votes in 2000
`Gore00`	Gore's votes in 2000
`Buchanan00`	Buchanan's votes in 2000

Ross Perot in 1996 and Pat Buchanan in 2000, using the votes for the former to predict the votes for the latter. We then compute R^2 from this regression model by first computing TSS and then SSR. Recall that the `resid()` function extracts the vector of residuals from the regression output.

```
florida <- read.csv("florida.csv")
## regress Buchanan's 2000 votes on Perot's 1996 votes
fit2 <- lm(Buchanan00 ~ Perot96, data = florida)
fit2

##
## Call:
## lm(formula = Buchanan00 ~ Perot96, data = florida)
##
## Coefficients:
## (Intercept)      Perot96
##     1.34575      0.03592

## compute TSS (total sum of squares) and SSR (sum of squared residuals)
TSS2 <- sum((florida$Buchanan00 - mean(florida$Buchanan00))^2)
SSR2 <- sum(resid(fit2)^2)
## coefficient of determination
(TSS2 - SSR2) / TSS2

## [1] 0.5130333
```

The result shows that 51% of the variation of Buchanan's 2000 votes can be explained by Perot's 1996 votes.

We turn this calculation into a function so that we can easily compute the coefficient of determination for different regression models (see section 1.3.4). The function takes as input the output from the `lm()` function, which is a list object containing various elements (see section 3.7.2). The value of the outcome variable can be recomputed from

the regression output object by summing the fitted value, which can be obtained using the `fitted()` function, and the residual for each observation.

```
R2 <- function(fit) {
    resid <- resid(fit) # residuals
    y <- fitted(fit) + resid # outcome variable
    TSS <- sum((y - mean(y))^2)
    SSR <- sum(resid^2)
    R2 <- (TSS - SSR) / TSS
    return(R2)
}
R2(fit2)
```

```
## [1] 0.5130333
```

Alternatively, we can obtain the value of R^2 by applying the `summary()` function to the output from the `lm()` function (see also section 7.3).

```
## built-in R function
fit2summary <- summary(fit2)
fit2summary$r.squared
```

```
## [1] 0.5130333
```

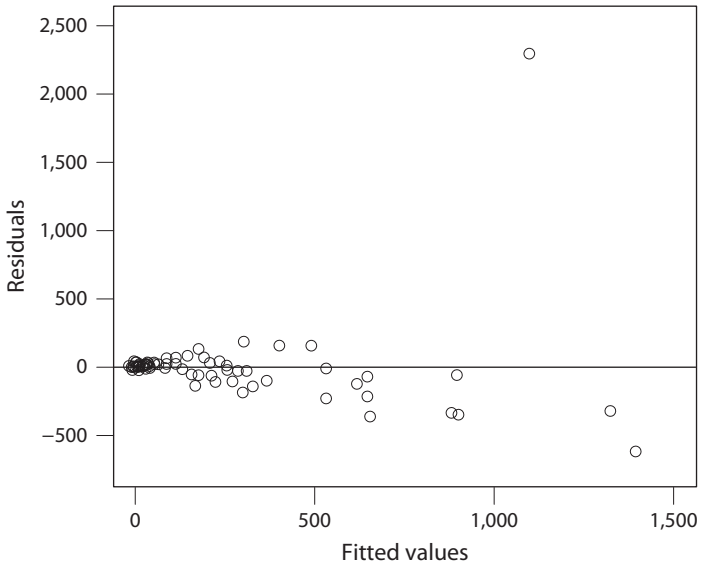
The resulting coefficient of determination appears relatively low given that we are predicting votes for a candidate from the same party using the previous election result. Earlier, we saw that Obama's vote shares at the state level are strongly correlated between the 2008 and 2012 elections. We can compute R^2 for that regression where the corresponding output object is `fit1`, which represents the output of the state-level regression. The coefficient of determination for the Florida regression proves to be much lower than that for the state-level regression.

```
R2(fit1)
```

```
## [1] 0.9671579
```

Given this unusually poor performance, it is useful to more closely inspect the residuals from the Florida regression. To do this, we create a *residual plot* where residuals are plotted against fitted values.

```
plot(fitted(fit2), resid(fit2), xlim = c(0, 1500), ylim = c(-750, 2500),
     xlab = "Fitted values", ylab = "Residuals")
abline(h = 0)
```



We observe an extremely large residual or *outlier*, where in the 2000 election, Buchanan received 2000 votes, substantially more than expected. The next line of code shows that this observation represents Palm Beach county. This can be seen by extracting the county name whose residual equals the maximum value of residuals.

```
florida$county[resid(fit2) == max(resid(fit2))]

## [1] PalmBeach
## 67 Levels: Alachua Baker Bay Bradford Brevard ... Washington
```

It turns out that in Palm Beach county, the so-called *butterfly ballot* was used for this election. A picture of this ballot is shown in figure 4.5. Voters are supposed to punch a hole that corresponds to the candidate they would like to vote for. However, as can be seen in the picture, the ballot is quite confusing. It appears that many supporters of Al Gore in this county mistakenly voted for Buchanan by punching the second hole from the top instead of the third hole. As mentioned at the beginning of the chapter, in the 2000 election, George Bush was elected to office by winning Florida with a razor-thin margin of 537 votes even though Gore won over half a million votes more than Bush in the entire country. It is widely believed that voting irregularities in Palm Beach county, as evident in the residual plot, cost Gore the presidency.

We now fit the same model without Palm Beach county. Later, we will see whether this removal improves the model fit, by comparing residual plots and regression lines with Palm Beach against those without it. We begin by computing the coefficient of determination without Palm Beach.

```
## data without Palm Beach
florida.pb <- subset(florida, subset = (county != "PalmBeach"))
```

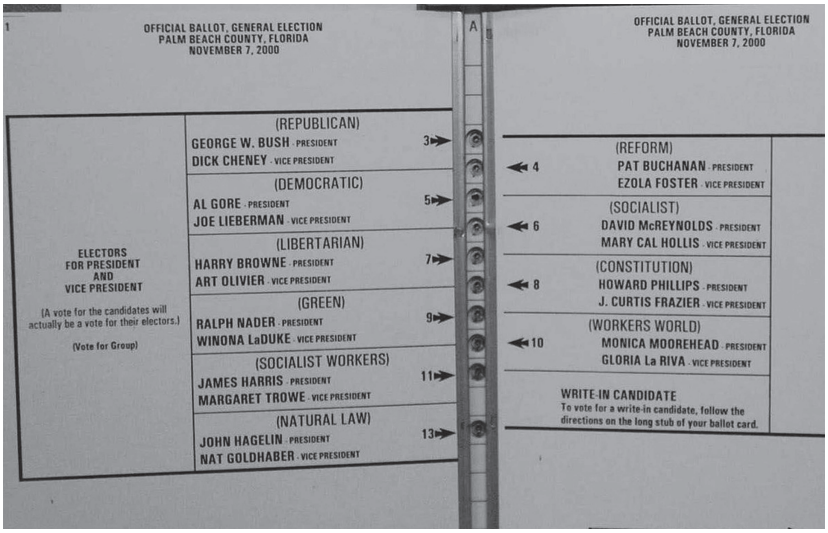


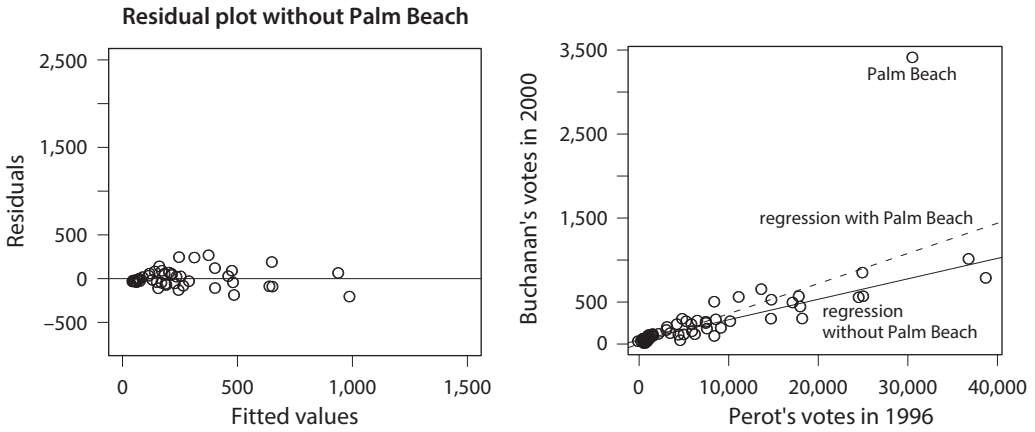
Figure 4.5. Butterfly Ballot in Palm Beach County.

```
fit3 <- lm(Buchanan00 ~ Perot96, data = florida.pb)
fit3

##
## Call:
## lm(formula = Buchanan00 ~ Perot96, data = florida.pb)
##
## Coefficients:
## (Intercept)      Perot96
##    45.84193      0.02435

## R-squared or coefficient of determination
R2(fit3)

## [1] 0.8511675
```

