6 Forecasting Numeric Data – Regression Methods

Mathematical relationships describe many aspects of everyday life. For example, a person's body weight can be described in terms of his or her calorie intake; one's income can be related to years of education and job experience; and the president's odds of being re-elected can be estimated by popular opinion poll numbers.

In each of these cases, numbers specify precisely how the data elements are related. An additional 250 kilocalories consumed daily is likely to result in nearly a kilogram of weight gain per month. Each year of job experience may be worth an additional $1,000 in yearly salary while years of education might be worth $2,500. A president is more likely to be re-elected with a high approval rating. Obviously, these types of equations do not perfectly model every case, but on average, the rules might work fairly well.

A large body of work in the field of statistics describes techniques for estimating such numeric relationships among data elements, a field of study known as regression analysis. These methods can be used for forecasting numeric data and quantifying the size and strength of a relationship between an outcome and its predictors.

By the end of this chapter, you will have learned how to apply regression methods to your own data. Along the way, you will learn:

- The basic statistical principles that linear regression methods use to fit equations to data, and how they describe relationships among data elements
- How to use R to prepare data for regression analysis, define a linear equation, and estimate the regression model
- How to use hybrid models known as regression trees and model trees, which allow decision trees to be used for numeric prediction

Until now, we have only looked at machine learning methods suitable for classification. The methods in this chapter will allow you to tackle an entirely new set of learning tasks. With that in mind, let's get started.

Understanding regression

Regression is concerned with specifying the relationship between a single numeric **dependent variable** (the value to be predicted) and one or more numeric **independent variables** (the predictors). We'll begin by assuming that the relationship between the independent and dependent variables follows a straight line.

> The origin of the term "regression" to describe the process of fitting lines to data is rooted in a study of genetics by *Sir Francis Galton* in the late 19th century. Galton discovered that fathers that were extremely short or extremely tall tended to have sons whose heights were closer to average. He called this phenomenon "regression to the mean".

You might recall from algebra that lines can be defined in a slope-intercept form similar to $y = a + bx$, where y is the dependent variable and x is the independent variable. In this formula, the **slope** b indicates how much the line rises for each increase in x. The variable a indicates the value of y when $x = 0$. It is known as the intercept because it specifies where the line crosses the vertical axis.

Regression equations model data using a similar slope-intercept format. The machine's job is to identify values of a and b such that the specified line is best able to relate the supplied x values to the values of y. It might not be a perfect match, so the machine should also have some way to quantify the margin of error. We'll discuss this in depth shortly.

Regression analysis is commonly used for modeling complex relationships among data elements, estimating the impact of a treatment on an outcome, and extrapolating into the future. Some specific use cases include:

- Examining how populations and individuals vary by their measured characteristics, for scientific research across fields as diverse as economics, sociology, psychology, physics, and ecology
- Quantifying the causal relationship between an event and the response, such as those in clinical drug trials, engineering safety tests, or marketing research
- Identifying patterns that can be used to forecast future behavior given known criteria, such as for predicting insurance claims, natural disaster damage, election results, and crime rates

Regression methods are also used for hypothesis testing, which involves determining whether data indicate that a presupposition is more likely to be true or false. The regression model's estimates of the strength and consistency of a relationship provide information that can be used to assess whether the findings are due to chance alone.

Because hypothesis testing is technically not a learning task, we will not cover it in depth. If you are interested in this topic, an introductory statistics textbook is a good place to get started.

Unlike the other machine learning methods we've covered thus far, regression analysis is not synonymous with a single algorithm. Rather, it is an umbrella for a large number of methods that can be adapted to nearly any machine learning task. If you were limited to choosing only a single analysis method, regression would be a good choice. You could devote an entire career to nothing else and perhaps still have much to learn.

In this chapter, we'll focus only on the most basic regression models—those that use straight lines. This is called **linear regression**. If there is only a single independent variable, this is known as **simple linear regression**, otherwise it is known as **multiple regression**. Both of these models assume that the dependent variable is continuous.

It is possible to use regression for other types of dependent variables and even for classification tasks. For instance, **logistic regression** can be used to model a binary categorical outcome, while **Poisson regression**—named after the French mathematician *Siméon Poisson*—models integer count data. The same basic principles apply to all regression methods, so once you understand the linear case, you can move on to the others.

Linear regression, logistic regression, Poisson regression, and many others fall in a class of models known as **generalized linear models** (**GLM**), which allow regression to be applied to many types of data. Linear models are generalized via the use of a **link function**, which specifies the mathematical relationship between x and y.

Despite the name, simple linear regression is not too simple to solve complex problems. In the next section, we'll see how the use of a simple linear regression model might have averted a tragic engineering disaster.

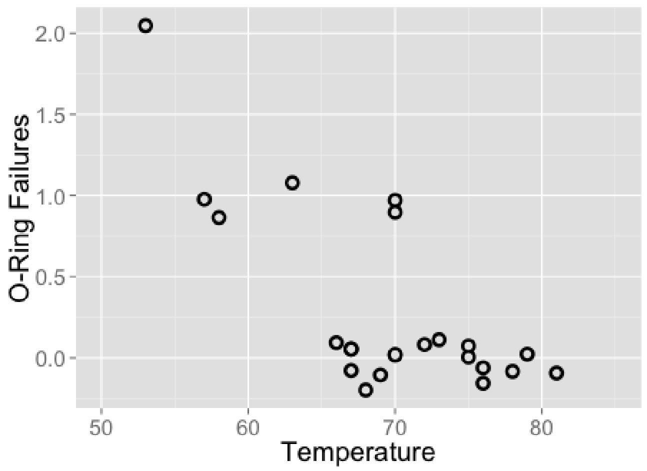
Simple linear regression

On January 28, 1986, seven crewmembers of the United States space shuttle Challenger were killed when O-rings responsible for sealing the joints of the rocket booster failed and caused a catastrophic explosion.

The night prior, there had been a lengthy discussion about how the low temperature forecast might affect the safety of the launch. The shuttle components had never been tested in such cold weather; therefore, it was unclear whether the equipment could withstand the strain from freezing temperatures. The rocket engineers believed that cold temperatures could make the components more brittle and less able to seal properly, which would result in a higher chance of a dangerous fuel leak. However, given the political pressure to continue with the launch, they needed data to support their hypothesis.

This section's analysis is based on data presented in *Risk analysis of the space shuttle: pre-Challenger prediction of failure*, *Journal of the American Statistical Association*, *Vol. 84*, pp. 945-957, by *S.R. Dalal*, *E.B. Fowlkes*, and *B. Hoadley*, (1989).

The scientists' discussion turned to data from 23 previous successful shuttle launches which recorded the number of O-ring failures versus the launch temperature. Since the shuttle has a total of six O-rings, each additional failure increases the odds of a catastrophic leak. The following scatterplot shows this data:



Examining the plot, there is an apparent trend between temperature and number of failures. Launches occurring at higher temperatures tend to have fewer O-ring failures. Additionally, the coldest launch (62 degrees F) had two rings fail, the most of any launch. The fact that the Challenger was scheduled to launch at a temperature about 30 degrees colder seems concerning. To put this risk in quantitative terms, we can turn to simple linear regression.

