Chapter 1

More general regression methods

The regression methods covered thus far only apply to situations in which we want a linear model for the relationship between two numerical variables. While this is an extremely useful capability, there are many important applications in the real world that do not conform to these limitations. A quick look at the examples in the earlier chapters makes this clear: The box-office revenue from a movie depends not only on its production budget, but also on various other factors such as its cast, genre, run time, advertising budget, and more. Airline fares depend not only on the distance traveled, but also on available inventory, advance purchase, travel season, advertising, and other variables.

It is clear that in many applications of practical interest, we want to model the effect of several variables acting in tandem, upon a response variable. In addition, we often find that one or more of these variables is categorical, instead of numerical. This suggests there are at least two directions in which we would like to generalize regression methods:

1. Model the relationship between a response variable and several predictor variables.

2. Accommodate categorical predictor variables.

We will consider categorical response variables later, since that is an important special topic in its own right.

This chapter introduces a number of key topics related to these generalizations. We begin by discussing the technical aspects of extending regression to multiple predictor variables, including software implementation in R. This is followed by examining the key theoretical assumptions and conditions that underlie a valid model. [**complete this paragraph after writing the bulk of this chapter**]

1.1 Multilinear regression

Linear regression with several predictor variables is known as multilinear regression (MLR). As with simple linear regression, we emphasize two major tasks in the modeling process

a. Construct/compute a suitable MLR model.

b. Assess the quality and validity of the model.

We will begin by focusing on how to setup MLR models, and compute the needed parameters. The task of assessing the models' quality and validity will be addressed in detail later.

Conceptually, the generalization from simple linear to multilinear is quite straightforward. To elucidate the parallel nature of the concepts, let us consider an example analogous to one we saw earlier.

Example:

A very small dataset consisting of only 4 observations is shown in the table. It contains 2 predictor variables (x, y) and 1 response variable (z). The goal is to find the plane of best fit using the least squares method – i.e., by minimizing the sum of the square of the errors.

x_i	y_i	z_i
0	0	12
1	0	20
0	1	42
3	3	30

Solution:
Here are the key steps we will follow

a. Assume the plane has the equation: $\hat{z} = b + mx + ny$, where b, m, n are to be determined

b. For each observation i , find the residual: $e_i = z_i - \hat{z}_i$

c. Compute the least squares cost function: $f = \sum(e_i)^2$

d. Minimize f: Set $\frac{\partial f}{\partial b} = \frac{\partial f}{\partial m} = \frac{\partial f}{\partial n} = 0$, and solve for b, m, n.

Computations based on the above steps:

a. Model: $\hat{z} = b + mx + ny$

b. Residuals: $e_1 = (12-b),\ e_2 = (20-b-m),\ e_3 = (42-b-n),\ e_4 = (30-b-3m-3n)$

c. Cost function: $f = (12-b)^2 + (20-b-m)^2 + (42-b-n)^2 + (30-b-3m-3n)^2$

d. Minimize:
$\frac{\partial f}{\partial b} = -2(12-b) - 2(20-b-m) - 2(42-b-n) - 2(30-b-3m-3n) = 0 \quad \Rightarrow$
$104 - 4b - 4m - 4n = 0$
$\frac{\partial f}{\partial m} = -2(20-b-m) - 6(30-b-3m-3n) = 0 \quad \Rightarrow 110 - 4b - 10m - 9n = 0$
$\frac{\partial f}{\partial n} = -2(42-b-n) - 6(30-b-3m-3n) = 0 \quad \Rightarrow 132 - 4b - 9m - 10n = 0$
Solve for b, m, n and get: $n = 138/11, m = -104/11, b = 252/11$

Answer: The plane of best fit is $\hat{z} = \frac{252}{11} - \frac{104}{11}x + \frac{138}{11}y$

In principle, the strategy used in this example can be extended to any (finite) number of predictor variables. In practice, MLR computations are almost always carried out using software or technology tools. However, before we do that, let us formulate the problem in clear mathematical terms.

Assume we have a dataset containing k predictor variables, $x_1, x_2, \ldots, x_k$, and a response variable, y. To begin with, we assume all variables are numerical. The form of the MLR model is taken to be

$$\hat{y} = b_0 + b_1 x_1 + b_2 x_2 + \ldots + b_k x_k, \qquad k \in \mathbb{N} \tag{1.1}$$

where $\hat{y}$ is the predicted response, and $b_0, b_1, \ldots, b_k$ are the unknown regression coefficients that we seek to determine. This is done using a least squares formulation that minimizes the sum of the square of the residuals. It is possible to write closed-form expressions for computing the regression coefficients in terms of the numerical values of the variables in our dataset. But the resulting computations are extremely cumbersome, and it is common practice to simply rely on technology to obtain the coefficient values. For the interested reader, we provide a summary of the usual matrix-based formalism for computing the regression coefficients in the boxed supplement below.

How to compute the MLR coefficients (optional reading!)

Assume our dataset contains n records (or observations), with k predictor variables and a response variable. The MLR model in equation (1.1) can be re-written as

$$y = b_0 + b_1 x_1 + b_2 x_2 + \ldots + b_k x_k + e$$

where y is the actual/true response and e is the residual. For convenience, we introduce the following matrix-vector notation to denote the variables in our dataset:

- Response variable $= \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$

- Matrix $\mathbf{X} = [\mathbf{1}, \mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_k]$, where each entry is an $n \times 1$ column vector. The first entry is an $n \times 1$ vector of 1's. The remaining entries are the predictor variables, each of which is an $n \times 1$ vector. Thus,

  $$\mathbf{x}_1 = \begin{bmatrix} x_{11} \\ x_{12} \\ \vdots \\ x_{1n} \end{bmatrix}, \quad \mathbf{x}_2 = \begin{bmatrix} x_{21} \\ x_{22} \\ \vdots \\ x_{2n} \end{bmatrix}, \quad \cdots \quad \mathbf{x}_k = \begin{bmatrix} x_{k1} \\ x_{k2} \\ \vdots \\ x_{kn} \end{bmatrix}$$

  The dimensions of $\mathbf{X}$ are $n \times (k+1)$, and its first row is: $[1, x_{11}, x_{21}, \ldots, x_{k1}]$

- Let $\mathbf{b} = [b_0, b_1, b_2, \ldots, b_k]^T$ denote the $k+1$ vector of regression coefficients.

In this notation the MLR model, when applied to the given dataset, looks like

$$\mathbf{y} = \mathbf{X}\mathbf{b} + \mathbf{e} \qquad (\text{OR } \ \hat{\mathbf{y}} = \mathbf{X}\mathbf{b})$$

where $\mathbf{e}$ is the $n \times 1$ vector of residuals.

The least squares cost function (sum of the square of the residuals) is

$$f(\mathbf{b}) = \mathbf{e}^T\mathbf{e} = (\mathbf{y} - \mathbf{X}\mathbf{b})^T(\mathbf{y} - \mathbf{X}\mathbf{b})$$

To minimize f, we set $\nabla f = 2[\mathbf{X}^T\mathbf{X}\ \mathbf{b} - \mathbf{X}^T\mathbf{y}] = \mathbf{0}$ and solve for $\mathbf{b}$. The result is

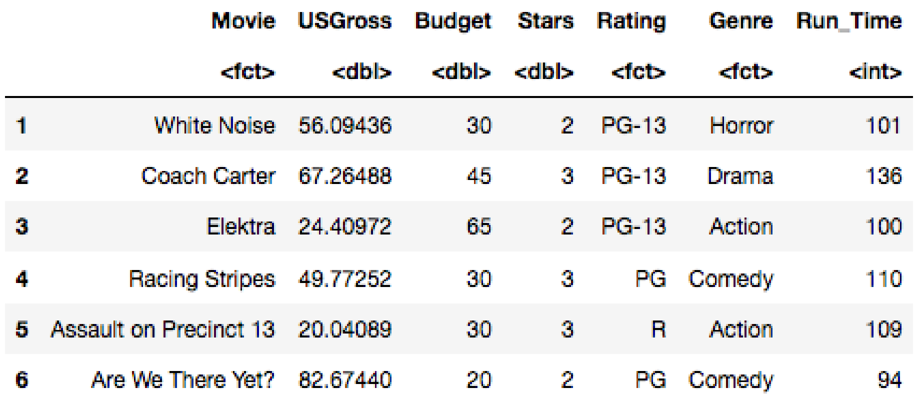
$$\mathbf{b} = (\mathbf{X}^T\mathbf{X})^{-1}\ \mathbf{X}^T\mathbf{y}$$

The optimal regression coefficients can be computed using this final result.

An important question that arises, especially when dealing with large datasets such as those seen in many data science and data analytics applications, is: How do we decide what variables to use as predictors? Should we use all the available variables? Or, is there some optimal subset that would yield the best model? We address questions such as these later, as part of our discussion on model optimization. Note that we will generally assume the choice of response variable is fixed.

The general form of the model given in equation (1.1) holds for any choice of predictor variables. In the R software framework, MLR is implemented using the same function that performs simple linear regression. To illustrate its usage, let us revisit the movies dataset that we considered in an earlier chapter.

Example:
The file `movies.csv` contains data on a sample of 120 movies produced in the United States, together with information on certain variables associated with each movie. The first few lines of the datafile are shown below



	Movie	USGross	Budget	Stars	Rating	Genre	Run_Time
	<fct>	<dbl>	<dbl>	<dbl>	<fct>	<fct>	<int>
1	White Noise	56.09436	30	2	PG-13	Horror	101
2	Coach Carter	67.26488	45	3	PG-13	Drama	136
3	Elektra	24.40972	65	2	PG-13	Action	100
4	Racing Stripes	49.77252	30	3	PG	Comedy	110
5	Assault on Precinct 13	20.04089	30	3	R	Action	109
6	Are We There Yet?	82.67440	20	2	PG	Comedy	94

In this example we will construct a multilinear regression model to predict a movie's `USGross` based on its `Budget`, `Stars` and `Run_Time`. We note that the variables `USGross` and `Budget`

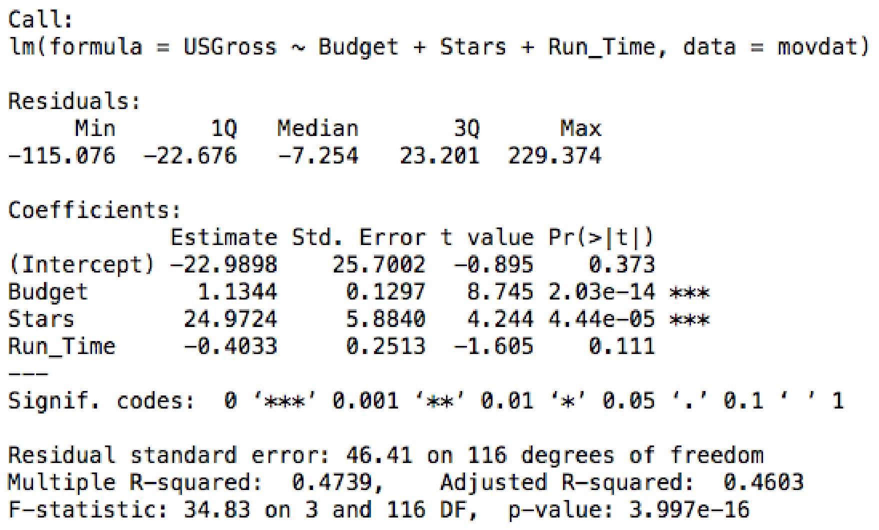
are in millions of dollars, `Run_Time` is in minutes, and `Stars` has no units.
Solution:
Assume a model of the form

$$\widehat{\texttt{USGross}} = b_0 + b_1(\texttt{Budget}) + b_2(\texttt{Stars}) + b_3(\texttt{Run\_Time}) \tag{1.2}$$

where $\widehat{\texttt{USGross}}$ is the predicted `USGross`, and $b_0, \ldots, b_3$ are constant regression coefficients. The following R code reads the data file, performs MLR, and prints out the summary results.

```r
# Read datafile and store data in a dataframe:
movdat = read.csv(file="https://cs.earlham.edu/~pardhan/sage_
   and_r/movies.csv", header=TRUE, sep=",")

# Construct MLR model using the lm() function:
lmresults = lm(USGross ~ Budget+Stars+Run_Time, data=movdat)

# summarize results:
summary(lmresults)
```

Here is the corresponding output:



```
Call:
lm(formula = USGross ~ Budget + Stars + Run_Time, data = movdat)

Residuals:
     Min       1Q   Median       3Q      Max 
-115.076  -22.676   -7.254   23.201  229.374 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) -22.9898    25.7002  -0.895    0.373    
Budget        1.1344     0.1297   8.745 2.03e-14 ***
Stars        24.9724     5.8840   4.244 4.44e-05 ***
Run_Time     -0.4033     0.2513  -1.605    0.111    
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 46.41 on 116 degrees of freedom
Multiple R-squared:  0.4739,	Adjusted R-squared:  0.4603 
F-statistic: 34.83 on 3 and 116 DF,  p-value: 3.997e-16
```

Note that in practice it may be helpful for your code to include a line that also displays the header of the data file, as the names of the variables must exactly match those in the data file. The output summarizes several key results of interest when constructing an MLR model. Our present goal is to just find the regression coefficients, which are shown in the table named `Coefficients:` under the column named `Estimate`. The left-most column shows the name of the variable whose coefficient is in the `Estimate` column. Accordingly, the MLR model is

$$\widehat{\texttt{USGross}} = -22.9898 + 1.1344(\texttt{Budget}) + 24.9724(\texttt{Stars}) - 0.4033(\texttt{Run\_Time}) \tag{1.3}$$

A common practice after constructing an MLR model is to provide an interpretation of key slope coefficients. In this example, we could say:

- For each 1 million dollar increase in a movie's `Budget`, the model predicts an average increase of 1.1344 million dollars in its `USGross`, when all other variables are held constant.

- For each 1 unit increase in a movie's `Stars` rating, the model predicts an average increase of 24.9724 million dollars in its `USGross`, when all other variables are held constant.

- For each 1 minute increase in a movie's `Run_Time`, the model predicts an average decrease of 0.4033 million dollars in its `USGross`, when all other variables are held constant.

We emphasize, the focus of this example has simply been on illustrating how to use R to compute an MLR model. The example has intentionally ignored the important question of whether the resulting model is valid or reliable for any practical purpose. In order to address those questions, we need to discuss some additional theoretical background on the assumptions and conditions that underlie MLR models.

Assumptions and conditions

Like simple linear regression, the necessary conditions for a valid MLR model broadly consist of four components:

a. Linearity: Each predictor variable must be approximately linearly related to the response variable, and have no significant outliers.

b. Normal residuals: The residuals resulting from the model must be approximately normally distributed, with mean=0.

c. Constant variance: The residuals must exhibit approximately constant variance as a function of predicted values.

d. Independent observations: The data in the sample must consist of independent observations.