Without Palm Beach, the coefficient of determination dramatically increases from 0.51 to 0.85. The improvement in model fit can also be easily seen through the residual plot as well as the scatter plot with regression lines. We find that the regression line is influenced by Palm Beach—removing it shifts the regression line considerably. The new regression line fits the remaining observations better.

```
## residual plot
plot(fitted(fit3), resid(fit3), xlim = c(0, 1500), ylim = c(-750, 2500),
     xlab = "Fitted values", ylab = "Residuals",
     main = "Residual plot without Palm Beach")
abline(h = 0) # horizontal line at 0
```

```
plot(florida$Perot96, florida$Buchanan00, xlab = "Perot's votes in 1996",
     ylab = "Buchanan's votes in 2000")
abline(fit2, lty = "dashed") # regression with Palm Beach
abline(fit3) # regression without Palm Beach
text(30000, 3250, "Palm Beach")
text(30000, 1500, "regression\n with Palm Beach")
text(30000, 400, "regression\n without Palm Beach")
```



Finally, it is important to emphasize that the model fit considered in this section is based on *in-sample predictions* rather than *out-of-sample predictions*. That is, model fit statistics, such as the coefficient of determination, describe how well one's model fits the sample at hand. If tailored too closely to a particular sample, which is called *overfitting*, the model may make less accurate predictions in another sample. In cases where we seek a general model that can be applied to other data, we need to be careful to avoid overfitting the model to a particular sample. In section 4.3.2, we will describe one way to adjust R^2 in order to reduce the possibility of overfitting.

4.3 Regression and Causation

Regression is a primary tool for making predictions in social science research. How can regression be used to draw causal inference? As we discussed in chapter 2, causal inference requires the prediction of counterfactual outcomes. For example, for units who received a treatment, we wish to predict the values of the outcome variable that would result without the treatment. Under certain assumptions, regression models can be used to predict counterfactual outcomes. We must be careful, however, because association, which can be quantified through regression, does not necessarily imply causation.

Table 4.7. Women as Policy Makers Data.

Variable	*Description*
`GP`	identifier for the Gram Panchayat (GP)
`village`	identifier for each village
`reserved`	binary variable indicating whether the GP was reserved for women leaders or not
`female`	binary variable indicating whether the GP had a female leader or not
`irrigation`	variable measuring the number of new or repaired irrigation facilities in the village since the reserve policy started
`water`	variable measuring the number of new or repaired drinking water facilities in the village since the reservation policy started

4.3.1 RANDOMIZED EXPERIMENTS

Our running example is a study that examines the causal effects of having female politicians in government on policy outcomes.[5] Do women promote different policies than men? To answer this question, it is not sufficient to simply compare policy outcomes between districts that elect some female politicians and those that elect only male politicians. This is because these two types of districts may differ in terms of many factors other than having female politicians. For example, if liberal districts may be more likely to elect female politicians, it is not clear whether policy differences can be attributed to ideology or politician's gender.

To overcome this potential confounding problem, the authors of the study took advantage of a randomized policy experiment in India where, since the mid-1990s, one-third of village council heads have been randomly reserved for female politicians. The CSV data set `women.csv` contains a subset of this data from West Bengal. The policy was implemented at the level of government called Gram Panchayat or GP. Each GP contains many villages. For this study, two villages were selected at random within each GP for detailed data collection. Table 4.7 shows the names and descriptions of the variables in this data set. Each observation in the data set represents a village and there are two villages associated with each GP.

We first check whether or not the reservation policy was properly implemented by computing the proportions of female politicians elected for the reserved seats as well as the unreserved ones. Since each GP has the same number of villages, we can simply compute the average across villages without creating a new data set at the GP level. For the reserved seats, this proportion should be equal to 1.

[5] This section is based on Raghabendra Chattopadhyay and Esther Duflo (2004) "Women as policy makers: Evidence from a randomized policy experiment in India." *Econometrica*, vol. 72, no. 5, pp. 1409–1443.

```
women <- read.csv("women.csv")
## proportion of female politicians in reserved GP vs. unreserved GP
mean(women$female[women$reserved == 1])
## [1] 1
mean(women$female[women$reserved == 0])
## [1] 0.07476636
```

It appears that the reservation policy has been followed. Every GP that was supposed to reserve a council position for women actually elected at least one female politician. In contrast, 93% of the GPs to which the reservation policy was not applicable had no female representative. Following what we learned in chapter 2, we can compare the mean policy outcomes between the villages in the reserved GPs and those in the unreserved GPs. We hypothesize that female politicians are more likely to support policies that female voters want. The researchers found that more women complain about the quality of drinking water than men, who more frequently complain about irrigation. We estimate the average causal effects of the reservation policy on the number of new or repaired irrigation systems and drinking water facilities in the villages since the policy was implemented. We use the difference-in-means estimator as in section 2.4.

```
## drinking water facilities
mean(women$water[women$reserved == 1]) -
    mean(women$water[women$reserved == 0])
## [1] 9.252423
## irrigation facilities
mean(women$irrigation[women$reserved == 1]) -
    mean(women$irrigation[women$reserved == 0])
## [1] -0.3693319
```

We find that the reservation policy increased the number of drinking water facilities in a GP on average by about 9 (new or repaired), whereas the policy had little effect on irrigation systems. This finding is consistent with the aforementioned hypothesis that female politicians tend to represent the interests of female voters.

How can we use regression to analyze the data from randomized experiments like this one? It turns out that regressing an outcome variable on a treatment variable yields a slope coefficient identical to the difference in average outcomes between the two groups. In addition, the resulting intercept corresponds to the average outcome among the control units. More generally, when the predictor X is binary, taking a value of either 0 or 1, the linear model defined in equation (4.1) yields the estimated coefficients

of the following expressions:

$$\hat{\alpha} = \underbrace{\frac{1}{n_0}\sum_{i=1}^{n}(1-X_i)Y_i}_{\text{mean outcome among the control}},$$

$$\hat{\beta} = \underbrace{\frac{1}{n_1}\sum_{i=1}^{n}X_iY_i}_{\text{mean outcome among the treated}} - \underbrace{\frac{1}{n_0}\sum_{i=1}^{n}(1-X_i)Y_i}_{\text{mean outcome among the control}}.$$

In this equation, $n_1 = \sum_{i=1}^{n} X_i$ is the size of the treatment group and $n_0 = n - n_1$ is the size of the control group. Thus, $\hat{\beta}$ can be interpreted as the estimated average treatment effect.

Using our experimental data, we confirm this numerical equivalence between regression coefficients and average outcomes. That is, we observe that the estimated slope coefficient is equal to the corresponding *difference-in-means estimator*.

```
lm(water ~ reserved, data = women)

##
## Call:
## lm(formula = water ~ reserved, data = women)
##
## Coefficients:
## (Intercept)     reserved
##      14.738        9.252

lm(irrigation ~ reserved, data = women)

##
## Call:
## lm(formula = irrigation ~ reserved, data = women)
##
## Coefficients:
## (Intercept)     reserved
##      3.3879      -0.3693
```

We can directly connect the potential outcomes covered in chapter 2 to the regression model:

$$Y(X) = \alpha + \beta X + \epsilon.$$

Since the regression model predicts the average outcome given a value of the predictor, the estimated average treatment effect equals the estimated slope coefficient when X is binary. Recall that $\hat{\beta}$ represents the estimated change in Y when X is increased by one unit. Then, we have $\widehat{Y(1)} - \widehat{Y(0)} = (\hat{\alpha} + \hat{\beta}) - \hat{\alpha} = \hat{\beta}$, while the estimated average outcome for the control group is equal to the estimated intercept, i.e., $\widehat{Y(0)} = \hat{\alpha}$. Thus,

the linear regression model provides an alternative, but numerically equivalent, way to analyze experimental data in this setting.

When applied to experimental data with a single, binary treatment, the estimated slope coefficient of the linear regression model can be interpreted as an estimate of average treatment effect and is numerically equivalent to the **difference-in-means estimator**. The estimated intercept, on the other hand, is equal to the estimated average outcome under the control condition. The randomization of treatment assignment permits this **causal interpretation** of association identified under a linear regression model.