Simple linear regression defines the relationship between a dependent variable and a single independent predictor variable using a line denoted by an equation in the following form:

$$y = \alpha + \beta x$$

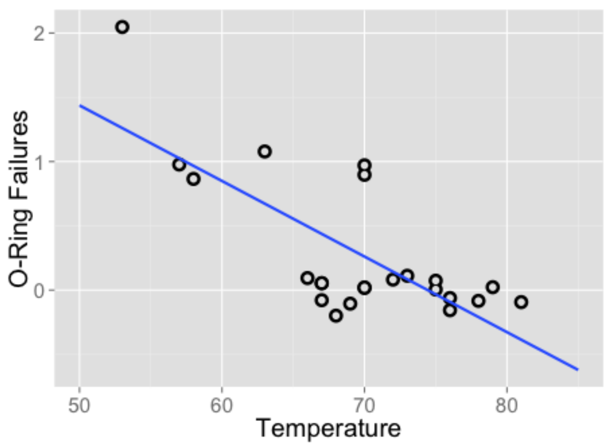
Don't be alarmed by the Greek characters; this equation can still be understood using the slope-intercept form described previously. The intercept, α (alpha), describes where the line crosses the y axis, while the slope, β (beta), describes the change in y given an increase of x. For the shuttle launch data, the slope would tell us the expected reduction in number of O-ring failures for each degree the launch temperature increases.

Greek characters are often used in the field of statistics to indicate variables that are parameters of a statistical function. Therefore, performing a regression analysis involves finding **parameter estimates** for α and β. The parameter estimates for alpha and beta are typically denoted using a and b, although you may find that some of this terminology and notation is used interchangeably.

Suppose we know that the estimated regression parameters in the equation for the shuttle launch data are:

- $a = 4.30$
- $b = -0.057$

Hence, the full linear equation is *y = 4.30 – 0.057x*. Ignoring for a moment how these numbers were obtained, we can plot the line on the scatterplot:

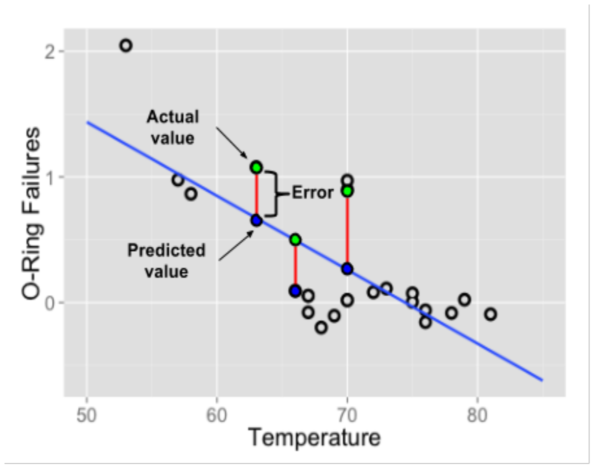


As the line shows, at 60 degrees Fahrenheit, we predict just under one O-ring failure. At 70 degrees Fahrenheit, we expect around 0.3 failures. If we extrapolate our model all the way out to 31 degrees—the forecasted temperature for the Challenger launch—we would expect about *4.30 – 0.057 * 31 = 2.53* O-ring failures. Assuming that each O-ring failure is equally likely to cause a catastrophic fuel leak, this means that the Challenger launch was about three times more risky than the typical launch at 60 degrees, and over eight times more risky than a launch at 70 degrees.

Notice that the line doesn't predict the data exactly. Instead, it cuts through the data somewhat evenly, with some predictions lower than expected and some higher. In the next section, we will learn about why this particular line was chosen.

Ordinary least squares estimation

In order to determine the optimal estimates of a and β, an estimation method known as **ordinary least squares (OLS)** was used. In OLS regression, the slope and intercept are chosen such that they minimize the sum of the squared errors, that is, the vertical distance between the predicted y value and the actual y value. These errors are known as **residuals**, and are illustrated for several points in the preceding diagram:



In mathematical terms, the goal of OLS regression can be expressed as the task of minimizing the following equation:

$$\sum\left(y_i - \hat{y}_i\right)^2 = \sum e_i^2$$

In plain language, this equation defines *e* (the error) as the difference between the actual *y* value and the predicted *y* value. The error values are squared and summed across all points in the data.

The caret character (^) above the *y* term is a commonly used feature of statistical notation. It indicates that the term is an estimate for the true *y* value. This is referred to as the *y*-hat.

Though the proof is beyond the scope of this book, it can be shown using calculus that the value of *b* that results in the minimum squared error is:

$$b = \frac{\sum\left(x_i - \bar{x}\right)\left(y_i - \bar{y}\right)}{\sum\left(x_i - \bar{x}\right)^2}$$

While the optimal value of *a* is:

$$a = \bar{y} - b\bar{x}$$

To understand these equations, you'll need to know another bit of statistical notation. The horizontal bar appearing over the *x* and *y* terms indicates the mean value of *x* or *y*. This is referred to as the *x*-bar or *y*-bar.

To understand these equations, we can break them into pieces. The denominator for *b* should look familiar; it is the same as the variance of *x*, which can be denoted as *Var(x)*. As we learned in *Chapter 2*, *Managing and Understanding Data*, calculating the variance involves finding the average squared deviation from the mean of *x*.

We have not computed the numerator before. This involves taking the sum of each data point's deviation from the mean *x* value multiplied by that point's deviation away from the mean *y* value. This is known as the **covariance** of *x* and *y*, denoted as *Cov(x, y)*. With this in mind, we can re-write the formula for *b* as:

$$b = \frac{Cov(x, y)}{Var(x)}$$

If you would like to follow along with these examples, download the `challenger.csv` file from the Packt Publishing's website and load to a data frame using the command `launch <- read.csv("challenger.csv")`.

Given this formula, it is easy to calculate the value of *b* using R functions. Assume that our shuttle launch data are stored in a data frame named `launch`, the independent variable *x* is `temperature`, and the dependent variable *y* is `distress_ct`. We can then use R's built-in `cov()` and `var()` functions to estimate *b*:

```
> b <- cov(launch$temperature, launch$distress_ct) /
         var(launch$temperature)
> b
[1] -0.05746032
```

From here, we can estimate *a* using the `mean()` function:

```
> a <- mean(launch$distress_ct) - b * mean(launch$temperature)
> a
[1] 4.301587
```

Estimating the regression equation in this way is not ideal, so R of course provides functions for doing this automatically. We will look at those shortly. First, we will expand our understanding of regression by learning a method for measuring the strength of a linear relationship and then see how linear regression can be applied to data having more than one independent variable.

Correlations

The **correlation** between two variables is a number that indicates how closely their relationship follows a straight line. Without additional qualification, correlation refers to Pearson's correlation coefficient, which was developed by the 20th century mathematician *Karl Pearson*. The correlation ranges between -1 and +1. The extreme values indicate a perfectly linear relationship, while a correlation close to zero indicates the absence of a linear relationship.