It is common practice to use graphical checks for most of these conditions, although this still leaves open some questions about exactly which graph to use for which condition. For example, to check linearity the recommended method is to graph the residuals versus each predictor variable (e vs x), instead of the response versus each predictor (y vs x). The rationale is that the residuals include the effect of all the variables in our model, and not just the bivariate relationship between y and x. Predictors that satisfy the linearity condition will show a random scatter of points above and below 0 in the e vs x plot, with no discernable curve or pattern.

The normal residuals condition is typically checked by plotting a histogram of the residuals, and/or with a normal probability plot. If the histogram looks nearly symmetric, unimodal and centered around 0, we assume the condition is met. In a normal probability plot, we want the residuals to approximately follow the diagonal reference line that indicates a perfect normal distribution.

The constant variance condition is checked using a scatter plot of the residuals versus predicted values. Again, the preference here is to plot against predicted values because they include the effect of the entire model. If the condition is satisfied, the residuals should exhibit a scatter of points above and below 0 in a roughly constant band throughout the plot.

Lastly, for the independent observations condition, it is usually necessary to obtain information about how the data were sampled. Ideally, we want a random sample that is representative of the underlying population of interest. A graphical check that is also often recommended is to plot the residuals versus the order in which the data are recorded. This graph will show any time-dependent pattern, if it is present in the data.

As an example, let us check whether the movies data set we saw earlier satisfies each of these conditions. Figure 1.1 shows a plot of the residuals versus each of the predictor variables.

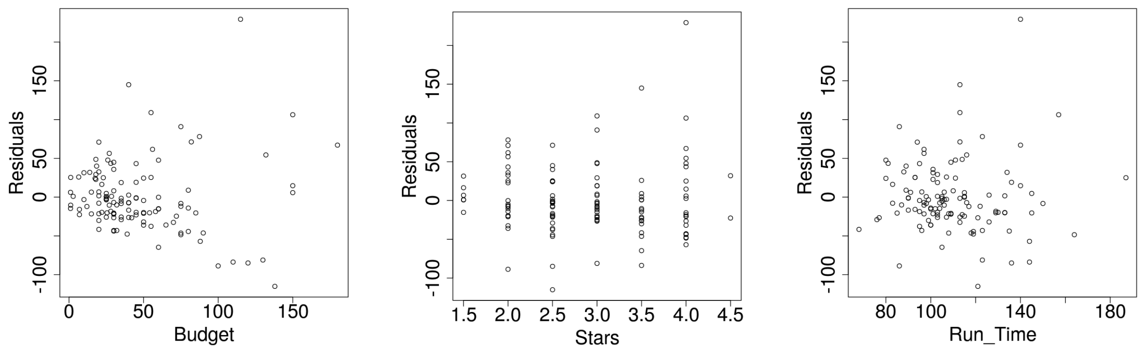
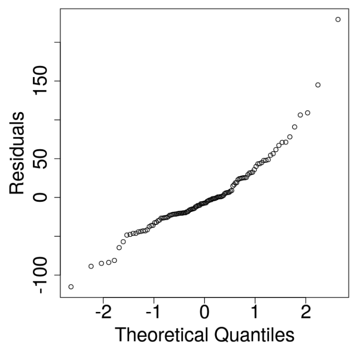


Figure 1.1: Scatterplots of residuals vs predictors for the movies example.

The `Budget` variable certainly seems to violate linearity, as there appears to be a downward sloping trend in the lower half of the domain. More importantly, all 3 graphs exhibit the presence of at least one significant outlier, with a residual value close to 200. Thus, it seems reasonable to say this dataset fails to satisfy the first condition. It is possible the data may satisfy the linearity condition if we remove the outlier and repeat the analysis.

In Figure 1.2 we see a normal probability plot and histogram of the residuals. Although the trend is not perfectly normal, it is within range of acceptable. This is particularly true because the sample size is 120, which is not small. The normal residuals condition is not as critical for larger sample sizes. Hence, it is reasonable to conclude the normal residuals condition is satisfied.

To check the constant variance condition, Figure 1.3 shows a plot of the residuals versus predicted values. As seen in the graph, the variance increases significantly when we move from lower to higher predicted values. Thus, these data clearly violate the constant variance condition.



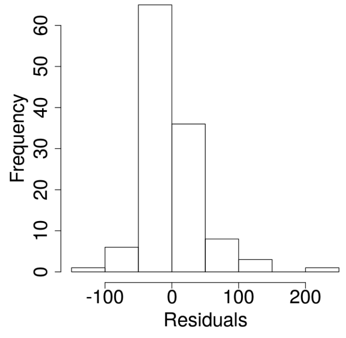


Figure 1.2: Residuals normal probability plot and histogram for the movies example.

Finally, to check the independent observations condition, we require more information about how the data were compiled – e.g., whether the sample is random, or at least representative of movies produced in the United States. Since we do not have access to this information, we cannot assume independence.

To summarize, the movies dataset fails to satisfy the linearity, the constant variance, and the independent observations conditions. Thus, it would not be appropriate to use multilinear regression in this situation.

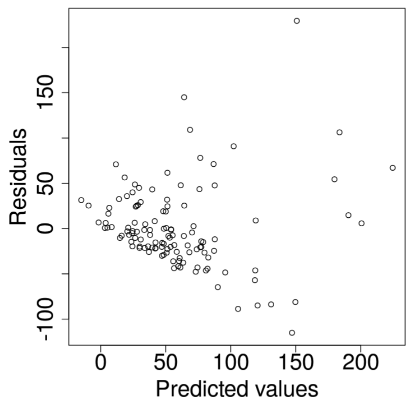


Figure 1.3: Residuals vs predicted for the movies example.

We will further demonstrate the use of all these strategies for checking the conditions in a variety of examples later.

The adjusted R^2 indicator

Adjusted R^2 is a measure of how closely the MLR model fits the sampled data. It estimates what proportion of the variance in the response (i.e., observed y values) is explained by the variance in the predictor variables. Recall, for a single predictor we had defined

$$R^2 = 1 - \frac{\text{variance in residuals}}{\text{variance in observed } y} \tag{1.4}$$

When we include more predictors, the aspiration is that each predictor will help make the model better than it is without that predictor. Since the R^2 value in (1.4) may improve slightly even with predictors that are only weakly correlated with the response, we would like to define a new indicator that only improves when the benefit from the predictor is sufficiently strong. This is the idea behind adjusted R^2, which is defined as

$$R^2_{adj} = 1 \; - \; \left(\frac{\text{variance in residuals}}{\text{variance in observed } y}\right) \times \left(\frac{n-1}{n-k-1}\right) \tag{1.5}$$

where n is the number of observations or records in the dataset, and k is the number of predictor variables. Notice that the multiplication factor $\left(\frac{n-1}{n-k-1}\right)$ is always greater than 1, which means R^2_{adj} is always smaller than R^2. Furthermore, as the number of predictors increase, R^2_{adj} will tend to decrease, unless the new predictors help to cut down the residuals significantly. This is exactly the behavior we want, since it is not biased in favor of increasing the number of predictors, as equation (1.4) is.

As an illustration, in the movies example seen earlier, the summary output from R shows that $R^2 = 0.4739$ and $R^2_{adj} = 0.4603$. To assess the quality of fit of the associated MLR model, we could say that about 46% of the variance in a movie's `USGross` is explained by this model. Later we will look at a strategy for selecting an optimal set of predictors in MLR models in such a way that the R^2_{adj} is maximized.

1.2 Categorical predictor variables

In many real world applications it is common to want an MLR model to include one or more categorical predictors. To illustrate with a familiar example, the movies dataset seen earlier contains a `Genre` variable that indicates the type of movie (comedy, horror, action, etc.). It is possible that `Genre` is a significant predictor of `USGross`, in conjunction with some of the other predictors. Similarly, an MLR model to predict airline fares might want to include variables such as season (summer, winter, etc.), and route competition (high, low), among others.

The simplest categorical predictors are binary variables, those with only two complementary opposite categories. It turns out that once we learn how to include binary variables in our model, we can reduce all categorical predictors to a collection of binary variables. Let us consider an example that shows how to include binary variables in linear regression models.

Example:
A dataset contains nutrition information on food items served at popular fast food restaurants. To illustrate how to work with categorical predictors, we consider a subset of these data, as shown in the table. For each food item, its type and total fat content (in grams) are listed. Our goal is to develop a linear regression model to predict total fat

Item	Name	Type	Fat (g)
1	Big Mac	Burger	27
2	French fries	FF	14
3	Chocolate shake	MS	9
4	Quarter pounder	Burger	26
5	Whopper	Burger	38
6	Grilled chikken SW	CS	16

from the type of the food item. For the purpose of this example we will reduce "Type" to a binary categorical variable by taking its value to be either "Burger" or "not Burger." However, this is not a restriction inherent to the method. In later examples we will show how to accommodate multiple categories without reducing or grouping any of them.

Solution
Here is our strategy:

i. Since Type is binary, we turn it into a numerical variable using 0's and 1's.

ii. We perform standard linear regression with the numerical variables Type and Fat.

Continuing on to the details, since we want to predict Fat from Type:
Predictor or x-variable = Type (no units)
Response or y-variable = Fat (grams)

We convert Type to numeric by defining: Burger=1, and not Burger=0.

Next, we compute the correlation and summary statistics using software and obtain:
$\bar{x} = 0.5,\ s_x = 0.5477,\ \bar{y} = 21.6667,\ s_y = 10.6333,\ r = 0.893$

The slope of the regression line is: $b_1 = r\frac{s_y}{s_x} = 0.893\left(\frac{10.6333}{0.5477}\right) = 17.337$ grams

The intercept is: $b_0 = \bar{y} - b_1\bar{x} = 21.6667 - (17.337)(0.5) = 12.998$ grams

Therefore, the line of best fit is: $\hat{y} = 12.998 + 17.337x$

The corresponding linear regression model is usually written as

$$\widehat{\text{Fat (g)}} = 12.998 + 17.337\ (\text{Type:Burger}) \tag{1.6}$$

The x-variable Type:Burger takes on the value 1 when the Type is Burger. Otherwise, its value is 0.

As always, we note the value of the slope, and provide a context-based interpretation:
When the Type of a fast food item is Burger, the model predicts that its Fat content, on average, will be 17.337 grams higher.

With a categorical predictor, the intercept of the model is also meaningful to interpret. In this example, we could say:
The model predicts that fast food items that are not Burgers will contain, on average, 12.998 grams of fat.

To summarize what we learned from this example

- Assign a value of 0 or 1 to the levels of a binary categorical predictor. This will convert it into a numerical variable. The category to which 0 is assigned is called the reference level.

- Proceed as usual to construct a linear or multilinear regression model including the categorical predictor in its converted numerical form.

- Write the final model using variable names that make your choice of reference level clear and explicit. For example, the variable name "Type:Burger" makes it explicit that the reference level is "not Burger" because it corresponds to Type:Burger=0.

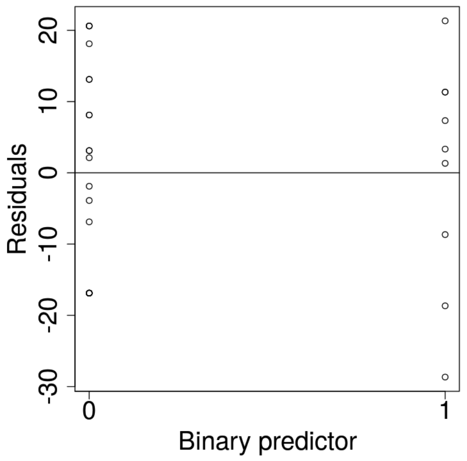
- Interpret the intercept of the model as its average prediction for the reference level, and the slope as the average change predicted with a change to the non-reference level.

A critically important question overlooked in this example is: What about the assumptions and conditions? The strategy used in the example is, essentially, a form of simple linear regression between two numerical variables. Thus, the conditions that must be satisfied are similar, and can be summarized as:

1. The residuals at both levels of the binary variable must be approximately normally distributed, with mean=0 (see graph for reference).

2. The residuals at both levels must have about equal variance.

3. the data must consist of independent observations.



The linearity condition is automatically satisfied for a binary predictor, since there are only two levels.

Generalization to non-binary predictors

When there are more than two categories in a predictor, as is true in many applications, the usual strategy is to split the predictor into multiple binary predictors. For instance, if a categorical predictor has k distinct levels, it can be split into $k-1$ binary predictors.

To illustrate, let us revisit the movies example we saw earlier. Our multilinear model for predicting `USGross` based on the predictors `Budget`, `Stars` and `Run_Time` had the form

$$\widehat{\texttt{USGross}} = b_0 + b_1(\texttt{Budget}) + b_2(\texttt{Stars}) + b_3(\texttt{Run\_Time})$$

Suppose we want to expand the model and also include the categorical predictor `Genre`, which consists of the 4 levels: `Action, Comedy, Drama, Horror`. We will replace `Genre` with the following 3 binary predictors: `Genre:Action, Genre:Comedy, Genre:Drama`. Each of these predictors has value 0 or 1, depending on the movie's `Genre`. For instance, if the movie's `Genre` is `Comedy`, then `Genre:Comedy`=1, and `Genre:Action` = `Genre:Drama` = 0. On the other hand, if the movie's `Genre` is `Horror`, then `Genre:Action` = `Genre:Comedy` = `Genre:Drama` = 0. This makes `Horror` the reference level for `Genre`, because it is the default category when a movie is not of any of the named binary types.

The final MLR model would have the form

$$\begin{aligned}\widehat{\texttt{USGross}} &= b_0 + b_1(\texttt{Budget}) + b_2(\texttt{Stars}) + b_3(\texttt{Run\_Time}) \\ &+ b_4(\texttt{Genre:Action}) + b_5(\texttt{Genre:Comedy}) + b_6(\texttt{Genre:Drama})\end{aligned}$$

Once the model is formulated, the process for finding the slope coefficients $b_0, \ldots, b_6$ remains the same – it is based on finding the best fit to the given dataset, by minimizing the sum of the square of the residuals. This is generally done using technology/software.

The overall process illustrated in this example readily extends to any number of categorical predictors, with any number of levels. However, it is important to note that increasing the number of predictors in an MLR model is neither desirable nor does it necessarily improve the model's reliability. In fact, for large-data applications that contain many variables, the question of how many predictors to use, and how to select the optimal ones, is so important that we will shortly devote an entire section to this topic.

1.3 Inference strategies for MLR

As with simple linear regression, the larger goal in MLR is to model the characteristics of some underlying population from which we draw a representative sample. Although our primary interest is in the population, the MLR model is developed entirely based on a single sample. Thus, a key task that remains is to develop rigorous and reliable strategies to extend sample-based understanding to population-based understanding. This is one of the key goals of inference methods.

In many modern MLR applications, the number of variables available to choose as predictors is large. In such situations we strive to select a smaller subset of the available variables, since it is generally not a good idea to use all of them as predictors. Inference methods also play an important role in this process of model optimization, by offering quantitative yardsticks based on which we can select an optimal set of predictors.