4.3.2 REGRESSION WITH MULTIPLE PREDICTORS

So far, we have included only one predictor in the linear regression model. However, a regression model can have more than one predictor. In general, a linear regression model with multiple predictors is defined as

$$Y = \alpha + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p + \epsilon.$$

In this model, α is the intercept, β_j is the coefficient for predictor X_j, ϵ is an error term, and p is the number of predictors and can be greater than 1. The interpretation of each coefficient β_j is the amount of change in the outcome variable associated with a one-unit increase in the corresponding predictor X_j *when all other predictors are held constant* or so-called *ceteris paribus*. Therefore, linear regression with multiple predictors enables researchers to assess the impact of each predictor.

The least squares method, as described in section 4.2.3, can be used to estimate the model parameters. That is, we choose the values of $(\hat{\alpha}, \hat{\beta}_1, \ldots, \hat{\beta}_p)$ such that the *sum of squared residuals* (SSR) is minimized. The SSR is defined as

$$\text{SSR} = \sum_{i=1}^{n} \hat{\epsilon}_i^2 = \sum_{i=1}^{n} (Y_i - \hat{\alpha} - \hat{\beta}_1 X_{i1} - \hat{\beta}_2 X_{i2} - \cdots - \hat{\beta}_p X_{ip})^2.$$

In the equation, $\hat{\epsilon}_i$ is the *residual* and X_{ij} is the value of the jth predictor for the ith observation. Recall that the residual is defined as the difference between the observed response Y and its predicted or fitted value $\widehat{Y} = \hat{\alpha} + \hat{\beta}_1 X_1 + \hat{\beta}_2 X_2 + \cdots + \hat{\beta} X_p$.

The validity of predictions based on a linear regression model critically rests on the assumption of linearity. The method of least squares always gives us the line that "best fits" the data in the sense of minimizing the SSR. However, this does not necessarily mean that the linear model is appropriate. While a comprehensive treatment of testing and relaxing this assumption is beyond the scope of this book, we must not forget that any model or method requires an assumption, and linear regression is no exception.

As an example of linear regression models with multiple predictors, we consider the randomized experiment on social pressure and turnout introduced in section 2.4.2. In that study, registered voters were randomly assigned to one of the four groups. We can fit a linear regression model, in which group assignment is used to predict turnout. Fitting the linear regression model is done via the `lm()` function as before.

One can add more than one predictor by simply using the + operator, for example, `lm(y ~ x1 + x2 + x3)`. In this example, since the `messages` variable is a factor, the `lm()` function automatically creates a set of *indicator* or dummy variables, each of which is equal to `1` if a voter is assigned to the corresponding group. These indicator variables will be used for computation but will not be saved in the data frame. The model includes all but the variable corresponding to the base level. The base level of a factor variable is the level displayed first when we apply the `levels()` function, which lists levels in alphabetical order. The other values of a factor variable are defined in relation to this base level value.

```
social <- read.csv("social.csv")
levels(social$messages) # base level is "Civic Duty"
## [1] "Civic Duty" "Control"    "Hawthorne"   "Neighbors"
```

Now we fit the linear regression model using this factor variable.

```
fit <- lm(primary2008 ~ messages, data = social)
fit
##
## Call:
## lm(formula = primary2008 ~ messages, data = social)
##
## Coefficients:
##       (Intercept)    messagesControl    messagesHawthorne
##          0.314538          -0.017899             0.007837
## messagesNeighbors
##          0.063411
```

Alternatively, one can create an indicator variable for each group and then specify the regression model using them. The results are identical to those given above.

```
## create indicator variables
social$Control <- ifelse(social$messages == "Control", 1, 0)
social$Hawthorne <- ifelse(social$messages == "Hawthorne", 1, 0)
social$Neighbors <- ifelse(social$messages == "Neighbors", 1, 0)
## fit the same regression as above by directly using indicator variables
lm(primary2008 ~ Control + Hawthorne + Neighbors, data = social)
```

Mathematically, the linear regression model we just fit is given by

$$Y = \alpha + \beta_1 \,\mathsf{Control} + \beta_2 \,\mathsf{Hawthorne} + \beta_3 \,\mathsf{Neighbors} + \epsilon.$$

In this model, each predictor is an indicator variable for the corresponding group. Since the base level of the `messages` variable is `"Civic Duty"`, the `lm()` function

excludes the corresponding indicator variable. Using the fitted model, we can predict the average outcome, which in this case is the average proportion of voters who turned out. For example, under the `Control` condition, the average outcome is predicted to be $\hat{\alpha} + \hat{\beta}_1 = 0.315 + (-0.018) = 0.297$ or 29.7%. Similarly, for the `Neighbors` group, the predicted average outcome is $\hat{\alpha} + \hat{\beta}_3 = 0.315 + 0.063 = 0.378$.

The predicted average outcome can be obtained using the `predict()` function. This function, like the `fitted()` function, takes the output from the `lm()` function and computes predicted values. However, unlike the `fitted()` function, which computes predicted values for the sample used to fit the model, the `predict()` function can take a new data frame as the `newdata` argument and make predictions for each observation in this data frame. The new data frame's variables must match the predictors of the fitted linear model, though they can have different values. In the current application, we create a new data frame using the `data.frame()` function. The resulting data frame contains the same variable `messages` as the predictor of the model but only four observations, each of which has one of the unique values of the original `messages` variable. We use the `unique()` function to extract these unique values and return them in the order of their first occurrence.

```
## create a data frame with unique values of "messages"
unique.messages <- data.frame(messages = unique(social$messages))
unique.messages
```

```
##     messages
## 1 Civic Duty
## 2  Hawthorne
## 3    Control
## 4  Neighbors
```

```
## make prediction for each observation from this new data frame
predict(fit, newdata = unique.messages)
```

```
##         1         2         3         4
## 0.3145377 0.3223746 0.2966383 0.3779482
```

As we saw in the case of a linear regression model with a single, binary predictor (see section 4.3.1), the predicted average outcome for each treatment condition equals the sample average within the corresponding subset of the data.

```
## sample average
tapply(social$primary2008, social$messages, mean)
```

```
## Civic Duty    Control  Hawthorne  Neighbors
##  0.3145377  0.2966383  0.3223746  0.3779482
```

To make the output of linear regression more interpretable, we can remove an intercept and use all four indicator variables (rather than removing the indicator variable for the base level in order to include a common intercept). This alternative

specification enables us to directly obtain the average outcome within each group as a coefficient for the corresponding indicator variable. To omit the intercept in linear regression, we simply use `-1` in the formula. The following code chunk illustrates this.

```
## linear regression without intercept
fit.noint <- lm(primary2008 ~ -1 + messages, data = social)
fit.noint

##
## Call:
## lm(formula = primary2008 ~ -1 + messages, data = social)
##
## Coefficients:
## messagesCivic Duty     messagesControl   messagesHawthorne
##             0.3145              0.2966              0.3224
##  messagesNeighbors
##             0.3779
```

Each coefficient above represents the average outcome for a given group. As a result, we can estimate an average treatment effect relative to the control for each treatment condition (`Civic Duty`, `Hawthorne`, or `Neighbors`) by calculating that treatment condition's coefficient minus the coefficient for the control group, which is the baseline group under this model with no intercept. The difference in the estimated causal effects between any two groups equals the difference between the corresponding coefficients, whether one uses the model with no intercept or the original model. Therefore, the average effect of the `Neighbors` treatment (relative to the `Control` condition) equals $0.378 - 0.297$ in the model with no intercept, or $0.063 - (-0.018)$ in the original model, either of which equals 0.081 or 8.1 percentage points. As was the case before, the same estimate of average causal effect can be obtained in two ways—through linear regression with a factor treatment variable or the difference-in-means estimator.

```
## estimated average effect of "Neighbors" condition
coef(fit)["messagesNeighbors"] - coef(fit)["messagesControl"]

## messagesNeighbors
##        0.08130991

## difference-in-means
mean(social$primary2008[social$messages == "Neighbors"]) -
    mean(social$primary2008[social$messages == "Control"])

## [1] 0.08130991
```

Finally, we can compute the *coefficient of determination* or R^2 as in section 4.2.6. When there are multiple predictors, however, we often compute the *adjusted* R^2 with the so-called *degrees of freedom* correction that accounts for the number of predictors.

Roughly speaking, the degrees of freedom refers to the number of observations that are "free to vary," which is often represented by the total number of observations minus the number of parameters to be estimated. In the current setting, the degrees of freedom equals $n - p - 1 = n - (p + 1)$ because n is the number of observations and $p + 1$ is the number of coefficients to be estimated, i.e., a coefficient for each of p predictors plus an intercept.