The following formula defines Pearson's correlation:

$$\rho_{x,y} = Corr\left(x, y\right) = \frac{Cov\left(x, y\right)}{\sigma_x \sigma_y}$$

Some more Greek notation has been introduced here: the first symbol (looks like a lowercase 'p') is *rho*, and it is used to denote the Pearson correlation statistic. The characters that look like 'q' turned sideways are *sigma*, and they indicate the standard deviation of *x* or *y*.

Using this formula, we can calculate the correlation between the launch temperature and the number of O-ring failures. Recall that the covariance function is `cov()` and the standard deviation function is `sd()`. We'll store the result in `r`, a letter that is commonly used to indicate the estimated correlation:

```
> r <- cov(launch$temperature, launch$distress_ct) /
         (sd(launch$temperature) * sd(launch$distress_ct))
> r
[1] -0.725671
```

Alternatively, we can use the built in correlation function, `cor()`:

```
> cor(launch$temperature, launch$distress_ct)
[1] -0.725671
```

Since the correlation is about -0.73, this implies that there is a fairly strong negative linear association between the temperature and the number of distressed O-rings. The negative association implies that an increase in temperature is correlated with fewer distressed O-rings. To the NASA engineers studying the O-ring data, this might have been a very clear indicator that a low-temperature launch could be problematic.

There are various rules-of-thumb used to interpret correlations. One method assigns a weak correlation to values between 0.1 and 0.3, moderate for 0.3 to 0.5, and strong for values above 0.5 (these also apply to similar ranges of negative correlations). However, these thresholds may be too lax for some purposes. Often, the correlation must be interpreted in context. For data involving human beings, a correlation of 0.5 may be considered extremely high; for data generated by mechanical processes, a correlation of 0.5 may be weak.

You have probably heard the expression "correlation does not imply causation". This is rooted in the fact that a correlation only describes the association between a pair of variables, yet there could be other explanations. For example, there may be a strong association between life expectancy and time per day spent watching movies, but before doctors start recommending that we all watch more movies, we need to rule out another explanation: older people watch fewer movies and are more likely to die.

Measuring the correlation between two variables gives us a way to quickly gauge relationships among independent variables and the dependent variable. This will be increasingly important as we start defining regression models with a larger number of predictors.

Multiple linear regression

Most real-world analyses have more than one independent variable. Therefore, it is likely that you will be using **multiple linear regression** most of the time you use regression for a numeric prediction task. The strengths and weaknesses of multiple linear regression are shown in the following table:

Strengths	Weaknesses
• By far the most common approach for modeling numeric data • Can be adapted to model almost any data • Provides estimates of the strength and size of the relationships among features and the outcome	• Makes strong assumptions about the data • The model's form must be specified by the user in advance • Does not do well with missing data • Only works with numeric features, so categorical data require extra processing • Requires some knowledge of statistics to understand the model

We can understand multiple regression as an extension of simple linear regression. The goal in both cases is similar: find values of beta coefficients that minimize the prediction error of a linear equation. The key difference is that there are additional terms for the additional independent variables.

Multiple regression equations generally follow the form of the following equation. The dependent variable y is specified as the sum of an intercept term plus the product of the estimated β value and the x value for each of i features. An error term (denoted by the Greek letter epsilon) has been added here as a reminder that the predictions are not perfect. This is the residual term noted previously.

$$y = \alpha + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_i x_i + \varepsilon$$

Let's consider for a moment the interpretation of the estimated regression parameters. You will note that in the preceding equation, a coefficient is estimated for each feature. This allows each feature to have a separate estimated effect on the value of y. In other words, y changes by the amount β_i for each unit increase in x_i. The intercept is then the expected value of y when the independent variables are all zero.

Since the intercept is really no different than any other regression parameter, it can also be denoted as β_0 (pronounced beta-naught) as shown in the following equation:

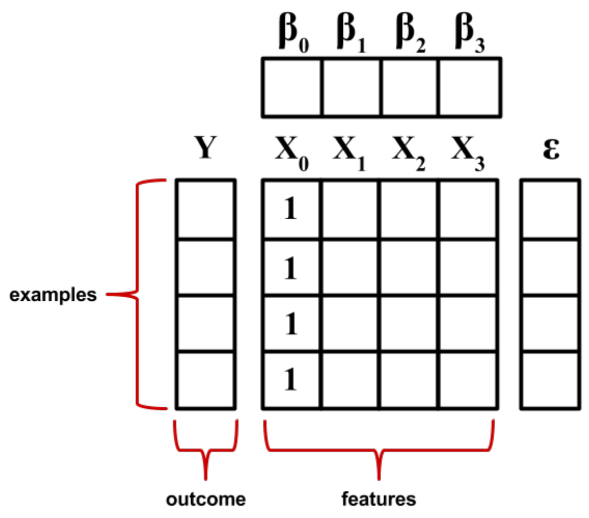
$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_i x_i + \varepsilon$$

This can be re-expressed using a condensed formulation:

$$Y = X\beta + \varepsilon$$

Even though this looks familiar, there are a few subtle changes. The dependent variable is now a vector, *Y*, with a row for every example. The independent variables have been combined into a matrix, *X*, with a column for each feature plus an additional column of '1' values for the intercept term. The regression coefficients β and errors ε are also now vectors.

The following figure illustrates these changes:



The goal now is to solve for the vector β that minimizes the sum of the squared errors between the predicted and actual *y* values. Finding the optimal solution requires the use of matrix algebra; therefore, the derivation deserves more careful attention than can be provided in this text. However, if you're willing to trust the work of others, the best estimate of the vector β can be computed as:

$$\hat{\beta} = \left(X^T X\right)^{-1} X^T Y$$

This solution uses a pair of matrix operations: the *T* indicates the transpose of matrix *X*, while the negative exponent indicates the matrix inverse. Using built-in R matrix operations, we can thus implement a simple multiple regression learner. Let's see if we can apply this formula to the Challenger launch data.

If you are unfamiliar with the preceding matrix operations, the Wikipedia pages for **transpose** and **matrix inverse** provide a thorough introduction and are quite understandable, even without a strong mathematics background.

Using the following code, we can create a simple regression function named `reg` which takes a parameter `y` and a parameter `x` and returns a matrix of estimated beta coefficients.

```
> reg <- function(y, x) {
    x <- as.matrix(x)
    x <- cbind(Intercept = 1, x)
    solve(t(x) %*% x) %*% t(x) %*% y
  }
```

This function uses several R commands we have not used previously. First, since we will be using the function with sets of columns from a data frame, the `as.matrix()` function is used to coerce the data into matrix form. Next, the `cbind()` function is used to bind an additional column onto the `x` matrix; the command `Intercept = 1` instructs R to name the new column `Intercept` and to fill the column with repeating `1` values. Finally, a number of matrix operations are performed on the `x` and `y` objects:

- `solve()` takes the inverse of a matrix
- `t()` is used to transpose a matrix
- `%*%` multiplies two matrices

By combining these as shown in the formula for the estimated beta vector, our function will return estimated parameters for the linear model relating `x` to `y`.