Before getting into details, it is helpful to clearly summarize the context and notation. As in the previous sections, we assume our MLR model contains k predictors, and has the form

$$\hat{y} = b_0 + b_1x_1 + b_2x_2 + \ldots + b_kx_k \tag{1.7}$$

The slope coefficients $b_0, \ldots, b_k$ are estimated by fitting the model to some sample dataset using least squares. This model approximates the true multilinear model for the underlying population, which is assumed to have the form

$$\hat{y} = \beta_0 + \beta_1x_1 + \beta_2x_2 + \ldots + \beta_kx_k \tag{1.8}$$

Each β_i in (1.8) is the true slope coefficient, for which the corresponding b_i in (1.7) is an approximation. We will introduce inference strategies to address the following questions:

1. Does the model in (1.7), taken as a whole, provide statistically significant predictions of the response? Technically, this is equivalent to asking whether at least one of the β_i in (1.8) is non-zero.

2. For each specific i, is the estimated b_i statistically significant? This is equivalent to asking whether the i^{th} predictor has a significant relationship to the response, given all the other predictors in the model.

3. Given a specific set of values for the predictors, say $\{x_1^c, x_2^c, \ldots, x_k^c\}$, what is the margin of error in the predicted response? This is analogous to computing confidence intervals and prediction intervals for $\hat{y}$, as we did earlier for a single predictor.

In the rest of this discussion, we will assume the reader is familiar with basic inference techniques for the single predictor case.

Significance of the model as a whole

This is essentially asking whether equation (1.7), taken together with the estimated values of $b_0, \ldots, b_k$, provides statistically significant predictions of the response. A hypothesis test based on the ANOVA framework is typically used to answer this question. We do not assume the reader has any background in ANOVA, nor is that necessary for a basic understanding of the inference processes we describe here. Our emphasis is primarily on the conceptual framework, logic, and interpretation of the significance test. Almost all the needed computations will be done using software.

The hypothesis test follows the usual sequence of steps. Before carrying out the mechanics, it is important to verify that the data satisfy the required conditions for MLR: Linear relationship between response and each predictor, normal residuals, constant variance, and independent observations. To help us understand the hypotheses, consider a relationship of the form (1.8) such that none of the predictors has any effect on the response. Then, clearly $\beta_1 = \beta_2 = \ldots = \beta_k = 0$. The logical complement of this statement is: $\beta_i \neq 0$ for at least one $i \in \{1, 2, \ldots, k\}$. This understanding will help us formulate the hypotheses. The steps for carrying out the test are summarized below.

Hypothesis test for significance of MLR model as a whole

This test is performed after computing the slope estimates b_i. A suitable significance level, say α, is chosen in advance. This is the threshold probability below which a P-value will be considered significant.

1. Verify that the data satisfy the required conditions for MLR.

2. Hypotheses
   H_0 (Null hypothesis): $\beta_i = 0$ for all $i \in \{1, 2, \ldots, k\}$
   H_A (Alternate hypothesis): $\beta_i \neq 0$ for at least one $i \in \{1, 2, \ldots, k\}$.

3. The test statistic is computed using software, based on ANOVA theory. It is known as the F-statistic, and it comes with an associated *df* (degrees of freedom) value. It basically measures how much the residuals from our sample-based model differ from those we would get if the null hypothesis were true. Let $\hat{F}$ denote the value of the F-statistic for our model.

4. Compute the P-value, which is the probability that $F > \hat{F}$ on the appropriate probability distribution curve. Typically, software output includes the P-value.

5. If P-value $< \alpha$, reject the null hypothesis and conclude the sample provides strong evidence that at least one $\beta_i \neq 0$. Thus, at least one predictor does have an effect on the response.
   If P-value $> \alpha$, retain the null hypothesis. The sample does not provide evidence that the model as a whole is a significant predictor.

We emphasize that the most important aspect of this test is a nuanced understanding of its logic, and how to interpret its results. Its computational aspects are far less important, and they are handled by software, in any case. Accordingly, here are some key observations

- If the test results indicate that the model as a whole is a significant predictor, it does not mean that every predictor in that model is significant. It only means that at least one of the $\beta_i \neq 0$.

- Conversely, if the results indicate that the model is not a significant predictor, it is still possible that one or more of the individual variables is a significant predictor.

These conclusions follow from the fact that our hypothesis test strictly looks at the effect of all the predictors acting together. If we repeat the test with different subsets of the same predictors – or with a single predictor acting alone – the results will generally not be the same. This fact is important to keep in mind, as it plays a key role in model optimization strategies which we will discuss later.

For illustration, consider the MLR model we computed earlier for the movies dataset

$$\widehat{\texttt{USGross}} = -22.9898 + 1.1344(\texttt{Budget}) + 24.9724(\texttt{Stars}) - 0.4033(\texttt{Run\_Time})$$

The corresponding model for the underlying population would have the form

$$\widehat{\texttt{USGross}} = \beta_0 + \beta_1(\texttt{Budget}) + \beta_2(\texttt{Stars}) + \beta_3(\texttt{Run\_Time})$$

To test for the significance of the the model as a whole, the hypotheses would be
H_0: $\beta_1 = \beta_2 = \beta_3 = 0$
H_A: At least one of $\beta_1, \beta_2, \beta_3$ is not 0

Here is a copy of the R output we saw earlier

```
Call:
lm(formula = USGross ~ Budget + Stars + Run_Time, data = movdat)

Residuals:
     Min       1Q   Median       3Q      Max
-115.076  -22.676   -7.254   23.201  229.374

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -22.9898    25.7002  -0.895    0.373
Budget        1.1344     0.1297   8.745 2.03e-14 ***
Stars        24.9724     5.8840   4.244 4.44e-05 ***
Run_Time     -0.4033     0.2513  -1.605    0.111
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 46.41 on 116 degrees of freedom
Multiple R-squared:  0.4739,    Adjusted R-squared:  0.4603
F-statistic: 34.83 on 3 and 116 DF,  p-value: 3.997e-16
```

The last line says: “`F-statistic: 34.83 on 3 and 116 DF, p-value: 3.997e-16`” It is telling us the results of the significance test on the model as a whole. The F value is 34.83 with 3 predictors and 116 degrees of freedom. In MLR modeling, the degrees of freedom is given by: $df = n - k - 1$, where n is the sample size, and k is the number of predictors. The P-value shown here indicates that if the null hypothesis is true, then there is a 3.997×10^{-16} probability of a valid sample producing $F = 34.83$. Since this P-value is very small, had our movies dataset been a valid sample, we would reject H_0 and conclude that at least one of $\beta_1, \beta_2, \beta_3$ is not 0. However, we've seen previously that the movies dataset does not satisfy all the needed conditions, and so we cannot conclude anything meaningful here.

Significance of individual predictors

It is helpful to first clarify the question we are asking:

> Does the j^{th} predictor have a significant effect on the response, given all the other predictors in the model?

We will address this question using both hypothesis tests and confidence intervals. The hypothesis test is carried out using similar steps and logic, except we revert back to computing a t-statistic instead of an F-statistic. As usual, the test results are only valid when the assumptions and conditions are satisfied. The hypotheses to test whether the j^{th} predictor is significant are

H_0: $\beta_j = 0$, given all the other predictors in the model.
H_A: $\beta_j \neq 0$, given all the other predictors in the model.

Although the movies example does not satisfy the assumptions and conditions, we will continue to use it to illustrate the mechanics of our tests. Accordingly, suppose we wanted to test whether `Run_Time` is significant, given the other two predictors in our model, the hypotheses would be

H_0: $\beta_3 = 0$, given the other two predictors in the model.
H_A: $\beta_3 \neq 0$, given the other predictors in the model.

The test statistic, which is based on the t-distribution, is computed in almost the same way as we do for the single predictor case

$$ts = \frac{\text{estimated slope} - \text{hypothesized slope}}{\text{standard error of the estimate}} = \frac{b_3 - 0}{SE(b_3)}, \qquad (df = n - k - 1)$$

From the software output we get $b_3 = -0.4033$, and $SE(b_3) = 0.2513$. Therefore, $ts = \frac{-0.4033}{0.2513} = -1.6049$ on 116 df. The corresponding P-value, via R code, is

```
pt(-1.6049, df=116)*2   # we use *2 because the test is two-sided
> 0.111235
```

This represents about an 11.1% probability, and depending on the choice of α, the outcome could go either way. If we take $\alpha = 0.05$ (a widely used value in many applications), we would conclude that `Run_Time` is NOT a significant predictor, given the overall model.

Notice that the t-score and P-value for each predictor are also included in the regression output from R. However, we still recommend developing a clear understanding of the logic and computations, as this will make it possible to use the test in special situations beyond the default that is included in the regression output.

Confidence intervals are another common tool used for inferences on slope estimates of individual predictors. The logic and computations are nearly identical to those of any other standard confidence interval, which is summarized by the formula

$$\text{confidence interval} = \text{point estimate} \quad \pm \quad t^* \times (\text{standard error of the estimate})$$

The point estimate and standard error of the estimate are provided in the software output, and t^* is based on $n - k - 1$ degrees of freedom. For the `Run_Time` predictor in the movies example

point estimate $= b_3 = -0.4033$
$t^* = 1.98$ (for 95% confidence, with $df = 116$; R-code: `qt(0.975,df=116)`)
standard error of the estimate $= SE(b_3) = 0.2513$
$CI = -0.4033 \pm (1.98)(0.2513) = [-0.901, 0.094]$
Interpretation: When all other variables are held constant, for each additional minute of run time, the model predicts that on average the change in a movie's `USGross` will be between a decrease of 0.901 million dollars to an increase of 0.094 million dollars. Since the CI includes 0, we would infer that `Run_Time` is not a significant predictor of `USGross` when all the other variables are included in the model.

Confidence intervals for the predicted response

When we use an MLR model for forecasting the response to a given input predictor set, there are certain key sources of variability in the predictions. The first is due to sampling variability, or the fact that our model is based on a sample, which is necessarily different from the underlying population. Another source of variability is due to the fact that even a model based on the entire population only predicts the average response to a given input. Of course, this is also true in simple linear regression with only one predictor variable. And, just as we saw for the one predictor case, there are two types of confidence intervals for the predicted response

1. Prediction interval for the individual response to a specific input set.
2. Confidence interval for the mean response to a specific input set.

The variability in the individual response is always higher than the variability in the mean response. For this reason, a prediction interval is always wider than a confidence interval for any given input set. Both of these intervals are valid under the same assumptions and conditions as before, and follow the same type of computational format, with $n - k - 1$ degrees of freedom

$$\text{confidence interval} = \text{point estimate} \quad \pm \quad t^* \times (\text{standard error of the estimate})$$

The key difference between the two intervals is in how we compute the standard error of the estimate, which is based on the underlying probability distribution model for each type of interval. Following our usual practice, we will simply summarize the relevant formulas here, without getting into details of the theory.

Assume our MLR model has been fit to a data set consisting of n cases (or observations), with k predictor variables, whose values are denoted by the $n \times 1$ vectors $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_k$. For example, suppose the data consists of the following spreadsheet

Item #	Fat (g)	Calories	Protein (g)
1	27	530	24
2	14	280	3
3	9	430	8
4	26	520	30
5	38	630	26

where we want to predict Fat, using Calories and Protein as predictors. Then $n = 5$, $k = 2$, and

$$\mathbf{x}_1 = \begin{bmatrix} 530 \\ 280 \\ 430 \\ 520 \\ 630 \end{bmatrix}, \quad \mathbf{x}_2 = \begin{bmatrix} 24 \\ 3 \\ 8 \\ 30 \\ 26 \end{bmatrix}$$

Our goal is to find a confidence interval for the predicted response, given some specific set of input predictor values. Let us denote this input set of predictors by the $k \times 1$ vector $\boldsymbol{x}_0 = \{x_1^0, x_2^0, \ldots, x_k^0\}^T$. An example of this for the nutrition data shown above might be, $\boldsymbol{x}_0 = \{400, 15\}^T$.

Using this notation, the standard error of the estimate for a confidence interval is

$$SE_\mu = s_e \sqrt{\boldsymbol{x}_0^T \; (\mathbf{X}^T\mathbf{X})^{-1} \; \boldsymbol{x}_0} \tag{1.9}$$

and for a prediction interval it is

$$SE_p = s_e \sqrt{1 + \boldsymbol{x}_0^T \; (\mathbf{X}^T\mathbf{X})^{-1} \; \boldsymbol{x}_0} \tag{1.10}$$

Here s_e is the residual standard error, which is usually given in the regression summary output, and $\mathbf{X}$ is an $n \times (k+1)$ matrix consisting of the predictor variables in the data set

$$\mathbf{X} = [\mathbf{1}, \mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_k]$$

The first entry is an $n \times 1$ vector of 1's, and the remaining entries are $n \times 1$ vectors of the predictor variables, as defined earlier.

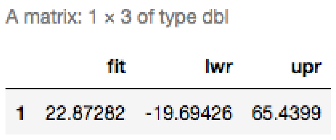
We note that due to the cumbersome nature of the computations involved in equations (1.9)-(1.10), we seldom compute these standard errors by hand. Nevertheless, the form of these equations does provide useful insights into the predicted response, and the sources of variability associated with it.

To illustrate how to use R for computing these intervals, consider the nutrition dataset above, where $n = 5$ and $k = 2$. The following R code inputs the data into a dataframe, performs MLR, and computes the predicted response for $\boldsymbol{x}_0 = \{400, 15\}^T$, together with 95% confidence and prediction intervals.

```r
# Example to illustrate how to compute confidence & prediction
# intervals in an MLR setting.  This example uses a very small
# dataset that we will input manually.

# Define the values of the variables in the dataset
Fat_g = c(27, 14, 9, 26, 38)
Calories = c(530, 280, 430, 520, 630)
Protein_g = c(24, 3, 8, 30, 26)

# Combine the variables into a dataframe that
# I will call "nutdata"
nutdata = data.frame(Fat_g, Calories, Protein_g)

# Construct MLR model to predict Fat, using Calories and
# Protein as predictors
nutout = lm(Fat_g ~ Calories+Protein_g, data=nutdata)
# Uncomment next line to see regression summary output
#summary(nutout)  # uncomment this line to see regression

# Define a dataframe with new input values of predictors
```

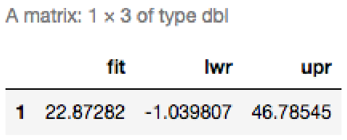
```r
newfooditem = data.frame(Calories=510, Protein_g=15)

# Compute 95% confidence and prediction intervals
predict(nutout, newdata=newfooditem, interval="prediction",
   level=0.95)
predict(nutout, newdata=newfooditem, interval="confidence",
   level=0.95)
```

The output, shown below, displays the predicted response, together with lower and upper bounds of the corresponding interval



A matrix: 1 × 3 of type dbl

	fit	lwr	upr
1	22.87282	-19.69426	65.4399



A matrix: 1 × 3 of type dbl

	fit	lwr	upr
1	22.87282	-1.039807	46.78545

In this example, the predicted fat content in a food item that contains 510 calories and 15 grams of protein is about 22.87 grams. The 95% prediction interval is $[-19.69426, 65.4399]$ grams, and the 95% confidence interval for the mean response is $[-1.039807, 46.78545]$ grams. Of course, a negative value for the amount of fat is not meaningful here, and we could just as well replace the lower bounds with 0. Note, also, that this dataset does not satisfy the necessary conditions for inference. Our primary intent here was to illustrate the use of R functions for computing confidence and prediction intervals in a simple setting.