Since one can always increase the (unadjusted) R^2 by including an additional predictor (which always decreases SSR), the degrees of freedom correction adjusts R^2 downwards as more predictors are included in the model. The formula of the adjusted R^2 is given by

$$\text{adjusted } R^2 = 1 - \frac{\text{SSR}/(n - p - 1)}{\text{TSS}/(n - 1)}.$$

SSR is divided by the number of observations n minus the number of coefficients to be estimated $(p + 1)$. TSS is divided by $(n - 1)$ since TSS estimates only one parameter, the mean of the outcome variable or $\overline{Y}$. As in section 4.2.6, we create a function that computes the adjusted R^2.

```
## adjusted R-squared
adjR2 <- function(fit) {
    resid <- resid(fit) # residuals
    y <- fitted(fit) + resid # outcome
    n <- length(y)
    TSS.adj <- sum((y - mean(y))^2) / (n - 1)
    SSR.adj <- sum(resid^2) / (n - length(coef(fit)))
    R2.adj <- 1 - SSR.adj / TSS.adj
    return(R2.adj)
}
adjR2(fit)
```

```
## [1] 0.003272788
```

```
R2(fit) # unadjusted R-squared calculation
```

```
## [1] 0.003282564
```

In this case, the difference between unadjusted and adjusted R^2 is small because the number of observations is large relative to the number of coefficients. Alternatively, we can obtain both adjusted and unadjusted R^2 by applying the `summary()` function to output from the `lm()` function (see also section 7.3).

```
fitsummary <- summary(fit)
fitsummary$adj.r.squared
```

```
## [1] 0.003272788
```

The **linear regression model with multiple predictors** is defined as

$$Y = \alpha + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p + \epsilon,$$

where the coefficient β_j represents the increase in the average outcome associated with a one-unit increase in X_j while holding the other variables constant. The coefficients are estimated by minimizing the sum of squared residuals. The **degrees of freedom** adjustment is often made when computing the coefficient of determination.

4.3.3 HETEROGENOUS TREATMENT EFFECTS

When applied to randomized experiments, linear regression with multiple predictors can also be helpful for exploring *heterogenous treatment effects*. Even if the average treatment effect is positive, for example, the same treatment may affect some individuals in a negative way. Identifying the characteristics associated with the direction and magnitude of the treatment effect is essential in determining who should receive the treatment. In the current application, we might hypothesize that the social pressure treatment would barely affect those who vote infrequently. In contrast, they may be the ones who would be most affected by such treatment. To illustrate the analysis of heterogenous treatment effects, we examine the difference in the estimated average causal effect of the `Neighbors` message between those who voted in the 2004 primary election and those who did not. We can do this by subsetting the data and then estimating the average treatment effect within each subset. Finally, we compare these two estimated average treatment effects.

```
## average treatment effect (ATE) among those who voted in 2004 primary
social.voter <- subset(social, primary2004 == 1)
ate.voter <-
    mean(social.voter$primary2008[social.voter$messages == "Neighbors"]) -
        mean(social.voter$primary2008[social.voter$messages == "Control"])
ate.voter

## [1] 0.09652525

## average effect among those who did not vote
social.nonvoter <- subset(social, primary2004 == 0)
ate.nonvoter <-
    mean(social.nonvoter$primary2008[social.nonvoter$messages == "Neighbors"]) -
        mean(social.nonvoter$primary2008[social.nonvoter$messages == "Control"])
ate.nonvoter

## [1] 0.06929617
```

```
## difference
ate.voter - ate.nonvoter

## [1] 0.02722908
```

We find that those who voted in the 2004 primary election have the estimated average effect of 9.7 percentage points, which is approximately 2.7 percentage points greater than those who did not vote in the election. This implies that the `Neighbors` message affects those who voted in the 2004 primary election more than those who did not.

The same analysis can be carried out through the use of linear regression with an *interaction effect* between the treatment variable `Neighbors` and the covariate of interest `primary2004`. In our application, the model is given by

$$Y = \alpha + \beta_1 \,\text{primary2004} + \beta_2 \,\text{Neighbors} + \beta_3 \,(\text{primary2004} \times \text{Neighbors}) + \epsilon. \tag{4.9}$$

The final predictor is the product of two indicator variables, primary2004 × Neighbors, which is equal to 1 if and only if an individual voted in the 2004 primary election (`primary2004 = 1`) and received the `Neighbors` treatment (`Neighbors = 1`).

Thus, according to the model, among the voters who turned out in the 2004 primary election (`primary2004 = 1`), the average effect of the `Neighbors` message equals $\beta_2 + \beta_3$, whereas the same effect for those who did not vote in the 2004 election (`primary2004 = 0`) equals β_2. Thus, the coefficient for the interaction term β_3 represents the additional average treatment effect the first group of voters receive relative to the second group.

More generally, an example of the linear regression model with an interaction term is

$$Y = \alpha + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_1 X_2 + \epsilon,$$

where the coefficient for the interaction term β_3 represents how the effect of X_1 depends on X_2 (or vice versa). To see this, set $X_2 = x_2$ and then compute the predicted value when $X_1 = x_1$. This is given by $\hat{\alpha} + \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2 + \hat{\beta}_3 x_1 x_2$. Now, compare this with the predicted value when X_1 is increased by one unit, i.e., $X_1 = x_1 + 1$. Under this scenario, the predicted value is $\hat{\alpha} + \hat{\beta}_1(x_1 + 1) + \hat{\beta}_2 x_2 + \hat{\beta}_3(x_1 + 1)x_2$. Then, subtracting the previous predicted value from this one, we obtain the following expression for how the change in the average outcome associated with a one-unit increase in X_1 depends on the value of X_2:

$$\hat{\beta}_1 + \hat{\beta}_3 x_2.$$

This is another linear equation. The intercept β_1 represents the increase in the average outcome associated with a one-unit increase in X_1 when $X_2 = 0$. Then, each one-unit increase in X_2 has the effect of further increasing X_1 by the slope $\hat{\beta}_3$.

An example of a linear regression model with an **interaction term** is

$$Y = \alpha + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_1 X_2 + \epsilon.$$

The model assumes that the effect of X_1 linearly depends on X_2. That is, as we increase X_2 by one unit, the change in the average outcome associated with a one-unit increase of X_1 goes up by β_3.

In R, an interaction term can be represented by a colon `:` with the syntax `x1:x2` producing an interaction term between the two variables `x1` and `x2`. We illustrate the use of interaction terms by focusing on the `Neighbors` and `Control` groups.

```
## subset Neighbors and Control groups
social.neighbor <- subset(social, (messages == "Control") |
                                  (messages == "Neighbors"))
## standard way to generate main and interaction effects
fit.int <- lm(primary2008 ~ primary2004 + messages + primary2004:messages,
              data = social.neighbor)
fit.int

##
## Call:
## lm(formula = primary2008 ~ primary2004 + messages + primary2004:messages,
##     data = social.neighbor)
##
## Coefficients:
##                     (Intercept)
##                         0.23711
##                     primary2004
##                         0.14870
##              messagesNeighbors
##                         0.06930
## primary2004:messagesNeighbors
##                         0.02723
```

Since the `Control` group is the baseline condition, the slope coefficients are estimated only for the `Neighbors` condition and its interaction with the `primary2004` variable.

Alternatively, an asterisk `*` generates two *main effect* terms as well as one interaction effect term. That is, the syntax `x1*x2` produces `x1`, `x2`, and `x1:x2`. In most applications, one should include the corresponding main effects when the model has an interaction term. The same regression model as above can be fitted using the following syntax.

```
lm(primary2008 ~ primary2004 * messages, data = social.neighbor)
```

To interpret each estimated coefficient, it is again helpful to consider the predicted average outcome. Among those who voted in the 2004 primary election, the estimated average effect of the `Neighbors` treatment can be written as the difference in the estimated average outcome between the treatment and control groups. In terms of model parameters, this difference is equal to $(\hat{\alpha} + \hat{\beta}_1 + \hat{\beta}_2 + \hat{\beta}_3) - (\hat{\alpha} + \hat{\beta}_1) = \hat{\beta}_2 + \hat{\beta}_3$, where $\hat{\beta}_2$ and $\hat{\beta}_3$ are excluded from the second part of the equation because for the control group, `Neighbors` equals `0`. In contrast, the estimated average treatment effect among those who did not vote is given by $(\hat{\alpha} + \hat{\beta}_2) - \hat{\alpha} = \hat{\beta}_2$. Thus, the difference in the estimated average treatment effect between those who voted in the 2004 primary election and those who did not equals the estimated coefficient for the interaction effect term, i.e., $(\hat{\beta}_2 + \hat{\beta}_3) - \hat{\beta}_2 = \hat{\beta}_3$. This implies that the coefficient for the interaction effect term β_3 characterizes how the average treatment effect varies as a function of the covariate.