Let's apply our `reg()` function to the shuttle launch data. As shown in the following code, the data include four features and the outcome of interest, `distress_ct` (the number of O-ring failures):

```
> str(launch)
'data.frame':23 obs. of  5 variables:
 $ o_ring_ct  : int  6 6 6 6 6 6 6 6 6 6 ...
 $ distress_ct: int  0 1 0 0 0 0 0 0 1 1 ...
 $ temperature: int  66 70 69 68 67 72 73 70 57 63 ...
 $ pressure   : int  50 50 50 50 50 50 100 100 200 200 ...
 $ launch_id  : int  1 2 3 4 5 6 7 8 9 10 ...
```

We can confirm that our function is working correctly by comparing its result to the simple linear regression model of O-ring failures versus temperature, which we found earlier to have parameters `a = 4.30` and `b = -0.057`. Since `temperature` is the third column of the `launch` data, we can run the `reg()` function as follows:

```
> reg(y = launch$distress_ct, x = launch[3])
                   [,1]
Intercept    4.30158730
temperature -0.05746032
```

These values exactly match our prior result, so let's use the function to build a multiple regression model. We'll apply it just as before, but this time specifying three columns of data instead of just one:

```
> reg(y = launch$distress_ct, x = launch[3:5])
                     [,1]
Intercept    3.814247216
temperature -0.055068768
pressure     0.003428843
launch_id   -0.016734090
```

This model predicts the number of O-ring failures versus temperature, pressure, and the launch ID number. The negative coefficients for the temperature and launch ID variables suggests that as temperature or the launch ID increases, the number of expected O-ring failures decreases. Applying the same interpretation to the coefficient for pressure, we learn that as the pressure increases, the number of O-ring failures is expected to increase.

> Even if you are not a rocket scientist, these findings seem reasonable. Cold temperatures make the O-rings more brittle and higher pressure will likely increase the strain on the part. But why would launch ID be associated with fewer O-ring failures? One explanation is that perhaps later launches used O-rings composed from a stronger or more flexible material.

So far, we've only scratched the surface of linear regression modeling. Although our work was useful to help understand exactly how regression models are built, R's functions for fitting linear regression models are not only likely faster than ours, but also more informative. Real-world regression packages provide a wealth of output to aid model interpretation. Let's apply our knowledge of regression to a more challenging learning task.

Example – predicting medical expenses using linear regression

In order for an insurance company to make money, it needs to collect more in yearly premiums than it spends on medical care to its beneficiaries. As a result, insurers invest a great deal of time and money to develop models that accurately forecast medical expenses.

Medical expenses are difficult to estimate because the most costly conditions are rare and seemingly random. Still, some conditions are more prevalent for certain segments of the population. For instance, lung cancer is more likely among smokers than non-smokers, and heart disease may be more likely among the obese.

The goal of this analysis is to use patient data to estimate the average medical care expenses for such population segments. These estimates could be used to create actuarial tables which set the price of yearly premiums higher or lower depending on the expected treatment costs.

Step 1 – collecting data

For this analysis, we will use a simulated dataset containing medical expenses for patients in the United States. These data were created for this book using demographic statistics from the U.S. Census Bureau, and thus approximately reflect real-world conditions.

> If you would like to follow along interactively, download the `insurance.csv` file from the Packt Publishing's website and save it to your R working folder.

The `insurance.csv` file includes 1,338 examples of beneficiaries currently enrolled in the insurance plan, with features indicating characteristics of the patient as well as the total medical expenses charged to the plan for the calendar year. The features are:

- `age`: This is an integer indicating the age of the primary beneficiary (excluding those above 64 years, since they are generally covered by the government).
- `sex`: This is the policy holder's gender, either `male` or `female`.
- `bmi`: This is the **body mass index** (**BMI**), which provides a sense of how over or under-weight a person is relative to their height. BMI is equal to weight (in kilograms) divided by height (in meters) squared. An ideal BMI is within the range of 18.5 to 24.9.
- `children`: This is an integer indicating the number of children / dependents covered by the insurance plan.
- `smoker`: This is `yes` or `no` depending on whether the insured regularly smokes tobacco.
- `region`: This is the beneficiary's place of residence in the U.S., divided into four geographic regions: `northeast`, `southeast`, `southwest`, or `northwest`.

It is important to give some thought to how these variables may be related to billed medical expenses. For instance, we might expect that older people and smokers are at higher risk of large medical expenses. Unlike many other machine learning methods, in regression analysis, the relationships among the features are typically specified by the user rather than detected automatically. We'll explore some of these potential relationships in the next section.

Step 2 – exploring and preparing the data

As we have done before, we will use the `read.csv()` function to load the data for analysis. We can safely use `stringsAsFactors = TRUE` because it is appropriate to convert the three nominal variables to factors:

```
> insurance <- read.csv("insurance.csv", stringsAsFactors = TRUE)
```

The `str()` function confirms that the data are formatted as we had expected:

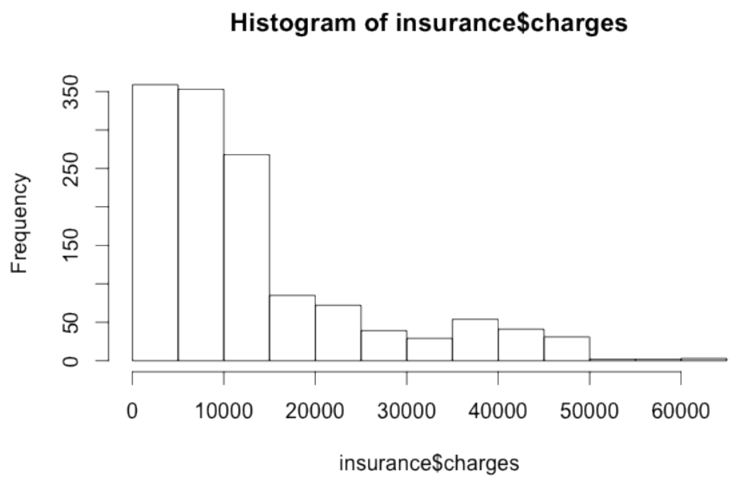
```
> str(insurance)
'data.frame':1338 obs. of  7 variables:
 $ age     : int  19 18 28 33 32 31 46 37 37 60 ...
 $ sex     : Factor w/ 2 levels "female","male": 1 2 2 2 2 1 ...
 $ bmi     : num  27.9 33.8 33 22.7 28.9 ...
 $ children: int  0 1 3 0 0 0 1 3 2 0 ...
 $ smoker  : Factor w/ 2 levels "no","yes": 2 1 1 1 1 1 ...
 $ region  : Factor w/ 4 levels "northeast","northwest", ...
 $ charges : num  16885 1726 4449 21984 3867 ...
```

Since the dependent variable is `charges`, let's take a look to see how it is distributed:

```
> summary(insurance$charges)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   1122    4740    9382   13270   16640   63770
```

Because the mean value is greater than the median, this implies that the distribution of insurance charges is right-skewed. We can confirm this visually using a histogram:

```
> hist(insurance$charges)
```



The large majority of individuals in our data have yearly medical expenses between zero and $15,000, although the tail of the distribution extends far past these peaks. Because linear regression assumes a normal distribution for the dependent variable, this distribution is not ideal. In practice, the assumptions of linear regression are often violated. If needed, we may be able to correct this later on.