1.4 Model optimization: How to select predictors

In many MLR applications, one of the most important questions that affects the quality of the model is how to choose the best predictor variables. This is especially true when datasets are large, and there are several variables available to choose from. The topic of model optimization strives to address this question.

A widely accepted guideline in model optimization is the principle of parsimony: The best model is the one that uses the smallest number of predictors needed to accomplish the modeling goals. For example, suppose we want a model that provides the closest fit to our sampled data. Then it would be reasonable to seek a set of predictors that yields the highest adjusted R^2 value. Recall, R^2_{adj} provides a direct measure of the closeness of a model's fit to the given data. Thus, if the dataset satisfies the necessary assumptions and conditions, we expect such a model to offer the highest prediction accuracy. There are two algorithmic strategies commonly used to achieve a goal such as maximizing R^2_{adj}

- Backward elimination
- Forward selection

Both strategies, essentially, offer a systematic process for trying different combinations of the available predictors, and choosing the set that works best. To illustrate, let us consider a hypothetical dataset that has 4 variables available to choose as predictors in an MLR model. Let us denote these variables as x_1, x_2, x_3, x_4.

In the **backward elimination** method, we start with the full model that includes all 4 of the available predictors. The model looks like

$$\hat{y} = b_0 + b_1 x_1 + b_2 x_2 + b_3 x_3 + b_4 x_4$$

The slope coefficients $b_0, \ldots, b_4$ are computed by fitting the model to the given dataset in the usual way. The software output will tell us the adjusted R^2 for this model. Next, we seek to eliminate any variables that are not good predictors, using the following steps

Step 1: Re-fit the model after dropping one variable at time, and check the new R^2_{adj}. The table shows an example of the outcome we might see

x_1	x_2	x_3	x_4	new R^2_{adj}
$\times$	✓	✓	✓	$\downarrow$
✓	$\times$	✓	✓	$\uparrow$
✓	✓	$\times$	✓	$\downarrow$
✓	✓	✓	$\times$	$\downarrow$

The first row shows that when x_1 is dropped, the resulting new model has lower R^2_{adj}. Thus, x_1 is a good predictor, and we must retain it in the model. On the other hand, when x_2 is dropped, R^2_{adj} goes up. Thus, we drop x_2 from the model. The remaining rows show that x_3 and x_4 are also good predictors that we must keep. If dropping two (or more) predictors causes R^2_{adj} to increase, we only drop the one that yields the highest increase in R^2_{adj}.

Step 2: Our reduced model now looks like

$$\hat{y} = b_0 + b_1 x_1 + b_3 x_3 + b_4 x_4$$

where the slope coefficients are recomputed by fitting the model to the given dataset, resulting in a new R^2_{adj}. We repeat the process of dropping one variable at time, refitting the model, and checking the R^2_{adj}. The table shows an example of the outcome we might see

x_1	x_3	x_4	new R^2_{adj}
$\times$	✓	✓	$\downarrow$
✓	$\times$	✓	$\downarrow$
✓	✓	$\times$	$\uparrow$

In this example we drop x_4 from the model, since it causes R^2_{adj} to go up.

Step 3: Repeat the process – refit the reduced model to the given dataset, drop one variable at time, and check the R^2_{adj}.

x_1	x_3	new R^2_{adj}
$\times$	✓	$\downarrow$
✓	$\times$	$\downarrow$

Conclusion: The remaining predictors (x_1 and x_3) comprise the optimal model, since dropping either of them will decrease the adjusted R^2. Thus, our final model looks like

$$\hat{y} = b_0 + b_1x_1 + b_3x_3$$

The **forward selection** method of MLR model optimization starts with no predictors in the model. Thus, the initial model looks like

$$\hat{y} = b_0$$

Predictors are added to the model one by one, following a sequence of steps that assesses their effect on R^2_{adj}. The steps below show an example of how this process is carried out.

Step 1: Re-fit the model after adding one predictor at a time, and check the R^2_{adj}. For example, here is what it might look like:

	R^2_{adj} %
x_1	56.3
x_2	35.1
x_3	59.5
x_4	14.7

Since x_3 has the largest R^2_{adj}, we include it in the model. Thus, the model now looks like: $\hat{y} = b_0 + b_3x_3$, and the corresponding $R^2_{adj} = 59.5$ %.

Step 2: Repeat the process – add one (of the remaining) predictors at a time, refit the model, and check the R^2_{adj}. For example

	R^2_{adj} %
$x_1 + x_3$	64.7
$x_2 + x_3$	55.4
$x_4 + x_3$	54.8

We add x_1 to the model, since it increases the R^2_{adj} the most. The resulting model is $\hat{y} = b_0 + b_1x_1 + b_3x_3$, with $R^2_{adj} = 64.7$ %.

Step 3: Repeat the process – add one (of the remaining) predictors at a time and check R^2_{adj}. For example

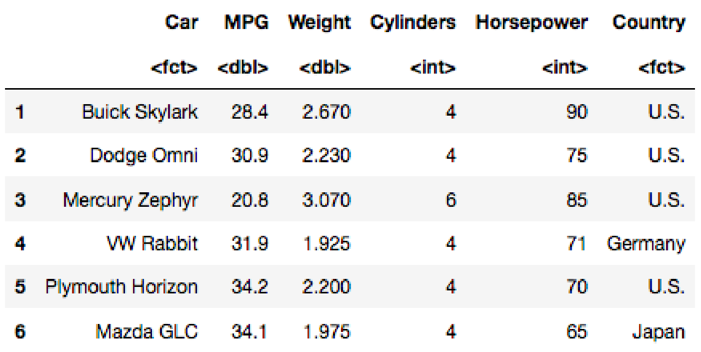
	R^2_{adj} %
$x_2 + (x_1 + x_3)$	59.1
$x_4 + (x_1 + x_3)$	54.9

Conclusion: None of the remaining predictors increases the R^2_{adj}. So the optimal model consists of the predictors x_1 and x_3.

Example:

The file `cars.csv` contains data on fuel efficiency and related variables for 34 cars manufactured in three different countries. The variables include MPG (miles per gallon), weight of the car (in tons), number of cylinders, horsepower, and country of origin. The header of the file, together with the R code that produced it are given below.

```
carsdat = read.csv(file="https://cs.earlham.edu/~pardhan/sage_
   and_r/cars.csv", header=TRUE, sep=",")
head(carsdat)
```



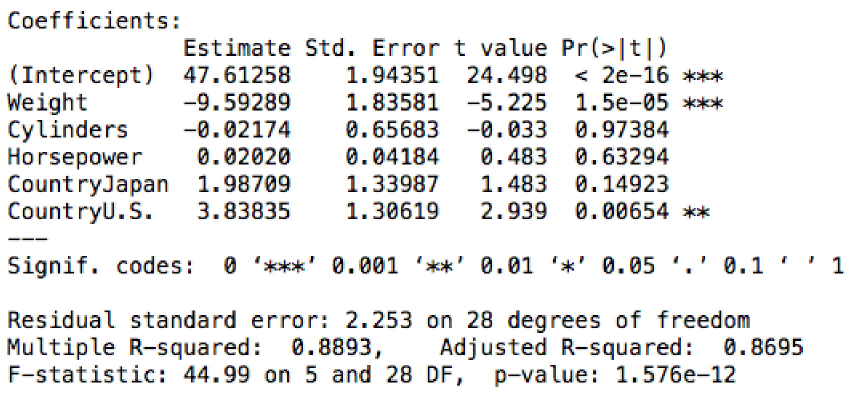
	Car	MPG	Weight	Cylinders	Horsepower	Country
	<fct>	<dbl>	<dbl>	<int>	<int>	<fct>
1	Buick Skylark	28.4	2.670	4	90	U.S.
2	Dodge Omni	30.9	2.230	4	75	U.S.
3	Mercury Zephyr	20.8	3.070	6	85	U.S.
4	VW Rabbit	31.9	1.925	4	71	Germany
5	Plymouth Horizon	34.2	2.200	4	70	U.S.
6	Mazda GLC	34.1	1.975	4	65	Japan

Assume these data satsify all the needed conditions for MLR, and find the optimal model for predicting MPG from the other variables.

Solution

We will take the optimal model to be the one that maximizes the adjusted R^2. Our response variable is `MPG` and the available predictor variables are `Weight`, `Cylinders`, `Horsepower`, and `Country`. We note that `Country` is a categorical predictor with 3 levels: Japan, U.S., Germany.

Using the backward elimination strategy, we begin by including all the available predictor variables. The resulting linear model is summarized in the following R output



```
Coefficients:
             Estimate Std. Error t value Pr(>|t|)    
(Intercept)  47.61258    1.94351  24.498  < 2e-16 ***
Weight       -9.59289    1.83581  -5.225  1.5e-05 ***
Cylinders    -0.02174    0.65683  -0.033  0.97384    
Horsepower    0.02020    0.04184   0.483  0.63294    
CountryJapan  1.98709    1.33987   1.483  0.14923    
CountryU.S.   3.83835    1.30619   2.939  0.00654 ** 
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 2.253 on 28 degrees of freedom
Multiple R-squared:  0.8893,	Adjusted R-squared:  0.8695 
F-statistic: 44.99 on 5 and 28 DF,  p-value: 1.576e-12
```

Observe that $R^2_{adj} = 0.8695$ for this full model.

Next, we refit the data after dropping one variable at a time The resulting R^2_{adj} values are shown in the table below

Dropped variable	R^2_{adj}
`Country`	0.8401
`Horsepower`	0.873
`Cylinders`	0.874
`Weight`	0.7512

As seen in the table, R^2_{adj} increases the most when `Cylinders` is dropped. The resulting model contains the 3 predictors `Weight`, `Horsepower`, `Country`, with $R^2_{adj} = 0.874$. Continuing the process, we refit the data after dropping one variable at a time and recompute R^2_{adj}

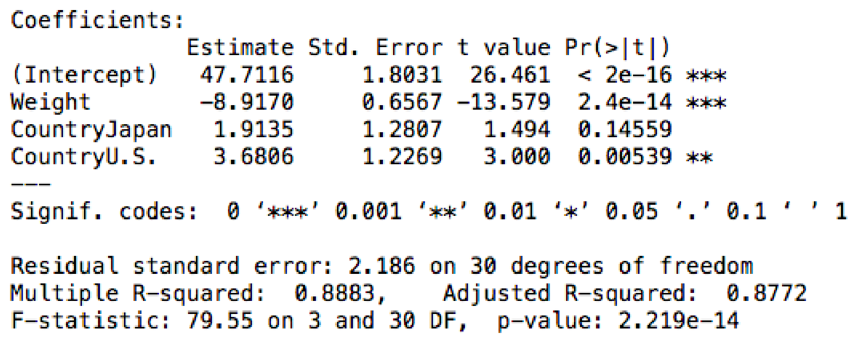
Dropped variable	R^2_{adj}
`Country`	0.8452
`Horsepower`	0.8772
`Weight`	0.7158

The R^2_{adj} increases slightly when `Horsepower` is dropped. Thus, we proceed to the next model, which consists of two predictors `Weight`, and `Country`, with $R^2_{adj} = 0.8772$. Again, we refit the data after dropping one variable and check R^2_{adj}

Dropped variable	R^2_{adj}
`Country`	0.8498
`Weight`	0.1505

The table shows that R^2_{adj} decreases if we drop either of the two remaining predictors. Thus, the final MLR model that maximizes R^2_{adj} consists of the predictors `Weight` and `Country`. Using the coefficient values given in the summary output below, the optimal model is

$$\boxed{\widehat{\texttt{MPG}} = 47.7116 - 8.917(\texttt{Weight}) + 1.9135(\texttt{Country:Japan}) + 3.6806(\texttt{Country:U.S.})}$$



```
Coefficients:
               Estimate Std. Error t value Pr(>|t|)
(Intercept)     47.7116     1.8031  26.461  < 2e-16 ***
Weight          -8.9170     0.6567 -13.579  2.4e-14 ***
CountryJapan     1.9135     1.2807   1.494  0.14559
CountryU.S.      3.6806     1.2269   3.000  0.00539 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 2.186 on 30 degrees of freedom
Multiple R-squared:  0.8883,	Adjusted R-squared:  0.8772
F-statistic: 79.55 on 3 and 30 DF,  p-value: 2.219e-14
```

Note that in this example, if we use the forward selection algorithm instead of backward elimination, we arrive at the exact same result. It is left as an exercise for the reader to carry out the forward selection algorithm and show this.

Remark: There are situations where forward selection might produce a slightly different model than backward elimination. In such situations we recommend picking the model with the larger R^2_{adj}.

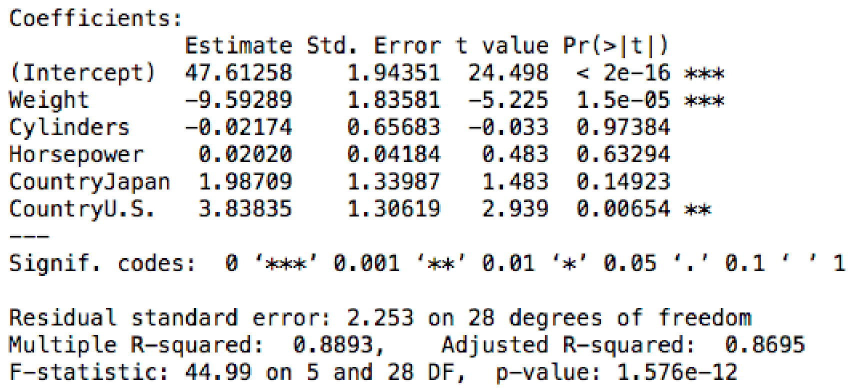
Other criteria for model optimization

At this point we have discussed optimal model selection based solely on one criterion, namely maximizing R^2_{adj}. Another commonly used optimization criterion involves minimizing the P-values. In this approach, variables are selected for inclusion in the model if they are significant predictors, as indicated by their P-value. The end result is a model that contains only those predictors whose P-value is below some chosen significance level. The process is carried out using algorithmic strategies similar to the backward elimination and forward selection procedures seen earlier.