While we have so far focused on a factor or categorical variable, it is also possible to use a continuous variable as a predictor. The use of continuous variables requires a stronger linearity assumption that a one-unit increase in the predictor leads to an increase of the same size in the outcome, regardless of the baseline value. In the current application, we consider the age of the voter in 2008 as a predictor. We first compute this variable by subtracting the year of birth variable from the year of election.

```
social.neighbor$age <- 2008 - social.neighbor$yearofbirth
summary(social.neighbor$age)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   22.00   43.00   52.00   51.82   61.00  108.00
```

Thus, in this subset of the data, the ages of voters vary from 22 to 108. We now explore how the average causal effect of the `Neighbors` treatment changes as a function of age. To do this, we use the `age` variable instead of the `primary2004` variable in the linear regression model given in equation (4.9):

$$Y = \alpha + \beta_1 \,\text{age} + \beta_2 \,\text{Neighbors} + \beta_3 \,(\text{age} \times \text{Neighbors}) + \epsilon.$$

We can use the same computation strategy as above to understand how the average treatment effect changes as a function of age. Consider a group of voters who are x years old. The estimated average treatment effect of the `Neighbors` message for these voters is given by $(\hat{\alpha} + \hat{\beta}_1 x + \hat{\beta}_2 + \hat{\beta}_3 x) - (\hat{\alpha} + \hat{\beta}_1 x) = \hat{\beta}_2 + \hat{\beta}_3 x$. In contrast, among the voters who are $(x + 1)$ years old, the estimated average effect is $\{\hat{\alpha} + \hat{\beta}_1(x + 1) + \hat{\beta}_2 + \hat{\beta}_3(x + 1)\} - \{\hat{\alpha} + \hat{\beta}_1(x + 1)\} = \hat{\beta}_2 + \hat{\beta}_3(x + 1)$. Thus, the estimated coefficient for the interaction effect term $\hat{\beta}_3 = \{\hat{\beta}_2 + \hat{\beta}_3(x+1)\} - (\hat{\beta}_2 + \hat{\beta}_3 x)$ represents the estimated difference in the average treatment effect between two groups of voters whose ages differ by one year.

To compute this estimated difference in R, we first fit the linear regression model with the interaction term between the `age` and `Neighbors` variables. We use the syntax `age * messages`, which produces the main terms and the interaction term.

```
fit.age <- lm(primary2008 ~ age * messages, data = social.neighbor)
fit.age

##
## Call:
## lm(formula = primary2008 ~ age * messages, data = social.neighbor)
##
## Coefficients:
##           (Intercept)                    age
##             0.0894768              0.0039982
##     messagesNeighbors  age:messagesNeighbors
##             0.0485728              0.0006283
```

The result suggests that the estimated difference in the average treatment effect between two groups of voters whose ages differ by one year is equal to 0.06 percentage points. Based on this regression model, we can also compute the estimated average treatment effect for different ages. We choose 25, 45, 65, and 85 years old for illustration. We use the `predict()` function by providing the `newdata` argument with a data frame that contains these ages as separate observations.

```
## age = 25, 45, 65, 85 in Neighbors group
age.neighbor <- data.frame(age = seq(from = 25, to = 85, by = 20),
                           messages = "Neighbors")
## age = 25, 45, 65, 85 in Control group
age.control <- data.frame(age = seq(from = 25, to = 85, by = 20),
                          messages = "Control")
## average treatment effect for age = 25, 45, 65, 85
ate.age <- predict(fit.age, newdata = age.neighbor) -
    predict(fit.age, newdata = age.control)
ate.age

##          1          2          3          4
## 0.06428051 0.07684667 0.08941283 0.10197899
```

Researchers have found that the linearity assumption is inappropriate when modeling turnout. While people become more likely to vote as they get older, their likelihood of voting starts decreasing in their 60s or 70s. One common strategy to address this phenomenon is to model turnout as a *quadratic function* of age by including the square of age as an additional predictor. Consider the following model, which also includes

interaction terms:

$$Y = \alpha + \beta_1 \text{ age} + \beta_2 \text{ age}^2 + \beta_3 \text{ Neighbors} + \beta_4 (\text{age} \times \text{Neighbors}) + \beta_5 (\text{age}^2 \times \text{Neighbors}) + \epsilon. \quad (4.10)$$

In R, a formula can contain mathematical functions such as a square or natural logarithm using the `I()` function. For example, to include a square of the `x` variable in a formula, we can use the syntax `I(x^2)`. The `I()` function enables other arithmetic operations such as `I(sqrt(x))` and `I(log(x))`. We now fit the model specified in equation (4.10).

```
fit.age2 <- lm(primary2008 ~ age + I(age^2) + messages + age:messages +
                 I(age^2):messages, data = social.neighbor)
fit.age2
```

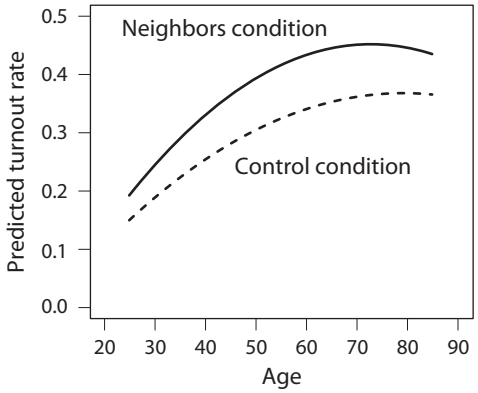
```
##
## Call:
## lm(formula = primary2008 ~ age + I(age^2) + messages + age:messages +
##     I(age^2):messages, data = social.neighbor)
##
## Coefficients:
##               (Intercept)                        age
##                -9.700e-02                  1.172e-02
##                  I(age^2)          messagesNeighbors
##                -7.389e-05                 -5.275e-02
##    age:messagesNeighbors  I(age^2):messagesNeighbors
##                 4.804e-03                 -3.961e-05
```

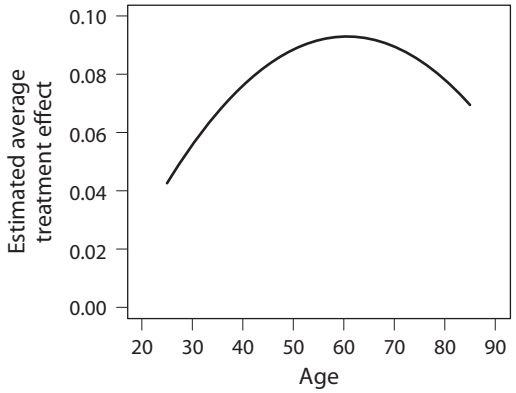
In a complicated model like this one, the coefficients no longer have an easy interpretation. In such situations, it is best to predict the average outcome under various scenarios using the `predict()` function and then compute quantities of interest. Here, we predict the average turnout rate for voters of different ages, ranging from 25 to 85, under the `Neighbors` and `Control` conditions. We then compute the average treatment effect as the difference between the two conditions and characterize it as a function of age. The following syntax accomplishes this task.

```
## predicted turnout rate under the Neighbors treatment condition
yT.hat <- predict(fit.age2,
                  newdata = data.frame(age = 25:85, messages = "Neighbors"))
## predicted turnout rate under the Control condition
yC.hat <- predict(fit.age2,
                  newdata = data.frame(age = 25:85, messages = "Control"))
```

For ease of interpretation, we plot the results. The first plot displays the predicted turnout as a function of age separately for the `Neighbors` and `Control` groups. The second plot shows the estimated average treatment effect as a function of age.

```
## plotting the predicted turnout rate under each condition
plot(x = 25:85, y = yT.hat, type = "l", xlim = c(20, 90), ylim = c(0, 0.5),
     xlab = "Age", ylab = "Predicted turnout rate")
lines(x = 25:85, y = yC.hat, lty = "dashed")
text(40, 0.45, "Neighbors condition")
text(45, 0.15, "Control condition")
## plotting the average treatment effect as a function of age
plot(x = 25:85, y = yT.hat - yC.hat, type = "l", xlim = c(20, 90),
     ylim = c(0, 0.1), xlab = "Age",
     ylab = "Estimated average\n treatment effect")
```





We find that according to this model, the estimated average treatment effect peaks around 60 years old, and the effect size is much smaller among young and old voters.

4.3.4 REGRESSION DISCONTINUITY DESIGN

The discussion in chapter 2 implies that we can interpret the association between treatment and outcome variables as causal if there is no confounding variable. This was the case in the experimental studies we analyzed in sections 4.3.1–4.3.3. In observational studies, however, the treatment assignment is not randomized. As a result, confounding factors, rather than the treatment variable, may explain the outcome difference between the treatment and control groups. In section 2.5, we discussed several research design strategies to address this potential selection bias problem. Here, we introduce another research design for observational studies called *regression discontinuity design* (RD design).