Another problem at hand is that regression models require that every feature is numeric, yet we have three factor type in our data frame. We will see how R's linear regression function treats our variables shortly.

The `sex` variable is divided into `male` and `female` levels, while `smoker` is divided into `yes` and `no`. From the `summary()` output, we know that region has four levels, but we need to take a closer look to see how they are distributed.

```
> table(insurance$region)
northeast northwest southeast southwest
      324       325       364       325
```

Here, we see that the data have been divided nearly evenly among four geographic regions.

Exploring relationships among features – the correlation matrix

Before fitting a regression model to data, it can be useful to determine how the independent variables are related to the dependent variable and each other. A **correlation matrix** provides a quick overview of these relationships. Given a set of variables, it provides a correlation for each pairwise relationship.

To create a correlation matrix for the four numeric variables in the `insurance` data frame, use the `cor()` command:

```
> cor(insurance[c("age", "bmi", "children", "charges")])
               age       bmi   children    charges
age      1.0000000 0.1092719 0.04246900 0.29900819
bmi      0.1092719 1.0000000 0.01275890 0.19834097
children 0.0424690 0.0127589 1.00000000 0.06799823
charges  0.2990082 0.1983410 0.06799823 1.00000000
```

At the intersection of each row and column pair, the correlation is listed for the variables indicated by that row and column. The diagonal is always `1` since there is always a perfect correlation between a variable and itself. The values above and below the diagonal are identical since correlations are symmetrical. In other words, `cor(x, y)` is equal to `cor(y, x)`.

None of the correlations in the matrix are considered strong, but there are some notable associations. For instance, `age` and `bmi` appear to have a moderate correlation, meaning that as `age` increases, so does `bmi`. There is also a moderate correlation between `age` and `charges`, `bmi` and `charges`, and `children` and `charges`. We'll try to tease out these relationships more clearly when we build our final regression model.

Visualizing relationships among features – the scatterplot matrix

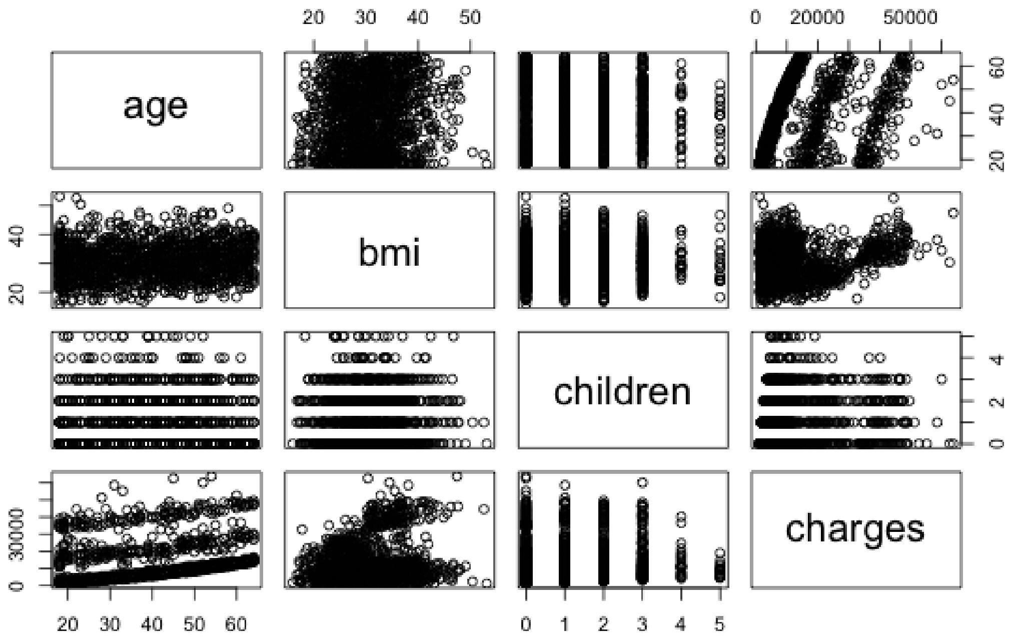
It can also be helpful to visualize the relationships among features, perhaps by using a scatterplot. Although we could create a scatterplot for each possible relationship, doing so for a large number of features might become tedious.

An alternative is to create a **scatterplot matrix** (sometimes abbreviated as **SPLOM**), which is simply a collection of scatterplots arranged in a grid. It is used to detect patterns among three or more variables. The scatterplot matrix is not a true multi-dimensional visualization because only two features are examined at a time. Still, it provides a general sense of how the data may be interrelated.

We can use R's graphical capabilities to create a scatterplot matrix for the four numeric features: `age`, `bmi`, `children`, and `charges`. The `pairs()` function is provided in a default R installation and provides basic functionality for producing scatterplot matrices. To invoke the function, simply provide it the data frame to present. Here, we'll limit the `insurance` data frame to the four numeric variables of interest:

```
> pairs(insurance[c("age", "bmi", "children", "charges")])
```

This produces the following diagram:



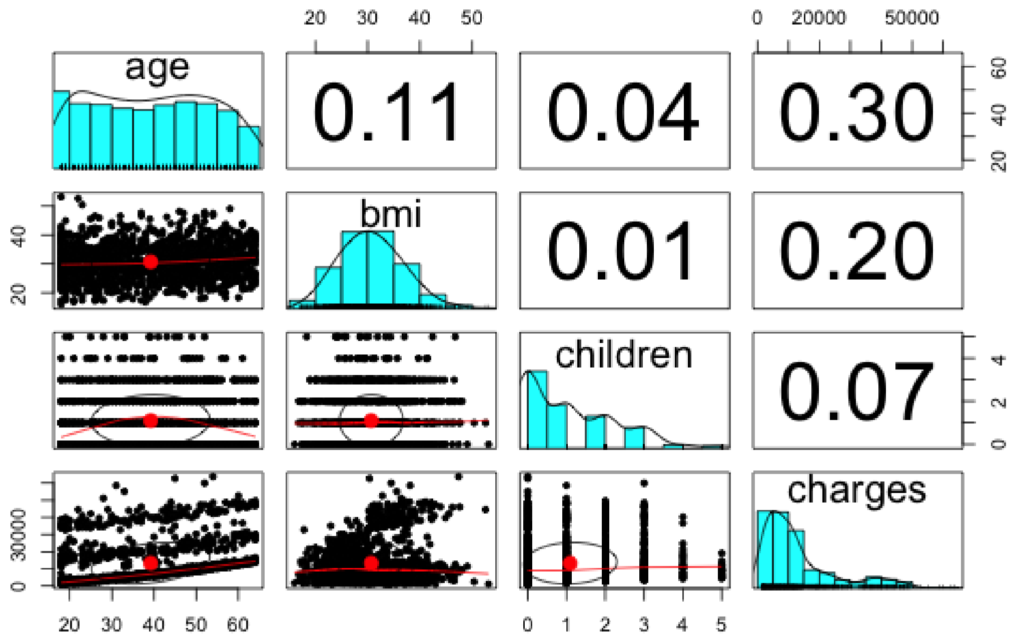
As with the correlation matrix, the intersection of each row and column holds the scatterplot of the variables indicated by the row and column pair. The diagrams above and below the diagonal are transpositions since the x axis and y axis have been swapped.