To illustrate, consider the cars dataset seen in the previous example, and suppose we want to use backward elimination to select significant predictors with P-value below, say, 0.05. As before, we start with the full model that includes all four of the available predictors:

$$\begin{aligned}\widehat{\texttt{MPG}} = b_0 + b_1(\texttt{Weight}) + b_2(\texttt{Cylinders}) + b_3(\texttt{Horsepower})\\ +b_4(\texttt{Country:Japan}) + b_5(\texttt{Country:U.S.})\end{aligned}$$

The corresponding output, after fitting the model to the data, is



```
Coefficients:
              Estimate Std. Error t value Pr(>|t|)    
(Intercept)   47.61258    1.94351  24.498  < 2e-16 ***
Weight        -9.59289    1.83581  -5.225  1.5e-05 ***
Cylinders     -0.02174    0.65683  -0.033  0.97384    
Horsepower     0.02020    0.04184   0.483  0.63294    
CountryJapan   1.98709    1.33987   1.483  0.14923    
CountryU.S.    3.83835    1.30619   2.939  0.00654 ** 
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 2.253 on 28 degrees of freedom
Multiple R-squared:  0.8893,    Adjusted R-squared:  0.8695 
F-statistic: 44.99 on 5 and 28 DF,  p-value: 1.576e-12
```

The last column in the table of coefficients shows the P-value associated with each predictor. We look for predictors with P-value ≥ 0.05 and drop the one with the largest P-value. In this example, `Cylinders` will be dropped, since it has the highest P-value.

We refit the model with the remaining predictors and repeat the process

```
Coefficients:
             Estimate Std. Error t value Pr(>|t|)    
(Intercept)  47.63064    1.83284  25.987  < 2e-16 ***
Weight       -9.62411    1.54738  -6.220 8.73e-07 ***
Horsepower    0.01977    0.03907   0.506  0.61664    
CountryJapan  1.98090    1.30370   1.519  0.13948    
CountryU.S.   3.83481    1.27920   2.998  0.00553 ** 
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.214 on 29 degrees of freedom
Multiple R-squared:  0.8893,    Adjusted R-squared:  0.874 
F-statistic: 58.25 on 4 and 29 DF,  p-value: 1.916e-13
```

This time `Horsepower` is dropped, since it's P-value is the largest.

Continuing the process, we refit the model with the remaining predictors and obtain

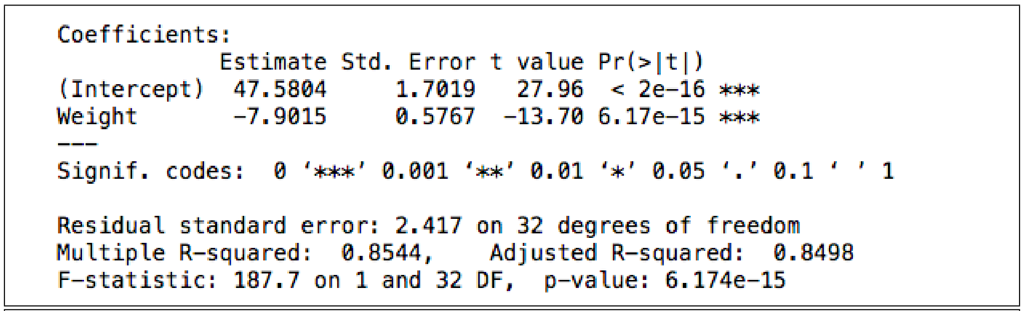
```
Coefficients:
             Estimate Std. Error t value Pr(>|t|)    
(Intercept)   47.7116     1.8031  26.461  < 2e-16 ***
Weight        -8.9170     0.6567 -13.579  2.4e-14 ***
CountryJapan   1.9135     1.2807   1.494  0.14559    
CountryU.S.    3.6806     1.2269   3.000  0.00539 ** 
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.186 on 30 degrees of freedom
Multiple R-squared:  0.8883,    Adjusted R-squared:  0.8772 
F-statistic: 79.55 on 3 and 30 DF,  p-value: 2.219e-14
```

Now we see both the remaining predictors are significant. Note that the categorical predictor `Country` is considered significant because it has at least one level (i.e., `Country:U.S.`) with P-value < 0.05. Although `Country:Japan` has large P-value, we cannot drop individual levels of a categorical predictor. We can only retain, or drop, the variable in its entirety. Thus our final model, based on retaining only the significant predictors, is

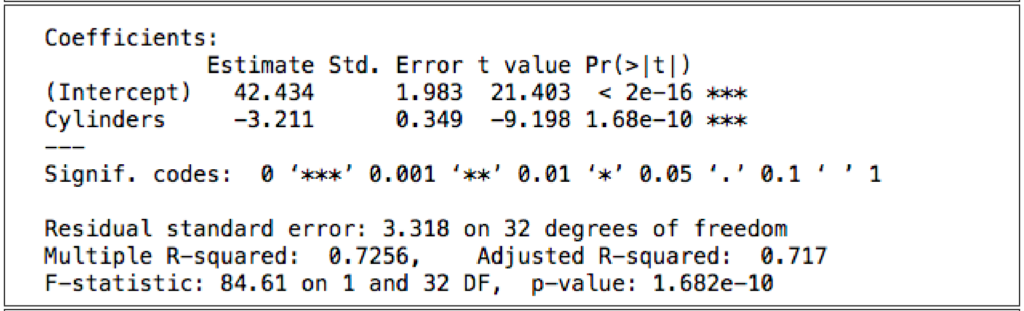
$$\boxed{\widehat{\texttt{MPG}} = 47.7116 - 8.917(\texttt{Weight}) + 1.9135(\texttt{Country:Japan}) + 3.6806(\texttt{Country:U.S.})}$$

As it turns out, this model is exactly the same as what we found earlier by maximizing R^2_{adj}. However, this will not always be the case. In general, the results from the P-value method will depend on the choice of significance level, and we would expect to get somewhat different models as we vary the significance level.

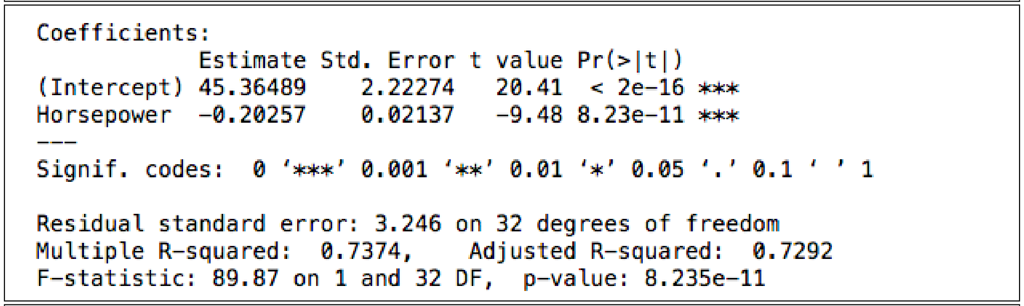
For completeness, let us look at how to implement the P-value criterion in a forward selection algorithm. To start the process a model with no predictors is assumed: $\widehat{\texttt{MPG}} = b_0$. Next, we augment the model with each predictor acting alone, carry out linear regression, and note the P-values



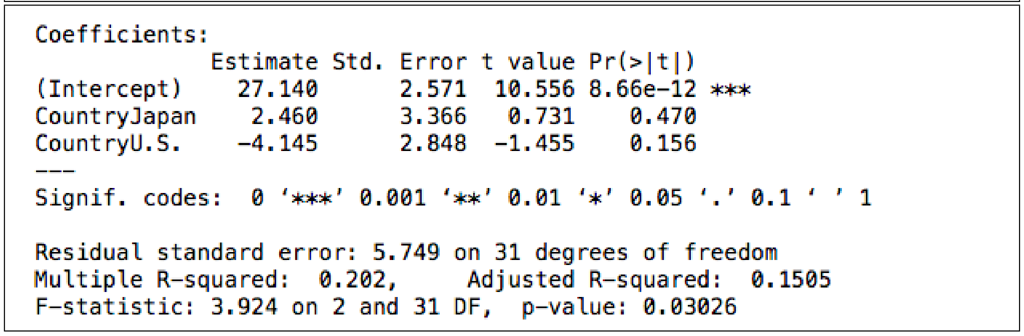
```
Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)  47.5804     1.7019   27.96  < 2e-16 ***
Weight       -7.9015     0.5767  -13.70 6.17e-15 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 2.417 on 32 degrees of freedom
Multiple R-squared:  0.8544,    Adjusted R-squared:  0.8498 
F-statistic: 187.7 on 1 and 32 DF,  p-value: 6.174e-15
```



```
Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)   42.434      1.983  21.403  < 2e-16 ***
Cylinders     -3.211      0.349  -9.198 1.68e-10 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 3.318 on 32 degrees of freedom
Multiple R-squared:  0.7256,    Adjusted R-squared:  0.717 
F-statistic: 84.61 on 1 and 32 DF,  p-value: 1.682e-10
```



```
Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) 45.36489    2.22274   20.41  < 2e-16 ***
Horsepower  -0.20257    0.02137   -9.48 8.23e-11 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 3.246 on 32 degrees of freedom
Multiple R-squared:  0.7374,    Adjusted R-squared:  0.7292 
F-statistic: 89.87 on 1 and 32 DF,  p-value: 8.235e-11
```



```
Coefficients:
             Estimate Std. Error t value Pr(>|t|)    
(Intercept)    27.140      2.571  10.556 8.66e-12 ***
CountryJapan    2.460      3.366   0.731    0.470    
CountryU.S.    -4.145      2.848  -1.455    0.156    
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 5.749 on 31 degrees of freedom
Multiple R-squared:  0.202,    Adjusted R-squared:  0.1505 
F-statistic: 3.924 on 2 and 31 DF,  p-value: 0.03026
```

The smallest P-value is `6.17e-15`, and it corresponds to the predictor `Weight`. So, we include it in the model, which then becomes: $\widehat{\texttt{MPG}} = 47.5804 - 7.9015(\texttt{Weight})$.

Next, we refit the model after adding each of the remaining predictors one at a time. As seen in the table of coefficients below, only the predictor `Country` yields a P-value < 0.05. We reiterate, a categorical predictor is considered significant if at least one of its levels yields a significant P-value. In this example, `Country:U.S.` has P-value < 0.05. Thus, our model now looks like: $\widehat{\texttt{MPG}} = 47.7116 - 8.917(\texttt{Weight}) + 1.9135(\texttt{Country:Japan}) + 3.6806(\texttt{Country:U.S.})$.

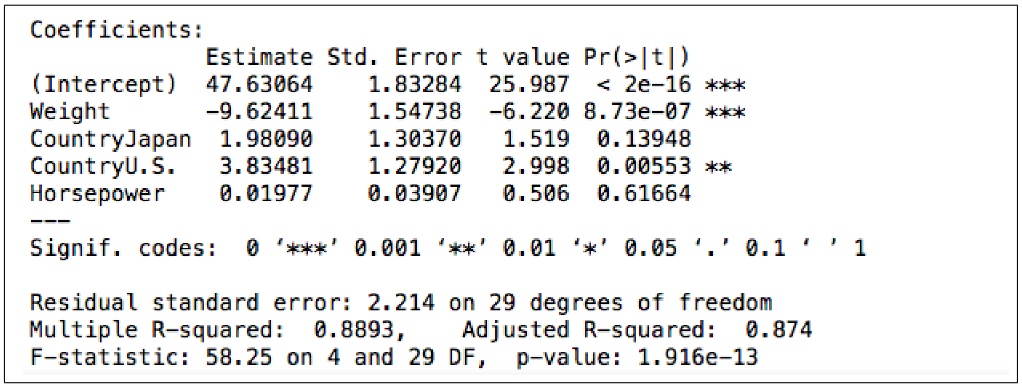
```
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 47.61179    1.76934  26.909  < 2e-16 ***
Weight      -8.01964    1.53134  -5.237 1.09e-05 ***
Cylinders    0.05638    0.67517   0.084    0.934
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 2.456 on 31 degrees of freedom
Multiple R-squared:  0.8544,	Adjusted R-squared:  0.845
F-statistic: 90.95 on 2 and 31 DF,  p-value: 1.07e-13
```

```
Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 47.624093   1.740462  27.363  < 2e-16 ***
Weight      -7.607007   1.522372  -4.997 2.17e-05 ***
Horsepower  -0.008804   0.042010  -0.210    0.835
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 2.454 on 31 degrees of freedom
Multiple R-squared:  0.8546,	Adjusted R-squared:  0.8452
F-statistic: 91.08 on 2 and 31 DF,  p-value: 1.05e-13
```

```
Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)   47.7116     1.8031  26.461  < 2e-16 ***
Weight        -8.9170     0.6567 -13.579  2.4e-14 ***
CountryJapan   1.9135     1.2807   1.494  0.14559
CountryU.S.    3.6806     1.2269   3.000  0.00539 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 2.186 on 30 degrees of freedom
Multiple R-squared:  0.8883,	Adjusted R-squared:  0.8772
F-statistic: 79.55 on 3 and 30 DF,  p-value: 2.219e-14
```

To continue the process, we add each of the remaining predictors to the model and check the P-values again.

```
Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  47.76936    1.89069  25.266  < 2e-16 ***
Weight       -9.08222    1.48062  -6.134  1.1e-06 ***
CountryJapan  1.89679    1.30910   1.449  0.15809
CountryU.S.   3.67995    1.24750   2.950  0.00623 **
Cylinders     0.07698    0.61590   0.125  0.90139
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 2.223 on 29 degrees of freedom
Multiple R-squared:  0.8884,	Adjusted R-squared:  0.873
F-statistic: 57.71 on 4 and 29 DF,  p-value: 2.158e-13
```



```
Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  47.63064    1.83284  25.987  < 2e-16 ***
Weight       -9.62411    1.54738  -6.220 8.73e-07 ***
CountryJapan  1.98090    1.30370   1.519  0.13948
CountryU.S.   3.83481    1.27920   2.998  0.00553 **
Horsepower    0.01977    0.03907   0.506  0.61664
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 2.214 on 29 degrees of freedom
Multiple R-squared:  0.8893,    Adjusted R-squared:  0.874
F-statistic: 58.25 on 4 and 29 DF,  p-value: 1.916e-13
```

As seen in the tables, none of these predictors produces a significant P-value. Thus, we drop them, and our final optimized model is

$$\boxed{\widehat{\texttt{MPG}} = 47.7116 - 8.917(\texttt{Weight}) + 1.9135(\texttt{Country:Japan}) + 3.6806(\texttt{Country:U.S.})}$$

Now that we have seen how to use R^2_{adj} and P-values for optimal model selection, which method is better? As you might expect, there is no simple or universally accepted answer to this question. Generally, in situations where prediction accuracy is important, the R^2_{adj} method is preferred. This is often the case in large data applications such as those seen in many data science and data analytics projects. However, the P-value method is preferable in situations where identifying the most influential predictors is an important goal. This may occur, for example, when a researcher is interested in studying the impact of specific variables within a dataset or model.