Table 4.8. Members of the British Parliament Personal Wealth Data

Variable	*Description*
`surname`	surname of the candidate
`firstname`	first name of the candidate
`party`	party of the candidate (`labour` or `tory`)
`ln.gross`	log gross wealth at the time of death
`ln.net`	log net wealth at the time of death
`yob`	year of birth of the candidate
`yod`	year of death of the candidate
`margin.pre`	margin of the candidate's party in the previous election
`region`	electoral region
`margin`	margin of victory (vote share)

As an application of RD design, we consider how much politicians can increase their personal wealth due to holding office. Scholars investigated this question by analyzing members of Parliament (MPs) in the United Kingdom.[6] The authors of the original study collected information about personal wealth at the time of death for several hundred competitive candidates who ran for office in general elections between 1950 and 1970. The data are contained in the CSV file `MPs.csv`. The names and descriptions of the variables in this data set appear in table 4.8.

A naive comparison of MPs and non-MPs in terms of their wealth is unlikely to yield valid causal inference because those who became MPs differ from those who did not in terms of many observable and unobservable characteristics. Instead, the key intuition behind RD design is to compare those candidates who narrowly won office with those who barely lost it. The idea is that when one's margin of victory switches from a negative number to a positive number, we would expect a large, discontinuous, positive jump in the personal wealth of electoral candidates if serving in office actually financially benefits them. Assuming that nothing else is going on at this point of discontinuity, we can identify the average causal effect of being an MP at this threshold by comparing the candidates who barely won the election with those who barely lost it. Regression is used to predict the average personal wealth at the point of discontinuity.

A simple scatter plot with regression lines is the best way to understand RD design. To do this, we plot the outcome variable, log net wealth at the time of death, against the margin of victory. We take the natural logarithmic transformation of wealth because this variable is quite skewed by a small number of politicians accumulating a large amount of wealth (see the discussion in section 3.4.1). We then separately fit a linear regression model to the observations with a positive margin (i.e., the candidates who won elections and became MPs) and another regression model to those with a negative margin (the candidates who lost). The difference in predicted values at the point of

[6] This application is based on Andrew C. Eggers and Jens Hainmueller (2009) "MPs for sale? Returns to office in postwar British politics." *American Political Science Review*, vol. 103, no. 4, pp. 513–533.

discontinuity, i.e., a zero margin of victory, between the two regressions represents the average causal effect on personal wealth of serving as an MP.

We begin by subsetting the data based on party (Labour and Tory) and then fit two regressions for each data set.

```
## load the data and subset them into two parties
MPs <- read.csv("MPs.csv")
MPs.labour <- subset(MPs, subset = (party == "labour"))
MPs.tory <- subset(MPs, subset = (party == "tory"))
## two regressions for Labour: negative and positive margin
labour.fit1 <- lm(ln.net ~ margin,
                  data = MPs.labour[MPs.labour$margin < 0, ])
labour.fit2 <- lm(ln.net ~ margin,
                  data = MPs.labour[MPs.labour$margin > 0, ])
## two regressions for Tory: negative and positive margin
tory.fit1 <- lm(ln.net ~ margin, data = MPs.tory[MPs.tory$margin < 0, ])
tory.fit2 <- lm(ln.net ~ margin, data = MPs.tory[MPs.tory$margin > 0, ])
```

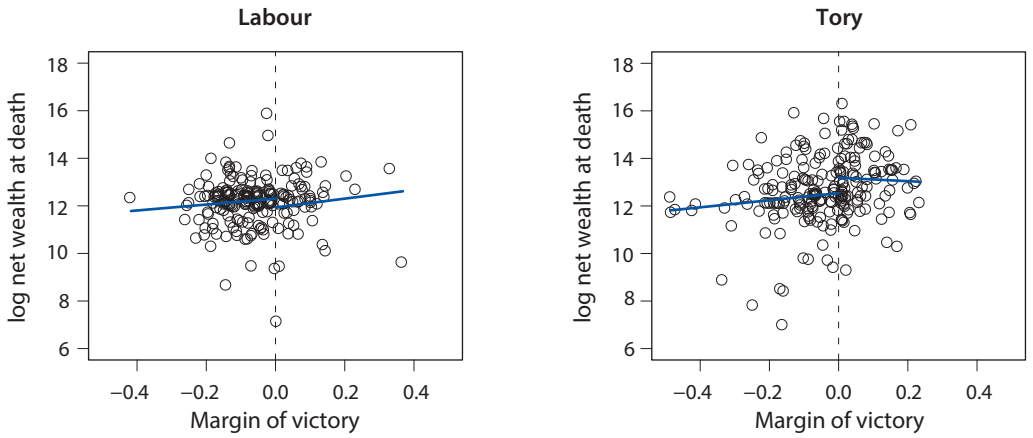
To predict the outcome using a specific value of predictor, we can use the `predict()` function by specifying a new data frame, `newdata`, as the argument. We conduct a separate analysis for Labour and Tory candidates to estimate each party's causal effect of interest.

```
## Labour: range of predictions
y1l.range <- c(min(MPs.labour$margin), 0) # min to 0
y2l.range <- c(0, max(MPs.labour$margin)) # 0 to max
## prediction
y1.labour <- predict(labour.fit1, newdata = data.frame(margin = y1l.range))
y2.labour <- predict(labour.fit2, newdata = data.frame(margin = y2l.range))
## Tory: range of predictions
y1t.range <- c(min(MPs.tory$margin), 0) # min to 0
y2t.range <- c(0, max(MPs.tory$margin)) # 0 to max
## predict outcome
y1.tory <- predict(tory.fit1, newdata = data.frame(margin = y1t.range))
y2.tory <- predict(tory.fit2, newdata = data.frame(margin = y2t.range))
```

We can now plot the predicted values for each party in the scatter plot of log net wealth and electoral margin.

```
## scatter plot with regression lines for Labour
plot(MPs.labour$margin, MPs.labour$ln.net, main = "Labour",
     xlim = c(-0.5, 0.5), ylim = c(6, 18), xlab = "Margin of victory",
     ylab = "log net wealth at death")
```

```
abline(v = 0, lty = "dashed")
## add regression lines
lines(y1l.range, y1.labour, col = "blue")
lines(y2l.range, y2.labour, col = "blue")
## scatter plot with regression lines for Tory
plot(MPs.tory$margin, MPs.tory$ln.net, main = "Tory", xlim = c(-0.5, 0.5),
     ylim = c(6, 18), xlab = "Margin of victory",
     ylab = "log net wealth at death")
abline(v = 0, lty = "dashed")
## add regression lines
lines(y1t.range, y1.tory, col = "blue")
lines(y2t.range, y2.tory, col = "blue")
```



The result suggests that Tory MPs financially benefit from serving in office whereas Labour MPs do not. How large is the effect for Tory candidates? We can numerically compute the differences in prediction at the zero margin and put them back on the original scale (pounds) since net wealth is measured on a log scale. Recall from section 3.4.1 that the *inverse function* of the natural logarithm is the exponential function, given by `exp()` in R.

```
## average net wealth for Tory MP
tory.MP <- exp(y2.tory[1])
tory.MP
##         1
## 533813.5
## average net wealth for Tory non-MP
tory.nonMP <- exp(y1.tory[2])
```

```
tory.nonMP
```

```
##        2
## 278762.5
```

```
## causal effect in pounds
tory.MP - tory.nonMP
```

```
##        1
## 255050.9
```

The estimated effect of being an MP on the personal wealth of Tory candidates is a little above 250,000 pounds. Since the average net wealth for Tory non-MPs is predicted to be a little above 270,000 pounds, the estimated effect is quite substantial. Being an MP almost doubles one's net wealth at death.

How should one examine the *internal validity* of regression discontinuity design? One way is a *placebo test*. A placebo test finds a case where the effect is theoretically known to be zero and then shows that the estimated effect is indeed close to zero. The name comes from the fact that in a medical study a placebo is supposed to have zero effect on health outcomes (though much evidence suggests that a placebo often has effects, perhaps via psychological mechanisms). In the current application, we estimate the average treatment effect on the margin of victory for the same party in the *previous* election. Since being an MP in the future should not affect the past election result, this effect should be zero if the RD design is valid. If the estimated effect is far from zero, on the other hand, it would suggest a possible violation of the assumption of regression discontinuity. For example, the incumbent party may be engaged in election fraud in order to win close elections.

```
## two regressions for Tory: negative and positive margin
tory.fit3 <- lm(margin.pre ~ margin, data = MPs.tory[MPs.tory$margin < 0, ])
tory.fit4 <- lm(margin.pre ~ margin, data = MPs.tory[MPs.tory$margin > 0, ])
## the difference between two intercepts is the estimated effect
coef(tory.fit4)[1] - coef(tory.fit3)[1]
```

```
## (Intercept)
## -0.01725578
```

The estimated effect on the previous margin of victory is less than 2 percentage points. This small effect size gives empirical support for the claim that RD design is applicable to this study. In chapter 7, we will more formally answer the question of how small is small enough to reach this conclusion.