Do you notice any patterns in these plots? Although some look like random clouds of points, a few seem to display some trends. The relationship between `age` and `charges` displays several relatively straight lines, while `bmi` and `charges` has two distinct groups of points. It is difficult to detect trends in any of the other plots.

If we add more information to the plot, it can be even more useful. An enhanced scatterplot matrix can be created with the `pairs.panels()` function in the `psych` package. If you do not have this package installed, type `install packages("psych")` to install it on your system then load it using the `library(psych)` command. Then, we can create a scatterplot matrix as we had done previously:

```
> pairs.panels(insurance[c("age", "bmi", "children", "charges")])
```

This produces a slightly more informative scatterplot matrix, as follows:



Above the diagonal, the scatterplots have been replaced with a correlation matrix. On the diagonal, a histogram depicting the distribution of values for each feature is shown. Finally, the scatterplots below the diagonal now are presented with additional visual information.

The oval-shaped object on each scatterplot is a **correlation ellipse**. It provides a visualization of how strongly correlated the variables are. The dot at the center of the ellipse indicates the point of the mean value for the x axis variable and y axis variable. The correlation between the two variables is indicated by the shape of the ellipse; the more it is stretched, the stronger the correlation. An almost perfectly round oval, as with `bmi` and `children`, indicates a very weak correlation (in this case 0.01).

The curve drawn on the scatterplot is called a **loess smooth**. It indicates the general relationship between the x axis and y axis variables. It is best understood by example. The curve for `age` and `children` is an upside-down U, peaking around middle age. This means that the oldest and youngest people in the sample have fewer children than those around middle age. Because this trend is non-linear, this finding could not have been inferred from the correlations alone. On the other hand, the loess smooth for `age` and `bmi` is a line sloping gradually up, implying that BMI increases with age, but we had already inferred this from the correlation matrix.

Step 3 – training a model on the data

To fit a linear regression model to data with R, the `lm()` function can be used. This is included in the `stats` package, which should be included and loaded by default with your R installation. The `lm()` syntax is as follows:

Multiple regression modeling syntax

using the `lm()` function in the `stats` package

Building the model:

```
m <- lm(dv ~ iv, data = mydata)
```

- `dv` is the dependent variable in the `mydata` data frame to be modeled
- `iv` is an R formula specifying the independent variables in the `mydata` data frame to use in the model
- `data` specifies the data frame in which the `dv` and `iv` variables can be found

The function will return a regression model object that can be used to make predictions. Interactions between independent variables can be specified using the * operator.

Making predictions:

```
p <- predict(m, test)
```

- `m` is a model trained by the `lm()` function
- `test` is a data frame containing test data with the same features as the training data used to build the model.

The function will return a vector of predicted values.

Example:

```
ins_model <- lm(charges ~ age + children + sex + smoker,
                            data = insurance)
ins_pred <- predict(ins_model, insurance_test)
```

The following command fits a linear regression model called `ins_model`, which relates the six independent variables to the total medical charges. The R formula syntax uses the tilde character ~ to describe the model; the dependent variable `charges` goes to the left of the tilde while the independent variables go to the right, separated by the + sign. There is no need to specify the regression model's intercept term, as it is assumed by default:

```
> ins_model <- lm(charges ~ age + children + bmi + sex +
  smoker + region, data = insurance)
```

Because the `.` character can be used to specify all features (excluding those already specified in the formula), the following command is equivalent to the preceding command:

```
> ins_model <- lm(charges ~ ., data = insurance)
```

After building the model, simply type the name of the model object to see the estimated beta coefficients:

```
> ins_model3

Call:
lm(formula = charges ~ age + children + bmi + sex +
    smoker + region, data = insurance)

Coefficients:
    (Intercept)       age              children
       -11938.5       256.9            475.5

    bmi               sexmale          smokeryes
    339.2             -131.3           23848.5

  regionnorthwest  regionsoutheast  regionsouthwest
    -353.0            -1035.0            -960.1
```

Understanding the regression coefficients is fairly straightforward. The intercept tells us the value of `charges` when the independent variables are equal to zero.

As is the case here, quite often the intercept is difficult to interpret because it is impossible to have values of zero for all features. For example, since no person exists with age zero and BMI zero, the slope has no inherent meaning. For this reason, in practice, the intercept is often ignored.

The estimated beta coefficients indicate the increase in `charges` for an increase of one in each of the features when the other features are held constant. For instance, for each year that age increases, we would expect $256.90 higher medical expenses on average, assuming everything else is equal. Similarly, each additional child results in an average of $475.50 in additional medical expenses each year, and each unit of BMI increase is associated with an increase of $339.20 in yearly medical costs.

You might notice that although we only specified six features in our model formula, there are eight coefficients reported in addition to the intercept. This happened because the `lm()` function automatically applied a technique known as **dummy coding** to each of the factor type variables we included in the model.

Dummy coding allows a nominal feature to be treated as numeric by creating a binary variable for each category of the feature, which is set to `1` if the observation falls into that category or `0` otherwise. For instance, the `sex` variable has two categories, `male` and `female`. This will be split into two binary values, which R names `sexmale` and `sexfemale`. For observations where `sex = male`, then `sexmale = 1` and `sexfemale = 0`; if `sex = female`, then `sexmale = 0` and `sexfemale = 1`. The same coding applies to variables with three or more categories. The four-category feature `region` can be split into four variables: `regionnorthwest`, `regionsoutheast`, `regionsouthwest`, and `regionnortheast`.

When adding a dummy-coded variable to a regression model, one category is always left out to serve as the reference category. The estimates are then interpreted relative to the reference. In our model, R automatically held out the `sexfemale`, `smokerno`, and `regionnortheast` variables, making female non-smokers in the northeast region the reference group. Thus, males have $131.30 less medical costs each year relative to females and smokers cost an average of $23,848.50 more than non-smokers. Additionally, the coefficient for each of the other three regions in the model is negative, which implies that the northeast region tends to have the highest average medical expenses.

By default, R uses the first level of the `factor` variable as the reference. If you would prefer to use another level, the `relevel()` function can be used to specify the reference group manually. Use the `?relevel` command in R for more information.