In closing, we note that there are several other optimal model selection criteria in the literature. Some of these include AIC, BIC, etc. **(add brief discussion of these later?)**

1.5 Polynomial regression

Given the importance of satisfying the linearity condition in all the regression methods discussed thus far, an obvious question that readers likely have is: What are my options if my relationships are not linear? It turns out there are a variety of nonlinear functions used for fitting regression models in applications that warrant it. Perhaps the simplest of these is regression with polynomial functions, which is sometimes also called curvilinear regression. Such functions are particularly straightforward to implement within a linear regression framework, because the unknown slope coefficents remain in linear form. As a result, the linear solution techniques developed previously are sufficient for computing the model parameter values using a least squares minimization process.

To illustrate, consider a cubic model for fitting the relationship between a response variable y and a predictor variable x. The predicted response would then be

$$\hat{y} = b_0 + b_1 x + b_2 x^2 + b_3 x^3 \tag{1.11}$$

Let (x_i, y_i), $i = 1, 2, \ldots, n$, denote the pairwise data values for which we want to find the best fit. Following the usual least squares process, we define the residuals

$$e_i = y_i - \hat{y}_i = y_i - (b_0 + b_1 x_i + b_2 x_i^2 + b_3 x_i^3) \tag{1.12}$$

and the cost function

$$f(b_0, b_1, b_2, b_3) = \sum_1^n e_i^2 = \sum_1^n \left[y_i - (b_0 + b_1 x_i + b_2 x_i^2 + b_3 x_i^3)\right]^2 \tag{1.13}$$

The coefficients $b_0, \ldots, b_3$ are found by minimizing f, which requires setting its partial derivatives to 0

$$\begin{aligned}
\frac{\partial f}{\partial b_0} &= -2\sum_1^n \left[y_i - (b_0 + b_1 x_i + b_2 x_i^2 + b_3 x_i^3)\right] = 0 \\
\frac{\partial f}{\partial b_1} &= -2\sum_1^n \left[y_i - (b_0 + b_1 x_i + b_2 x_i^2 + b_3 x_i^3)\right] x_i = 0 \\
\frac{\partial f}{\partial b_2} &= -2\sum_1^n \left[y_i - (b_0 + b_1 x_i + b_2 x_i^2 + b_3 x_i^3)\right] x_i^2 = 0 \\
\frac{\partial f}{\partial b_3} &= -2\sum_1^n \left[y_i - (b_0 + b_1 x_i + b_2 x_i^2 + b_3 x_i^3)\right] x_i^3 = 0
\end{aligned}$$

Since the values of (x_i, y_i) are known, this yields a system of four linear algebraic equations in the four unknowns $b_0, \ldots, b_3$. Thus, the computational process for doing cubic regression with one predictor is about the same as solving a multilinear regression problem with three predictors. In fact, to take that analogy one step further, let us rewrite equation (1.11) as

$$\hat{y} = b_0 + b_1 u + b_2 v + b_3 w \tag{1.14}$$

where $u = x, v = x^2, w = x^3$. Equation (1.14) is, essentially, a linear model for y that contains 3 predictors, u, v, w. For all practical purposes, we can solve the cubic regression problem by carrying out multilinear regression on (1.14). This idea generalizes to polynomials of any degree. For example, a polynomial model of degree k with one predictor can be transformed to an equivalent MLR problem containing k predictors. It is important to note, however, that polynomials of degree higher than 2 or 3 are rarely ever used in regression. This is because high-degree polynomials are prone to high-amplitude oscillatory behavior.

Viewing polynomial regression as a special form of multilinear regression is also helpful for understanding the assumptions and conditions that must be satisfied. Recall, a valid MLR model must satisfy: Linear relationship between the response and each predictor, normal residuals, constant variance of residuals, and independent observations. In polynomial regression, the linearity condition is replaced by one that requires the response variable to have a linear or a polynomial dependence on each predictor. The other conditions remain the same.

Example:

The Population Reference Bureau, an international, non-profit, public policy development organization compiles data on a broad range of demographic, health and other characteristics of the world's population. One of their datasets, contained in the file `prb_data_mlr.csv`, provides information on the average income in a sample of 190 countries, together with the corresponding life expectancy of its people. Here is the file header and the associated R code

```
prbdat = read.csv(file="https://cs.earlham.edu/~pardhan/sage_
   and_r/prb_data_mlr.csv", header=TRUE, sep=",")
head(prbdat,4)
```

	FIPS	Name	Type	TimeFrame	GNI	LE_female	LE_male
	<fct>	<fct>	<fct>	<int>	<dbl>	<int>	<int>
1	LR	Liberia	Country	2017	0.70	64	62
2	CF	Central African Republic	Country	2017	0.70	54	50
3	CD	Congo, Dem. Rep.	Country	2017	0.73	61	58
4	BI	Burundi	Country	2017	0.77	62	59

Key variables of interest include `GNI, LE_female, LE_male`. Here `GNI` represents the Gross National Income per capita (in 1000 US dollars), `LE_female` and `LE_male` represent the Life Expectancy at birth of females and males (in years).

Suppose we want a model for predicting `LE_female` using `GNI` as the predictor. Here is a scatter plot of the data

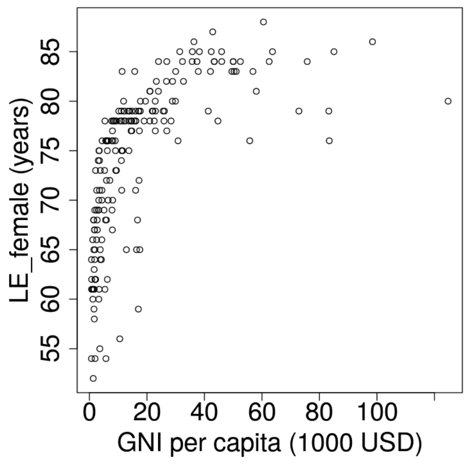


Figure 1.4: Scatterplot of female life expectancy vs GNI.

The relationship is, clearly, far from linear. But, we will still go ahead and fit a straight line model, so that it serves as a baseline for comparing other models. Of course, we will also keep in mind that the resulting model has no predictive value, as it fails to satisfy at least one of the required conditions (i.e., linear relationship). Accordingly, here is the regression output

```
Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)  69.7451     0.6299  110.72   <2e-16 ***
GNI           0.2434     0.0224   10.86   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 6.383 on 187 degrees of freedom
Multiple R-squared:  0.3869,    Adjusted R-squared:  0.3836 
F-statistic:   118 on 1 and 187 DF,  p-value: < 2.2e-16
```

Thus, the straight line model is

$$\widehat{\texttt{LE\_female}} = 69.7451 + 0.2434\,(\texttt{GNI}) \tag{1.15}$$

with $R^2_{adj} = 0.3836$. Incidentally, this R^2_{adj} value is not too bad for applications of this type, given the challenges and approximations inherent in collecting accurate demographic and macroeconomic data. However, the fact remains that the model is not valid, since it fails to satisfy the theoretical conditions.

Next, let us consider fitting a polynomial model to the same data. The first question that arises is, what degree of polynomial should we choose? It is strongly recommended that the degree be kept as low as possible to avoid the extreme oscillations that can arise with high degree polynomials. For this reason, quadratics and cubics tend to be the most common choice when doing polynomial regression.

In the present example, a quadratic model would have the form

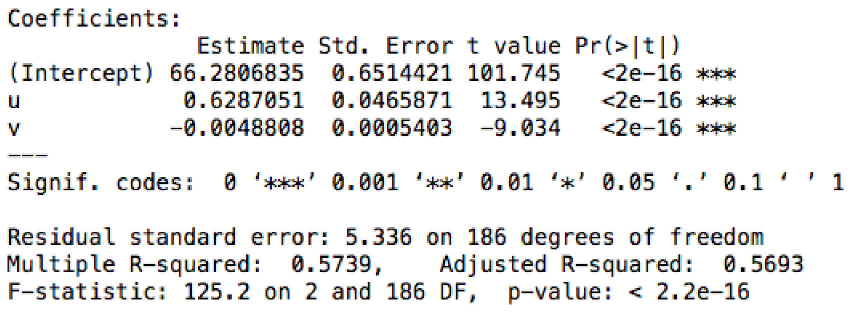
$$\widehat{\texttt{LE\_female}} = b_0 + b_1(\texttt{GNI}) + b_2(\texttt{GNI})^2 \tag{1.16}$$

To carry out the regression we treat this as a multilinear problem in two predictors: $u = \texttt{GNI}, v = (\texttt{GNI})^2$. The corresponding R code, with summary of regression output and diagnostic plots are shown below

```
# Read datafile and store data in a dataframe
prbdat = read.csv(file="https://cs.earlham.edu/~pardhan/sage_
   and_r/prb_data_mlr.csv", header=TRUE, sep=",")

# Fit quadratic polynomial regression model
y = prbdat$LE_female     # response variable
u = prbdat$GNI           # predictor 1
v = u*u                  # predictor 2
prbout = lm( y ~ u+v)    # regression using lm() function

# Print regression summary
summary(prbout)

# Setup larger margins for plots
par(mar = c(5,5,5,5)) # sets the margins of plot to be bigger
```

```
# Scatter plot original data and quadratic model
plot(prbdat$LE_female ~ prbdat$GNI, xlab="GNI␣per␣capita␣(1000␣
   USD)", ylab="LE_female␣(years)", cex.lab=2, cex.axis=2)
lines(prbout$fitted ~ prbdat$GNI, col="#FF0000", lwd=5)

# Plot residuals diagnostics
qqnorm(prbout$residuals, ylab="Residuals", main="", cex.lab=2,
   cex.axis=2)
hist(prbout$residuals, xlab="Residuals", breaks=6, main="", cex
   .lab=2, cex.axis=2)
plot(prbout$residuals ~ prbout$fitted, xlab="Predicted␣values",
    ylab="Residuals", main="", cex.lab=2, cex.axis=2)
```

The corresponding output from R:



```
Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept) 66.2806835  0.6514421 101.745   <2e-16 ***
u            0.6287051  0.0465871  13.495   <2e-16 ***
v           -0.0048808  0.0005403  -9.034   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 5.336 on 186 degrees of freedom
Multiple R-squared:  0.5739,    Adjusted R-squared:  0.5693
F-statistic: 125.2 on 2 and 186 DF,  p-value: < 2.2e-16
```

The regression output indicates that the quadratic model is

$$\widehat{\texttt{LE\_female}} = 66.28 + 0.6287\ (\texttt{GNI}) - 0.00488\ (\texttt{GNI})^2 \tag{1.17}$$

Notice that R^2_{adj} is now 0.5693, indicating a much better fit than the straight line model, for which the R^2_{adj} value was 0.3836. To check whether the conditions are met, let us first look at the diagnostic plots in Figure 1.5. Plot (a) suggests it is acceptable to treat the dependence between `LE_female` and `GNI` as quadratic. Plots (b)-(c) show that the distribution of residuals is not perfect, but is close enough to normal, especially for this sample size ($n = 190$). In plot (d) the variance of residuals seems to decrease as we go from lower to higher predicted values. This violates the constant variance condition. There are a couple of high `GNI` outliers in the dataset, and it is possible their removal will help with this problem. Lastly, for the independent observations condition, in the context of this application we will assume it is met if these data can be considered representative of the world's countries. It is not clear if that is true here. To summarize, the constant variance and independent observations conditions may not be met by this dataset.

To wrap up this example, let us consider fitting a cubic model of the form

$$\widehat{\texttt{LE\_female}} = b_0 + b_1(\texttt{GNI}) + b_2(\texttt{GNI})^2 + b_3(\texttt{GNI})^3 \tag{1.18}$$

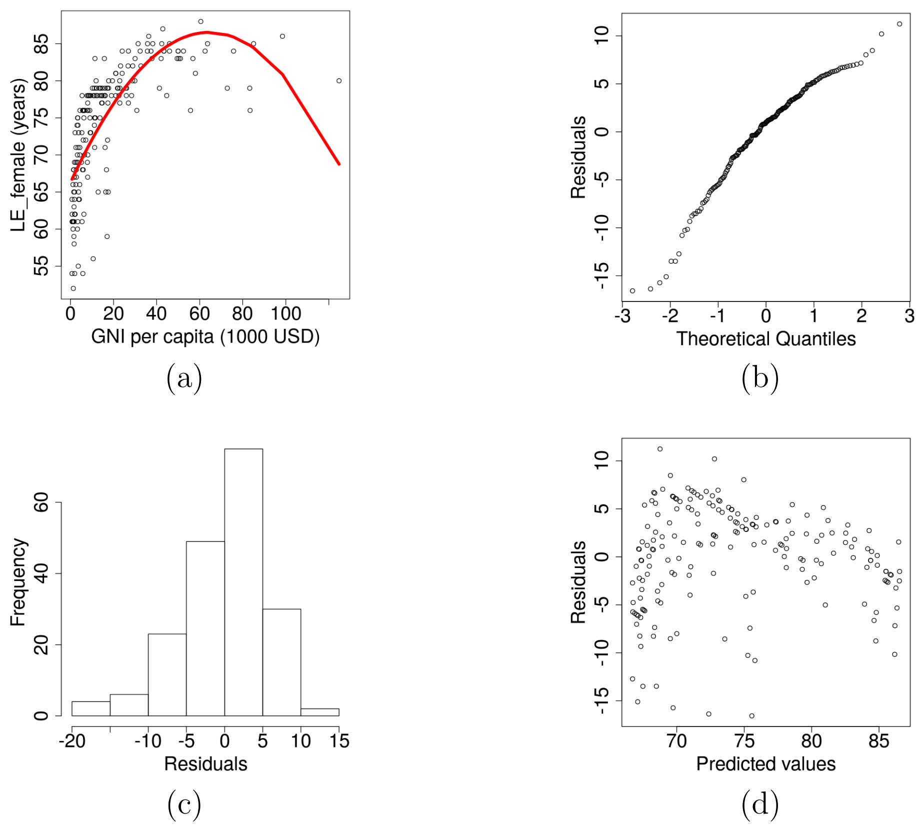
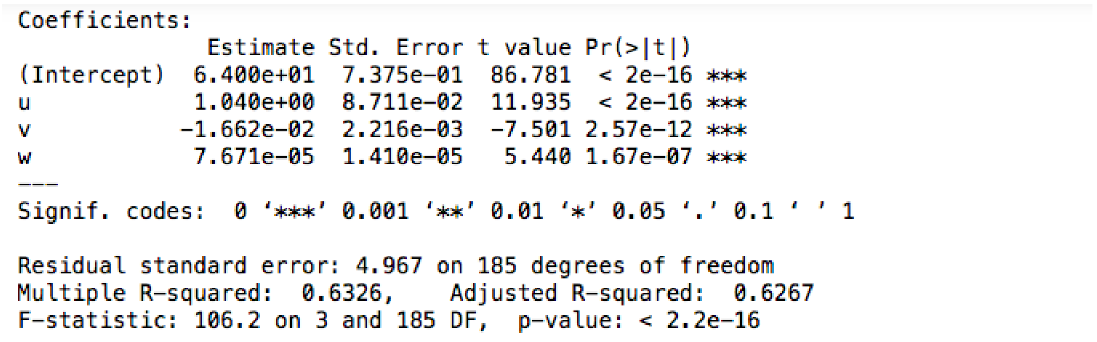


Figure 1.5: Diagnostics for `LE_female` vs `GNI` quadratic model: (a) Fitted model (in red). (b) Residuals normal probability plot. (c) Residuals histogram. (d) Residuals vs predicted.