While RD design can overcome the main difficulty of observational studies, i.e., potential confounding bias, this strength of *internal validity* comes at the cost of *external validity*. Specifically, the estimated causal effects obtained under this design apply only to the observations near the point of discontinuity. In our application, these observations represent candidates who narrowly won or lost elections. The degree to which MPs benefit financially from serving in office may be quite different for those

who won elections by a larger margin. Thus, although RD requires weaker assumptions than other approaches, the resulting estimates may not be generalizable to a larger population of interest.

Regression discontinuity design (RD design) is a research design strategy for causal inference in observational studies with possible confounding factors. RD design assumes that the change in outcome at the point of discontinuity can be attributed to the change in the treatment variable alone. While RD design often has strong internal validity, it may lack external validity because the result may not be generalizable to observations away from the point of discontinuity.

4.4 Summary

We began this chapter with a discussion of election forecasting. We showed that preelection polls can be used to obtain relatively accurate, though not perfect, **predictions** of election outcomes in the context of US presidential elections. We introduced **prediction error** and explained how the accuracy of prediction can be measured using statistics such as **bias** and the **root-mean-squared error**. We also discussed the problem of **classification**, which is the prediction of categorical outcomes. Two types of **misclassification** are possible—false positives and false negatives. For example, a voter who did turn out being classified as a nonvoter would be a false negative, whereas a voter who did not turn out being classified as a voter would be a false positive. There is a clear trade-off between the two: minimizing false positives tends to increase false negatives and vice versa.

We then introduced a **linear regression model** as a commonly used method to predict an outcome variable of interest using another variable. The model enables researchers to predict an outcome variable based on the values of explanatory variables or predictors. Predictions based on the linear regression model are typically obtained through the **method of least squares** by minimizing the sum of squared prediction errors. We discussed the exact relationship between linear regression and **correlation**, and the phenomenon called **regression towards the mean**. Finally, we presented several ways to assess model fit through the examination of the **coefficient of determination** and residuals. It is important to avoid overfitting one's model to the data at hand so that the model does not capture any idiosyncratic characteristics of the sample and instead identifies the systematic features of the data-generating process.

Despite our intuition, association discovered through regression does not necessarily imply **causation**. A regression's ability to predict observable outcomes does not necessarily entail ability to predict counterfactual outcomes. Yet, valid causal inference requires the latter. At the end of the chapter, we discussed the use of regression in the analysis of experimental and observational data. We discussed how to estimate heterogenous treatment effects using the linear regression model with **interaction terms**. We also discussed the **regression discontinuity design**. By exploiting the discontinuity in the treatment assignment mechanism, this design enables researchers

Table 4.9. Intrade Prediction Market Data from 2008 and 2012.

Variable	*Description*
`day`	date of the session
`statename`	full name of each state (including District of Columbia in 2008)
`state`	abbreviation of each state (including District of Columbia in 2008)
`PriceD`	closing price (predicted vote share) of the Democratic nominee's market
`PriceR`	closing price (predicted vote share) of the Republican nominee's market
`VolumeD`	total session trades of the Democratic Party nominee's market
`VolumeR`	total session trades of the Republican Party nominee's market

to credibly identify causal effects in observational studies. The main disadvantage of the regression discontinuity design, however, is the potential lack of **external validity**. Specifically, the empirical conclusions based on this design may not be applicable beyond the observations close to the discontinuity threshold.

4.5 Exercises

4.5.1 PREDICTION BASED ON BETTING MARKETS

Earlier in the chapter, we studied the prediction of election outcomes using polls. Here, we study the prediction of election outcomes based on betting markets. In particular, we analyze data for the 2008 and 2012 US presidential elections from the online betting company called Intrade. At Intrade, people trade contracts such as "Obama to win the electoral votes of Florida." Each contract's market price fluctuates based on its sales. Why might we expect betting markets like Intrade to accurately predict the outcomes of elections or of other events? Some argue that the market can aggregate available information efficiently. In this exercise, we will test this *efficient market hypothesis* by analyzing the market prices of contracts for Democratic and Republican nominees' victories in each state.

The data files for 2008 and 2012 are available in CSV format as `intrade08.csv` and `intrade12.csv`, respectively. Table 4.9 presents the names and descriptions of these data sets. Each row of the data sets represents daily trading information about the contracts for either the Democratic or Republican Party nominee's victory in a particular state. We will also use the election outcome data. These data files are `pres08.csv` (table 4.1) and `pres12.csv` (table 4.5).

1. We will begin by using the market prices on the day before the election to predict the 2008 election outcome. To do this, subset the data such that it contains the market information for each state and candidate on the day before the election only. Note that in 2008, Election Day was November 4. We compare the closing prices for the two candidates in a given state and classify a candidate whose contract has a higher price as the predicted winner of that state. Which states

were misclassified? How does this compare to the classification by polls presented earlier in this chapter? Repeat the same analysis for the 2012 election, which was held on November 6. How well did the prediction market do in 2012 compared to 2008? Note that in 2012 some less competitive states have missing data on the day before the election because there were no trades on the Republican and Democratic betting markets. Assume Intrade predictions would have been accurate for these states.

2. How do the predictions based on the betting markets change over time? Implement the same classification procedure as above on each of the last 90 days of the 2008 campaign rather than just the day before the election. Plot the predicted number of electoral votes for the Democratic Party nominee over this 90-day period. The resulting plot should also indicate the actual election result. Note that in 2008, Obama won 365 electoral votes. Briefly comment on the plot.

3. Repeat the previous exercise but this time use the seven-day *moving-average* price, instead of the daily price, for each candidate within a state. Just as in section 4.1.3, this can be done with a loop. For a given day, we take the average of the Session Close prices within the past seven days (including that day). To answer this question, we must first compute the seven-day average within each state. Next, we sum the electoral votes for the states Obama is predicted to win. Using the `tapply()` function will allow us to efficiently compute the predicted winner for each state on a given day.

4. Create a similar plot for 2008 statewide poll predictions using the data file `polls08.csv` (see table 4.2). Notice that polls are not conducted daily within each state. Therefore, within a given state, for each of the last 90 days of the campaign, we compute the average margin of victory from the most recent poll(s) conducted. If multiple polls occurred on the same day, average these polls. Based on the most recent predictions in each state, sum Obama's total number of predicted electoral votes. One strategy to answer this question is to program two loops—an inner loop with 51 iterations (for each state) and an outer loop with 90 iterations (for each day).

5. What is the relationship between the price margins of the Intrade market and the actual margin of victory? Using the market data from the day before the election in 2008 only, regress Obama's actual margin of victory in each state on Obama's price margin from the Intrade markets. Similarly, in a separate analysis, regress Obama's actual margin of victory on Obama's predicted margin from the latest polls within each state. Interpret the results of these regressions.

6. Do the 2008 predictions of polls and Intrade accurately predict each state's 2012 elections results? Using the fitted regressions from the previous question, forecast Obama's actual margin of victory for the 2012 election in two ways. First, use the 2012 Intrade price margins from the day before the election as the predictor in each state. Recall that the 2012 Intrade data do not contain market prices for all

Table 4.10. 2012 US Presidential Election Polling Data.

Variable	*Description*
`state`	abbreviated name of the state in which the poll was conducted
`Obama`	predicted support for Obama (percentage)
`Romney`	predicted support for Romney (percentage)
`Pollster`	name of the organization conducting the poll
`middate`	middate of the period when the poll was conducted

states. Ignore states without data. Second, use the 2012 poll-predicted margins from the latest polls in each state as the predictor, found in `polls12.csv`. Table 4.10 presents the names and descriptions of the 2012 US presidential election polling data.

4.5.2 ELECTION AND CONDITIONAL CASH TRANSFER PROGRAM IN MEXICO

In this exercise, we analyze the data from a study that seeks to estimate the electoral impact of *Progresa*, Mexico's *conditional cash transfer program* (CCT program).[7] The original study relied on a randomized evaluation of the CCT program in which eligible villages were randomly assigned to receive the program either 21 months (early Progresa) or 6 months (late Progresa) before the 2000 Mexican presidential election. The author of the original study hypothesized that the CCT program would mobilize voters, leading to an increase in turnout and support for the incumbent party (PRI, or Partido Revolucionario Institucional, in this case). The analysis was based on a sample of precincts that contain at most one participating village in the evaluation.