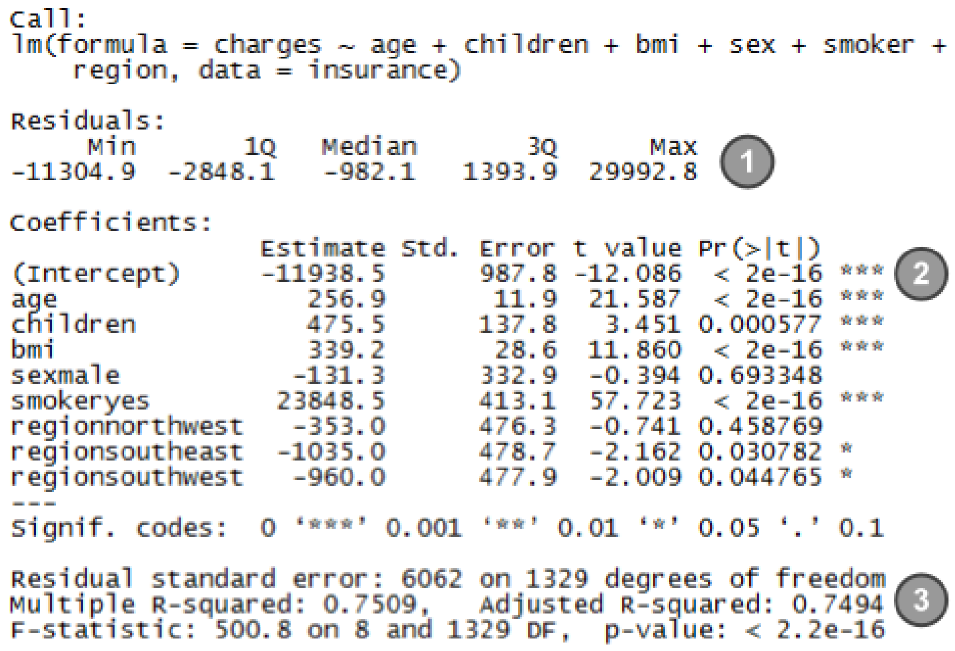
The results of the linear regression model make logical sense; old age, smoking, and obesity tend to be linked to additional health issues, while additional family member dependents may result in an increase in physician visits and preventive care such as vaccinations and yearly physical exams. However, we currently have no sense of how well the model is fitting the data. We'll answer this question in the next section.

Step 4 – evaluating model performance

The parameter estimates we obtained by typing `ins_model` tell us about how the independent variables are related to the dependent variable, but they tell us nothing about how well the model fits our data. To evaluate the model performance, we can use the `summary()` command on the stored model:

```
> summary(ins_model)
```

This produces the following output:



```
Call:
lm(formula = charges ~ age + children + bmi + sex + smoker +
    region, data = insurance)

Residuals:
     Min       1Q   Median       3Q      Max
-11304.9  -2848.1   -982.1   1393.9  29992.8

Coefficients:
                Estimate Std. Error t value Pr(>|t|)
(Intercept)     -11938.5      987.8 -12.086  < 2e-16 ***
age                256.9       11.9  21.587  < 2e-16 ***
children           475.5      137.8   3.451 0.000577 ***
bmi                339.2       28.6  11.860  < 2e-16 ***
sexmale           -131.3      332.9  -0.394 0.693348
smokeryes        23848.5      413.1  57.723  < 2e-16 ***
regionnorthwest   -353.0      476.3  -0.741 0.458769
regionsoutheast  -1035.0      478.7  -2.162 0.030782 *
regionsouthwest   -960.0      477.9  -2.009 0.044765 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1

Residual standard error: 6062 on 1329 degrees of freedom
Multiple R-squared: 0.7509,    Adjusted R-squared: 0.7494
F-statistic: 500.8 on 8 and 1329 DF,  p-value: < 2.2e-16
```

The `summary()` output may seem confusing at first, but the basics are easy to pick up. As indicated by the numbered labels in the preceding output, the output provides three key ways to evaluate the performance (that is, fit) of our model:

1. The `Residuals` section provides summary statistics for the errors in our predictions, some of which are apparently quite substantial. Since a residual is equal to the true value minus the predicted value, the maximum error of `29992.8` suggests that the model under-predicted expenses by nearly $30,000 for at least one observation. On the other hand, 50 percent of errors fall within the 1Q and 3Q values (the first and third quartile), so the majority of predictions were between $2,850 over the true value and $1,400 under the true value.

2. The stars (for example, `***`) indicate the predictive power of each feature in the model. The significance level (as listed by the `Signif.` codes in the footer) provides a measure of how likely the true coefficient is zero given the value of the estimate. The presence of three stars indicates a significance level of `0`, which means that the feature is extremely unlikely to be unrelated to the dependent variable. A common practice is to use a significance level of 0.05 to denote a statistically significant variable. If the model had few features that were statistically significant, it may be cause for concern, since it would indicate that our features are not very predictive of the outcome. Here, our model has several significant variables, and they seem to be related to the outcome in logical ways.
3. The `Multiple R-squared` value (also called the coefficient of determination) provides a measure of how well our model as a whole explains the values of the dependent variable. It is similar to the correlation coefficient in that the closer the value is to 1.0, the better the model perfectly explains the data. Since the R-squared value is `0.7494`, we know that nearly 75 percent of the variation in the dependent variable is explained by our model. Because models with more features always explain more variation, the `Adjusted R-squared` value corrects R-squared by penalizing models with a large number of independent variables. It is useful for comparing the performance of models with different numbers of explanatory variables.

Given the preceding three performance indicators, our model is performing fairly well. It is not uncommon for regression models of real-world data to have fairly low R-squared values; a value of 0.75 is actually quite good. The size of some of the errors is a bit concerning, but not surprising given the nature of medical expense data. However, as shown in the next section, we may be able to improve the model's performance by specifying the model in a slightly different way.

Step 5 – improving model performance

As mentioned previously, a key difference between regression modeling and other machine learning approaches is that regression typically leaves feature selection and model specification to the user. Consequently, if we have subject matter knowledge about how a feature is related to the outcome, we can use this information to inform the model specification and potentially improve the model's performance.

Model specification – adding non-linear relationships

In linear regression, the relationship between an independent variable and the dependent variable is assumed to be linear, yet this may not necessarily be true. For example, the effect of age on medical expenditures may not be constant throughout all age values; the treatment may become disproportionately expensive for the oldest populations.

If you recall, a typical regression equation follows a form similar to this:

$$y = \alpha + \beta_1 x$$

To account for a non-linear relationship, we can add a higher order term to the regression model, treating the model as a polynomial. In effect, we will be modeling a relationship like this:

$$y = \alpha + \beta_1 x + \beta_2 x^2$$

The difference between these two models is that a separate beta will be estimated, which is intended to capture the effect of the *x*-squared term. This allows the impact of age to be measured as a function of age squared.

To add the non-linear age to the model, we simply need to create a new variable:

```
> insurance$age2 <- insurance$age^2
```

Then, when we produce our improved model, we'll add both `age` and `age2` to the `lm()` formula, for example, `charges ~ age + age2`.