The R code, regression summary and diagnostic plots are shown below.

```
# Read datafile and store data in a dataframe
prbdat = read.csv(file="https://cs.earlham.edu/~pardhan/sage_
   and_r/prb_data_mlr.csv", header=TRUE, sep=",")

# Fit cubic regression model & generate plots
y = prbdat$LE_female
u = prbdat$GNI
v = u*u
w = u*v
prbout3 = lm( y ~ u+v+w)
summary(prbout3)
plot(prbdat$LE_female ~ prbdat$GNI, xlab="GNI␣per␣capita␣(1000␣
   USD)", ylab="LE_female␣(years)", cex.lab=2, cex.axis=2)
lines(prbout3$fitted ~ prbdat$GNI, col="#FF0000", lwd=5)
qqnorm(prbout3$residuals, ylab="Residuals", main="", cex.lab=2,
    cex.axis=2)
```

```r
hist(prbout3$residuals, xlab="Residuals", breaks=6, main="",
   cex.lab=2, cex.axis=2)
plot(prbout3$residuals ~ prbout3$fitted, xlab="Predicted␣values
   ", ylab="Residuals", main="", cex.lab=2, cex.axis=2)
```



```
Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  6.400e+01  7.375e-01  86.781  < 2e-16 ***
u            1.040e+00  8.711e-02  11.935  < 2e-16 ***
v           -1.662e-02  2.216e-03  -7.501 2.57e-12 ***
w            7.671e-05  1.410e-05   5.440 1.67e-07 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 4.967 on 185 degrees of freedom
Multiple R-squared:  0.6326,	Adjusted R-squared:  0.6267
F-statistic: 106.2 on 3 and 185 DF,  p-value: < 2.2e-16
```

According to the regression summary, the fitted cubic model is

$$\widehat{\texttt{LE\_female}} = 64.0 + 1.04\,(\texttt{GNI}) - 0.0166\,(\texttt{GNI})^2 + 7.67 \times 10^{-5}(\texttt{GNI})^3 \tag{1.19}$$

Since $R^2_{adj} = 0.6267$, the cubic model offers a somewhat better fit than the quadratic. This can also be seen in Figure 1.6:(a), which displays the cubic fit on a scatter plot of the data values. However, the rest of the diagnostic plots in Figure 1.6 suggest the behavior of the residuals is no better than in the quadratic case. In fact, the normal distribution plots are even less satisfactory than the corresponding ones for the quadratic model. The bottom line is, both models fall short on meeting the theoretical conditions, and so it would not be advisable to use them for making predictions.

The overall strategy demonstrated in this example readily applies to any other polynomial regression problem in one predictor variable. Key points to keep in mind include

- Check the assumptions/conditions.
- Use low degree polynomials.
- Use complete polynomials – i.e., include terms of all orders upto the polynomial degree.

Polynomial regression with more than one predictor is done in a very similar way, but the functions get rapidly cumbersome as the number of predictors increase. For example, if there are two predictors (say, u, v) and we use a 2nd degree polynomial, the predicted response would have the form

$$\hat{y} = b_0 + b_1 u + b_2 v + b_3 u^2 + b_4 v^2 + b_5 uv \tag{1.20}$$

and for a 3rd degree polynomial its form would be

$$\hat{y} = b_0 + b_1 u + b_2 v + b_3 u^2 + b_4 v^2 + b_5 uv + b_6 u^3 + b_7 v^3 + b_8 u^2 v + b_9 uv^2 \tag{1.21}$$

Note that the problem is still linear in the unknown coefficients ($b_1, b_2, \ldots,$ etc.), and so it can be readily transformed into a multilinear regression framework by introducing new

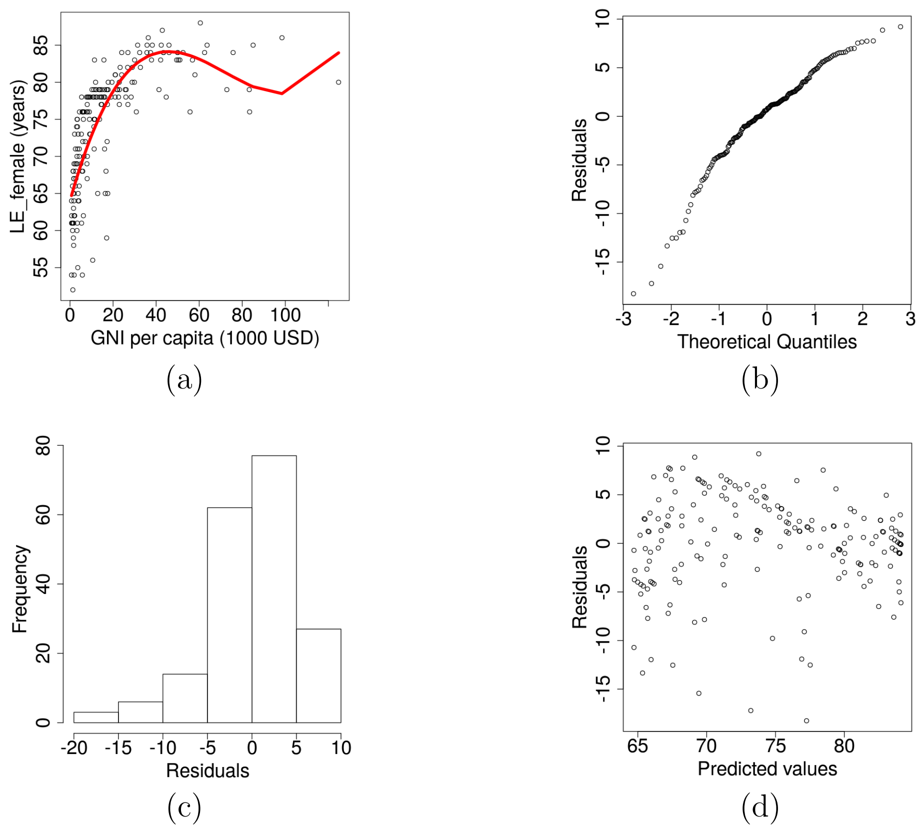


Figure 1.6: Diagnostic plots for cubic model: (a) Fitted model (in red). (b) Residuals normal probability plot. (c) Residuals histogram. (d) Residuals vs predicted.

variables to replace the nonlinear terms. For example, (1.20) can be written in the linear form

$$\hat{y} = b_0 + b_1 u + b_2 v + b_3 w + b_4 x + b_5 z \tag{1.22}$$

by introducing the variables $w = u^2, x = v^2, z = uv$. This effectively transforms the original 2-predictor nonlinear model into a 5-predictor linear model.

To summarize, regression using polynomial functions is relatively straightforward to set up and implement, because the underlying models remain linear in the unknown parameters. Thus, the computational procedures are very similar to those of multilinear regression. For this reason, it is routine practice to use the same software functions and utilities for doing polynomial regression.

(What is missing: (1) meaning of coefficients; (2) inferences; (3) (maybe?) example to demonstrate multiple predictors.)

1.6 Logistic regression

Earlier in this module we promised to address the topic of categorical response variables, which comprise an important application class in their own right. Logistic regression is a key step in that direction, as it provides a strategy to handle binary response variables. For example, suppose we want to determine whether an attempted credit card transaction online is a fraud or not. Here the response we seek is simply "yes" or "no" to the fraud question, while the inputs might include information about the transaction, such as the amount, the product being purchased, the user's location, name, purchase history, etc. This is essentially a classification problem, wherein a bunch of inputs are evaluated to determine whether the output belongs to class A, or not. Logistic regression is one of the tools used for performing this type of analysis.

To begin with, it is extremely helpful to invest some effort understanding the mathematical function that is at the foundation of logistic regression. Accordingly, consider

$$y = \frac{1}{1 + e^{-(b_0 + b_1 x)}} \tag{1.23}$$

with x = independent variable, y = response variable, b_0, b_1 = some given constants. The reader will, of course, notice the straight line form of the exponent in the denominator. Let us explore the graphical properties of y, and their dependence on the parameters b_0, b_1. Figure 1.7(a) shows the effect of varying b_1, while keeping b_0 fixed at 0. Conversely, in Figure 1.7(b), we see the effect of varying b_0, while keeping b_1 fixed.

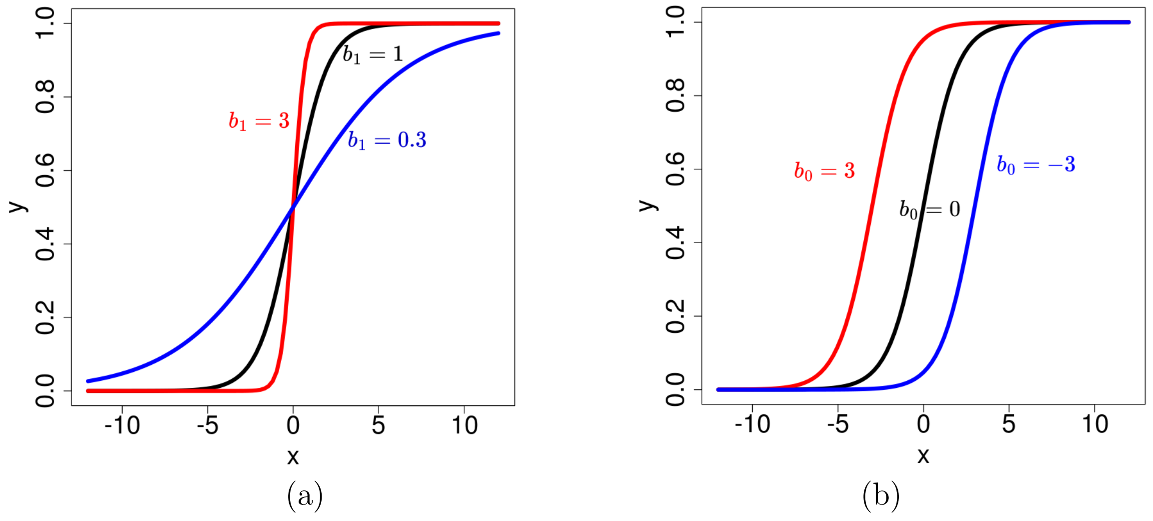


Figure 1.7: The function $y = \frac{1}{1+e^{-(b_0+b_1 x)}}$: (a) Effect of varying b_1, with $b_0 = 0$. (b) Effect of varying b_0, when $b_1 = 1$.

Judging from the graphs, what can you tell about the range of y? Does it depend on the values of b_0, b_1? How does b_1 affect the behavior of y? Let's summarize our observations

- The range is $0 < y < 1$, regardless of the values of b_0, b_1. In fact, it is easy to show that if $b_1 > 0$, then $\lim_{x \to -\infty} y = 0$, $\lim_{x \to \infty} y = 1$.
- As the magnitude of b_1 increases, the graph gets steeper when it transitions from 0 to 1. Although not seen in these graphs, we note that if $b_1 < 0$, the curve flips, and the y values decrease from 1 to 0.
- Changing b_0 shifts the curve horizontally, either left or right.

Putting these observations together, b_0 determines the x-location where the curve transitions from low to high y-values, and b_1 controls the steepness of the transition. This type of behavior is particularly suitable for doing binary classification, as logistic regression is designed to do. The idea is to simply treat the y-value as the probability of belonging to the class of interest, for example, a fraudulent credit card transaction. This works well as a classification mechanism because there is a sharp transition from very low probability to very high probability, when x crosses a threshold value.

Moving on to the technical details, equation (1.23) can be algebraically rearranged as

$$\frac{y}{1-y} = e^{b_0 + b_1 x} \qquad \text{OR} \qquad \ln\left[\frac{y}{1-y}\right] = b_0 + b_1 x$$

The expression $\ln(\frac{y}{1-y})$ is known as the logit transformation of y. If we define $z = \ln(\frac{y}{1-y})$, this last equation represents a straight line relationship between x and z. Logistic regression is based on this idea, and from a big-picture standpoint, one may view it as a linear model for predicting the logit of the response probabilities. However, a word of caution might be in order, because there are many important differences from linear regression, both in concept and in implementation. We will address some of these differences here, and leave the rest for the references.

Let us consider the most essential questions from the point of view of practical, hands-on modeling. Suppose we want to fit a logistic regression model to a dataset containing n observations of a pair of (x, y) variables. Assume the predictor x is a numerical variable, and the response y is a binary/indicator variable whose values are 0 or 1. Let p denote the probability that $y = 1$, given some x value. The logistic model for predicting p is

$$z = b_0 + b_1 x, \quad \text{were } z = \ln\left[\frac{p}{1-p}\right] \tag{1.24}$$

Here b_0, b_1 are coefficients whose values are to be determined by fitting the model to the input dataset. Before we do that, let us look at the concept of "the odds of an event occuring," which arises frequently in statistics. We define it as

$$\text{odds of A} = \frac{\text{probability that A occurs}}{\text{probability that A does not occur}} = \frac{P(A)}{1 - P(A)}$$

For example, the odds of tossing a coin twice and getting both heads are $\frac{1/4}{1-1/4} = \frac{1}{3}$. We say, the odds of getting both heads are 1 to 3. Similarly, the odds of NOT getting a 6 if we roll a fair 6-sided die are $\frac{5/6}{1-5/6}$, or 5 to 1. Notice that the numerical value of odds can range from 0 to ∞, although probability values only range from 0 to 1. When we take the natural log of the odds, we get what is known as the "log odds." Thus, log odds range from $-\infty$ to ∞, which makes it possible to model them with a straight line.

Returning to equation (1.24), notice that it is simply a linear model for predicting the log odds of $y = 1$ based on the value of x. In fact, these ideas readily generalize to multiple predictors, and the general form of the logistic regression model for predicting p, the probability that $y = 1$, with k predictors is

$$z = b_0 + b_1x_1 + b_2x_2 + \ldots + b_kx_k, \quad \text{were } z = \ln\left[\frac{p}{1-p}\right] \tag{1.25}$$

As usual, $(x_1, \ldots, x_k)$ denote the predictors, and $(b_0, b_1, \ldots, b_k)$ the slope coefficients. To predict the response $\hat{y}$ we first find z, and then compute p by inverting the formula $z = \ln[p/(1-p)]$. The resulting p will be a value between 0-1, and represents the probability of belonging to the category $y = 1$. Typically, if $p < 0.5$, the predicted response is $\hat{y} = 0$; otherwise, it is $\hat{y} = 1$.