The data we analyze are available as the CSV file `progresa.csv`. Table 4.11 presents the names and descriptions of variables in the data set. Each observation in the data represents a precinct, and for each precinct the file contains information about its treatment status, the outcomes of interest, socioeconomic indicators, and other precinct characteristics.

1. Estimate the impact of the CCT program on turnout and support for the incumbent party (PRI) by comparing the average electoral outcomes in the "treated" (early Progresa) precincts versus the ones observed in the "control" (late Progresa) precincts. Next, estimate these effects by regressing the outcome variable on the treatment variable. Interpret and compare the estimates under these approaches. Here, following the original analysis, use the turnout and support rates as shares of the eligible voting population (`t2000` and `pri2000s`, respectively). Do the results support the hypothesis? Provide a brief interpretation.

[7] This exercise is based on the following articles: Ana de la O (2013) "Do conditional cash transfers affect voting behavior? Evidence from a randomized experiment in Mexico." *American Journal of Political Science*, vol. 57, no. 1, pp. 1–14 and Kosuke Imai, Gary King, and Carlos Velasco (2015) "Do nonpartisan programmatic policies have partisan electoral effects? Evidence from two large scale randomized experiments." Working paper.

Table 4.11. Conditional Cash Transfer Program (Progresa) Data.

Variable	*Description*
`treatment`	whether an electoral precinct contains a village where households received early Progresa
`pri2000s`	PRI votes in the 2000 election as a share of precinct population above 18
`pri2000v`	official PRI vote share in the 2000 election
`t2000`	turnout in the 2000 election as a share of precinct population above 18
`t2000r`	official turnout in the 2000 election
`pri1994`	total PRI votes in the 1994 presidential election
`pan1994`	total PAN votes in the 1994 presidential election
`prd1994`	total PRD votes in the 1994 presidential election
`pri1994s`	total PRI votes in the 1994 election as a share of precinct population above 18
`pan1994s`	total PAN votes in the 1994 election as a share of precinct population above 18
`prd1994s`	total PRD votes in the 1994 election as a share of precinct population above 18
`pri1994v`	official PRI vote share in the 1994 election
`pan1994v`	official PAN vote share in the 1994 election
`prd1994v`	official PRD vote share in the 1994 election
`t1994`	turnout in the 1994 election as a share of precinct population above 18
`t1994r`	official turnout in the 1994 election
`votos1994`	total votes cast in the 1994 presidential election
`avgpoverty`	precinct average of village poverty index
`pobtot1994`	total population in the precinct
`villages`	number of villages in the precinct

2. In the original analysis, the author fits a linear regression model that includes, as predictors, a set of pretreatment covariates as well as the treatment variable. Here, we fit a similar model for each outcome that includes the average poverty level in a precinct (`avgpoverty`), the total precinct population in 1994 (`pobtot1994`), the total number of voters who turned out in the previous election (`votos1994`), and the total number of votes cast for each of the three main competing parties in the previous election (`pri1994` for PRI, `pan1994` for Partido Acción Nacional or PAN, and `prd1994` for Partido de la Revolución Democrática or PRD). Use the same outcome variables as in the original analysis, which are based on the shares of the voting age population. According to this model, what are the estimated average effects of the program's availability on turnout and support for the incumbent party? Are these results different from those you obtained in the previous question?

3. Next, we consider an alternative, and more natural, model specification. We will use the original outcome variables as in the previous question. However, our model should include the previous election outcome variables measured as shares of the voting age population (as done for the outcome variables `t1994`, `pri1994s`, `pan1994s`, and `prd1994s`) instead of those measured in counts. In addition, we apply the natural logarithmic transformation to the precinct population variable when including it as a predictor. As in the original model, our model includes the average poverty index as an additional predictor. Are the results based on these new model specifications different from those we obtained in the previous question? If the results are different, which model fits the data better?

4. We examine the balance of some pretreatment variables used in the previous analyses. Using box plots, compare the distributions of the precinct population (on the original scale), average poverty index, previous turnout rate (as a share of the voting age population), and previous PRI support rate (as a share of the voting age population) between the treatment and control groups. Comment on the patterns you observe.

5. We next use the official turnout rate `t2000r` (as a share of the registered voters) as the outcome variable rather than the turnout rate used in the original analysis (as a share of the voting age population). Similarly, we use the official PRI's vote share `pri2000v` (as a share of all votes cast) rather than the PRI's support rate (as a share of the voting age population). Compute the average treatment effect of the CCT program using a linear regression with the average poverty index, the log-transformed precinct population, and the previous official election outcome variables (`t1994r` for the previous turnout; `pri1994v`, `pan1994v`, and `prd1994v` for the previous PRI, PAN, and PRD vote shares). Briefly interpret the results.

6. So far we have focused on estimating the average treatment effects of the CCT program. However, these effects may vary from one precinct to another. One important dimension to consider is poverty. We may hypothesize that since individuals in precincts with higher levels of poverty are more receptive to cash transfers, they are more likely to turn out in the election and support the incumbent party when receiving the CCT program. Assess this possibility by examining how the average treatment effect of the policy varies by different levels of poverty for precincts. To do so, fit a linear regression with the following predictors: the treatment variable, the log-transformed precinct population, the average poverty index and its square, the interaction between the treatment and the poverty index, and the interaction between the treatment and the squared poverty index. Estimate the average effects for unique observed values and plot them as a function of the average poverty level. Comment on the resulting plot.

Table 4.12. Brazilian Government Transfer Data.

Variable	*Description*
`pop82`	population in 1982
`poverty80`	poverty rate of the state in 1980
`poverty91`	poverty rate of the state in 1991
`educ80`	average years in education of the state in 1980
`educ91`	average years in education of the state in 1991
`literate91`	literacy rate of the state in 1991
`state`	state
`region`	region
`id`	municipal ID
`year`	year of measurement

4.5.3 GOVERNMENT TRANSFER AND POVERTY REDUCTION IN BRAZIL

In this exercise, we estimate the effects of increased government spending on educational attainment, literacy, and poverty rates.[8] Some scholars argue that government spending accomplishes very little in environments of high corruption and inequality. Others suggest that in such environments, accountability pressures and the large demand for public goods will drive elites to respond. To address this debate, we exploit the fact that until 1991, the formula for government transfers to individual Brazilian municipalities was determined in part by the municipality's population. This meant that municipalities with populations below the official cutoff did not receive additional revenue, while states above the cutoff did. The data set `transfer.csv` contains the variables shown in table 4.12.

1. We will apply the regression discontinuity design to this application. State the required assumption for this design and interpret it in the context of this specific application. What would be a scenario in which this assumption is violated? What are the advantages and disadvantages of this design for this specific application?

2. Begin by creating a variable that determines how close each municipality was to the cutoff that determined whether states received a transfer or not. Transfers occurred at three separate population cutoffs: 10,188, 13,584, and 16,980. Using these cutoffs, create a single variable that characterizes the difference from the *closest* population cutoff. Following the original analysis, standardize this measure by dividing the difference by the corresponding cutoff, and multiplying it by 100. This will yield a normalized percentage score for the difference between the population of each state and the cutoff, relative to the cutoff value.

[8] This exercise is based on Stephan Litschig and Kevin M. Morrison (2013) "The impact of intergovernmental transfers on education outcomes and poverty reduction." *American Economic Journal: Applied Economics*, vol. 5, no. 4, pp. 206–240.

3. Begin by subsetting the data to include only those municipalities within 3 points of the funding cutoff on either side. Using regressions, estimate the average causal effect of government transfer on each of the three outcome variables of interest: educational attainment, literacy, and poverty. Give a brief substantive interpretation of the results.

4. Visualize the analysis performed in the previous question by plotting data points, fitted regression lines, and the population threshold. Briefly comment on the plot.

5. Instead of fitting linear regression models, we compute the difference-in-means of the outcome variables between the groups of observations above the threshold and below it. How do the estimates differ from what you obtained in question 3? Is the assumption invoked here identical to the one required for the analysis conducted in question 3? Which estimates are more appropriate? Discuss.

6. Repeat the analysis conducted in question 3 but vary the width of the analysis window from 1 to 5 percentage points below and above the threshold. Obtain the estimate for every percentage point. Briefly comment on the results.

7. Conduct the same analysis as in question 3 but this time using the measures of poverty rate and educational attainment taken in 1980, before the population-based government transfers began. What do the results suggest about the validity of the analysis presented in question 3?