Transformation – converting a numeric variable to a binary indicator

Suppose we have a hunch that the effect of a feature is not cumulative, but rather it has an effect only once a specific threshold has been reached. For instance, BMI may have zero impact on medical expenditures for individuals in the normal weight range, but it may be strongly related to higher costs for the obese (that is, BMI of 30 or above).

We can model this relationship by creating a binary indicator variable that is 1 if the BMI is at least 30 and 0 otherwise. The estimated beta for this binary feature would then indicate the average net impact on medical expenses for individuals with BMI of 30 or above, relative to those with BMI less than 30.

To create the feature, we can use the `ifelse()` function, which for each element in a vector tests a specified condition and returns a value depending on whether the condition is true or false. For BMI greater than or equal to 30, we will return `1`, otherwise `0`:

```
> insurance$bmi30 <- ifelse(insurance$bmi >= 30, 1, 0)
```

We can then include the `bmi30` variable in our improved model, either replacing the original `bmi` variable or in addition, depending on whether or not we think the effect of obesity occurs in addition to a separate BMI effect. Without good reason to do otherwise, we'll include both in our final model.

[If you have trouble deciding whether or not to include a variable, a common practice is to include it and examine the significance level. Then, if the variable is not statistically significant, you have evidence to support excluding it in the future.]

Model specification – adding interaction effects

So far, we have only considered each feature's individual contribution to the outcome. What if certain features have a combined impact on the dependent variable? For instance, smoking and obesity may have harmful effects separately, but it is reasonable to assume that their combined effect may be worse than the sum of each one alone.

When two features have a combined effect, this is known as an **interaction**. If we suspect that two variables interact, we can test this hypothesis by adding their interaction to the model. Interaction effects can be specified using the R formula syntax. To interact the obesity indicator (`bmi30`) with the smoking indicator (`smoker`), we would write a formula in the form `charges ~ bmi30*smoker`

The `*` operator is shorthand that instructs R to model `charges ~ bmi30 + smokeryes + bmi30:smokeryes`

The `:` (colon) operator in the expanded form indicates that `bmi30:smokeryes` is the interaction between the two variables. Note that the expanded form automatically also included the `bmi30` and `smoker` variables as well as the interaction.

[Interactions should never be included in a model without also adding each of the interacting variables. If you always create interactions using the `*` operator, this will not be a problem since R will add the required components for you automatically.]

Putting it all together – an improved regression model

Based on a bit of subject matter knowledge of how medical costs may be related to patient characteristics, we developed what we think is a more accurately-specified regression formula. To summarize the improvements, we:

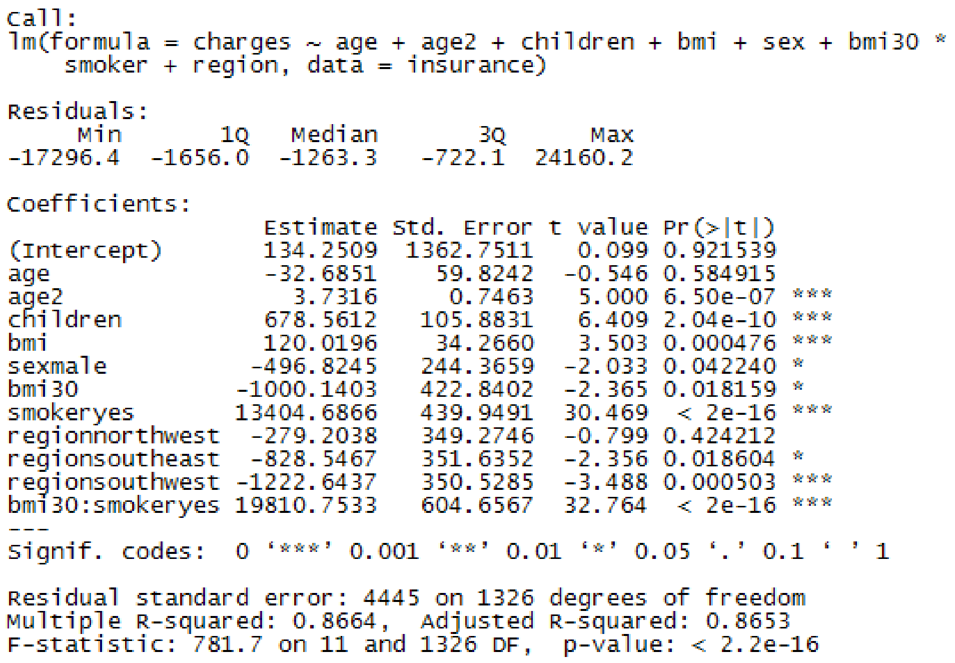
- Added a non-linear term for age
- Created an indicator for obesity
- Specified an interaction between obesity and smoking

We'll train the model using the `lm()` function as before, but this time we'll add the newly constructed variables and the interaction term:

```
> ins_model2 <- lm(charges ~ age + age2 + children + bmi + sex +
                     bmi30*smoker + region, data = insurance)
```

Next, we summarize the results:

```
> summary(ins_model2)
```



```
Call:
lm(formula = charges ~ age + age2 + children + bmi + sex + bmi30 *
    smoker + region, data = insurance)

Residuals:
     Min       1Q   Median       3Q      Max
-17296.4  -1656.0  -1263.3   -722.1  24160.2

Coefficients:
                  Estimate Std. Error t value Pr(>|t|)
(Intercept)       134.2509  1362.7511   0.099 0.921539
age               -32.6851    59.8242  -0.546 0.584915
age2                3.7316     0.7463   5.000 6.50e-07 ***
children          678.5612   105.8831   6.409 2.04e-10 ***
bmi               120.0196    34.2660   3.503 0.000476 ***
sexmale          -496.8245   244.3659  -2.033 0.042240 *
bmi30           -1000.1403   422.8402  -2.365 0.018159 *
smokeryes       13404.6866   439.9491  30.469  < 2e-16 ***
regionnorthwest  -279.2038   349.2746  -0.799 0.424212
regionsoutheast  -828.5467   351.6352  -2.356 0.018604 *
regionsouthwest -1222.6437   350.5285  -3.488 0.000503 ***
bmi30:smokeryes 19810.7533   604.6567  32.764  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 4445 on 1326 degrees of freedom
Multiple R-squared: 0.8664,  Adjusted R-squared: 0.8653
F-statistic: 781.7 on 11 and 1326 DF,  p-value: < 2.2e-16
```

The model fit statistics help to determine whether our changes improved the performance of the regression model. Relative to our first model, the R-squared value has improved from 0.75 to about 0.87. Our model is now explaining 87 percent of the variation in medical treatment costs. Additionally, our theories about the model's functional form seem to be validated. The higher-order `age2` term is statistically significant, as is the obesity indicator, `bmi30`. The interaction between obesity and smoking suggests a massive effect; in addition to the increased costs of over $13,404 for smoking alone, obese smokers spend another $19,810 per year. This may suggest that smoking exacerbates diseases associated with obesity.

[redacted]