So, how do we compute the slope coefficients $b_0, \ldots, b_k$? This is the same as asking, how do we best fit equation (1.25) to the given data? Unfortunately, the simple least squares approach we used in linear regression does not work here. The key difficulty is that we cannot plug data values into our model equation and compute residuals, as we do in linear regression. This is because our data does not provide the values of p, the probabilities corresponding to each set of observations. Moreover, we cannot use the observed y-values in the place of p, because they will result in $z = +\infty$, or $z = -\infty$. Consequently, a new optimization approach is needed to fit the model to the data, in order to estimate the values of $b_0, \ldots, b_k$. In this new approach, the quality of the model's fit is measured by the "likelihood" function, which is defined as

$$L = \prod_{i=1}^{n} \hat{p}_i^{y_i}\,(1-\hat{p}_i)^{1-y_i} \quad \text{were } \hat{p}_i = \frac{1}{1+e^{-\hat{z}_i}} \tag{1.26}$$

Subscript i represents the i^{th} data point in the sample, n is the sample size, y_i is the i^{th} response value, $\hat{p}_i$ is the predicted response, and $\hat{z}_i$ is obtained from equation (1.25) using the i^{th} set of predictor values. Note that $\hat{z}_i$ contains the unknowns $b_0, \ldots, b_k$, and so it cannot be explicitly computed. Consequently, for a given input dataset, L is a nonlinear function of the parameter set $\{b_0, \ldots, b_k\}$.

It can be shown that the numerical value of the likelihood, L, increases as the model in (1.25) gets closer to fitting the data. Thus, the goal of the optimization strategy in logistic regression is to find the set $\{b_0, \ldots, b_k\}$ that maximizes the likelihood. In practice, it is usual to take the log of equation (1.26) and maximize the log-likelihood instead. This transforms the product sequence into a sequence of sums, and we get

$$\ln(L) = \sum_{i=1}^{n} \left[y_i \ln(\hat{p}_i) + (1-y_i)\ln(1-\hat{p}_i)\right] \tag{1.27}$$

There is no closed-form solution for maximizing $\ln(L)$, and some form of iterative strategy is typically employed to estimate the optimal values of $\{b_0, \dots, b_k\}$. A detailed exposition of the optimization process is beyond the scope of the present work.

Some additional concepts (optional reading)

It is useful to draw some parallels between these ideas and the usual least squares approach used in linear regression. The quantity $-2\ln(L)$ is known as the **deviance** of the model, and is analogous to the the sum of the squares of the residuals (RSS) in linear regression. When we maximize the log-likelihood in logistic regression, we essentialy minimize the deviance. This is analogous to minimizing the RSS in linear regression.

Residuals are defined in a variety of different ways in logistic regression. The simplest of these is the **raw residual**, which is essentially identical to how residuals are defined in linear regression

$$r_i = y_i - \hat{p}_i \quad \text{(raw residual)} \tag{1.28}$$

Here y_i is the observed response at the i^{th} data point, and $\hat{p}_i$ the predicted probability. When this residual is scaled by the standard deviation of its distribution, we get the **Pearson residual**

$$\rho_i = \frac{y_i - \hat{p}_i}{\sqrt{\hat{p}_i(1-\hat{p}_i)}} \tag{1.29}$$

The **deviance residual** is another common variant, and it is defined such that the sum of its squares equals the deviance of the model

$$d_i = \text{sign}\{y_i - \hat{p}_i\}\sqrt{-2[y_i\ln(\hat{p}_i) + (1-y_i)\ln(1-\hat{p}_i)]} \tag{1.30}$$

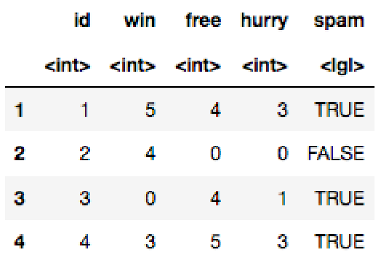
Deviance residuals are, in fact, what the optimization algorithm tries to minimize.

Another important concept is that of **null deviance**, which is a measure of how well a model containing only an intercept term predicts the response. In contrast, the **residual deviance** measures how well the response is predicted when the model includes all the predictors of interest. For a good model, we would expect the residual deviance to be much smaller than the null deviance. In fact, the magnitude of the difference between these two deviances is often used as a χ^2 test statistic for performing a hypothesis test on the significance of the model.

To illustrate the use of these ideas for performing binary classification, let us consider an example. Email spam filters are basically a mechanism for classifying mail coming into a user's inbox as "spam" or "not spam" based on algorithmic checks of the source and contents of the message. Modern-day spam filters are extremely complex, and incorporate numerous input variables to perform spam classification. To keep things clear and simple, in this example we consider a "toy" version of a spam filter that only incorporates 3 input variables, based on the frequency of occurrence of certain keywords in the email message. The keywords are: (1)

win, (2) free, and (3) hurry. We will construct the spam filter by fitting a logistic regression model to a training dataset consisting of 200 email messages whose keyword count and spam status are known. Accordingly, here are the first few lines of the dataset, together with the associated R code

```
emaildat = read.csv(file="https://cs.earlham.edu/~pardhan/sage_
   and_r/mock_email_data.csv", header=TRUE, sep=",")
head(emaildat,4)
```



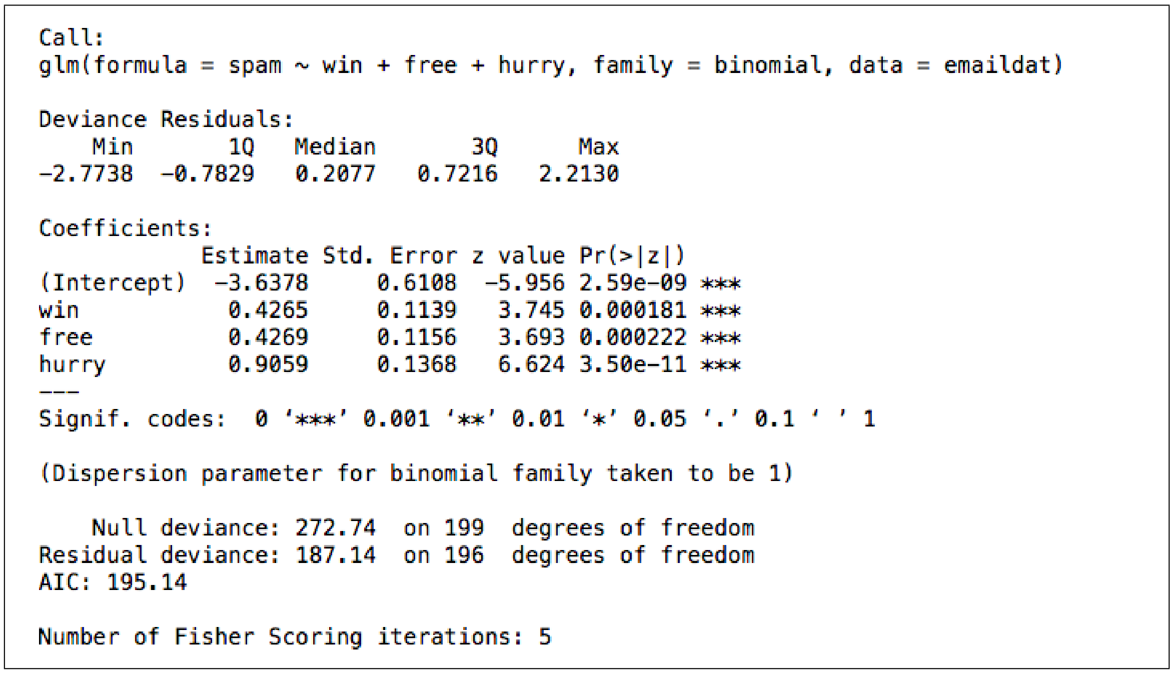
	id	win	free	hurry	spam
	<int>	<int>	<int>	<int>	<lgl>
1	1	5	4	3	TRUE
2	2	4	0	0	FALSE
3	3	0	4	1	TRUE
4	4	3	5	3	TRUE

We assume this dataset satisfies the conditions for logistic regression, which will be discussed in detail later. The response variable is `spam`, and it contains the binary categories `TRUE`/`FALSE`. The predictor variables are numerical, and are named `win`, `free` and `hurry`. Thus, the logistic model for predicting the probability the message is spam has the form

$$\begin{aligned}\hat{z} &= b_0 + b_1(\texttt{win}) + b_2(\texttt{free}) + b_3(\texttt{hurry}) \\ \hat{p} &= \frac{1}{1+e^{-\hat{z}}}\end{aligned} \tag{1.31}$$

We fit the model to the given dataset using the R function `glm()`, which stands for `generalized linear model`. The usage of this function is very similar to the `lm()` function that we've been using for multilinear regression, except we will give it an addtional input to indicate we want a model for predicting binary probabilities. Here is the corresponding code segment, followed by the output it produces

```
emaildat = read.csv(file="https://cs.earlham.edu/~pardhan/sage_
   and_r/mock_email_data.csv", header=TRUE, sep=",")
gmod = glm (spam ~ win+free+hurry, data=emaildat, family=
   binomial)
summary(gmod)
```



```
Call:
glm(formula = spam ~ win + free + hurry, family = binomial, data = emaildat)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.7738  -0.7829   0.2077   0.7216   2.2130

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  -3.6378     0.6108  -5.956 2.59e-09 ***
win           0.4265     0.1139   3.745 0.000181 ***
free          0.4269     0.1156   3.693 0.000222 ***
hurry         0.9059     0.1368   6.624 3.50e-11 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 272.74  on 199  degrees of freedom
Residual deviance: 187.14  on 196  degrees of freedom
AIC: 195.14

Number of Fisher Scoring iterations: 5
```

Although this output includes a variety of details, for our present purpose we focus on the coefficients table, which indicates the fitted model is

$$\ln\left[\frac{\hat{p}}{1-\hat{p}}\right] = \hat{z} = -3.6378 + 0.4265(\texttt{win}) + 0.4269(\texttt{free}) + 0.9059(\texttt{hurry}) \tag{1.32}$$

Thus, for example, if an email contains the keyword counts $\texttt{win} = 2, \texttt{free} = 1, \texttt{hurry} = 4$, then the predicted probability of spam can be computed as

$$\begin{aligned}\hat{z} &= -3.6378 + 0.4265(2) + 0.4269(1) + 0.9059(4) = 1.2657\\ \hat{p} &= \frac{1}{1+e^{-1.2657}} = 0.78\end{aligned}$$

The probability that this message is spam is 0.78. For the purpose of binary classification, the user must choose a threshold probability that determines what class the response belongs to. For example, a standard choice is 0.5. With that choice, if $\hat{p} = 0.78$, the email would be classified as spam.

Many conceptual ideas from linear regression can be readily extended to this new setting. For instance, we can associate practically useful meaning to the value of the slope coefficients. In the spam filter example, the coefficient of `win` in equation (1.32) is 0.4265. Thus, for each additional occurrence of the keyword “win” in an email, the model predicts the **log odds** of the message being spam increase by 0.4265 on average, when all other variables are held constant. **But**. . . what does that really mean from a practical standpoint? Let’s take a closer look:

$$
\begin{aligned}
\texttt{win} \text{ increases by } 1 \quad &\Rightarrow \quad \hat{z} \text{ increases by } 0.4265 \\
&\Rightarrow \quad \hat{z}_{new} - \hat{z}_{old} = 0.4265 \\
&\Rightarrow \quad \ln\left[\frac{\hat{p}_{new}}{1-\hat{p}_{new}}\right] - \ln\left[\frac{\hat{p}_{old}}{1-\hat{p}_{old}}\right] = 0.4265 \\
&\Rightarrow \quad \ln[\text{odds}_{\,new}] - \ln[\text{odds}_{\,old}] = 0.4265 \qquad (\text{where odds} = \frac{\hat{p}}{1-\hat{p}}) \\
&\Rightarrow \quad \ln\left[\frac{\text{odds}_{\,new}}{\text{odds}_{\,old}}\right] = 0.4265 \\
&\Rightarrow \quad \frac{\text{odds}_{\,new}}{\text{odds}_{\,old}} = e^{0.4265}
\end{aligned}
$$

The fraction in this last line is known as **the odds ratio**. Thus, we can say that for each additional occurrence of the word "win," the odds ratio of the message being spam is predicted to be $e^{0.4265} \approx 1.53$, on average, when all other variables are held constant. That means whenever `win` increases by 1, the odds of the corresponding email being spam are multiplied by 1.53, all else being held constant. Of course, it is not surprising that the response changes nonlinearly as a function of the predictor, given the nonlinear model we are using here. The other slope coefficients can also be interpreted in a similar way.

For the interested reader, we would like to also briefly describe what the rest of the software output in this example indicates. We have defined deviance residuals in the optional reading supplement above. A 5-number summary of these residuals is provided near the top of the output. The coefficients table includes the usual details about standard errors and P-values that are needed for carrying out inference tests. The null deviance indicates that a model containing no predictors would have a deviance of 272.74, while the residual deviance says it would be 187.14 if all three predictors are included. Since the residual deviance is smaller, it shows that inclusion of the three predictors does produce a model with a closer fit to the data. The degrees of freedom for the null deviance is 199, because it corresponds to only one unknown parameter (the intercept), with a sample size of 200. Likewise, $df = 196$ for the residual deviance since it includes 4 parameters (b_0, b_1, b_2, b_3). The last line indicates the number of iterations that were needed for the optimization algorithm to converge. We usually don't worry about this number, unless it gets very large, suggesting possible convergence problems.

Assumptions and conditions

As with all regression models, there are certain important theoretical requirements that must be met for logistic regression to provide a reliable and valid predictve model. Interestingly, some of the essential conditions for linear regression **are not** required for logistic regression. These include

- Linear relationship between response and each predictor.

- Normally distributed residuals.
- Constant variance of residuals.

Instead, logistic regression requires satisfying the following conditions

1. The response variable must be binary.
2. The logit (or log odds) of the predicted probabilities must be linearly related to each predictor, with no significant outliers.
3. The data must consist of independent observations.

Let us try to check these conditions for the spam filter example discussed earlier. The response variable is `spam`, which is certainly binary since it can only have a value of `TRUE` or `FALSE`. Figure 1.8 shows plots of the predicted logit versus each predictor. These are conditional density plots that show the binary category of the predicted response as a function of the predictor values. While these plots exhibit a generally downward sloping trend, it is not clear how close they are to a linear pattern. As for the third condition, the data were generated randomly for the purpose of demonstration in this example. Thus, it is reasonable to conclude the observations are independent. To summarize, we can say this dataset clearly satisfies two of the three listed conditions. Linear relationship between the logit and each predictor is less well satisfied.

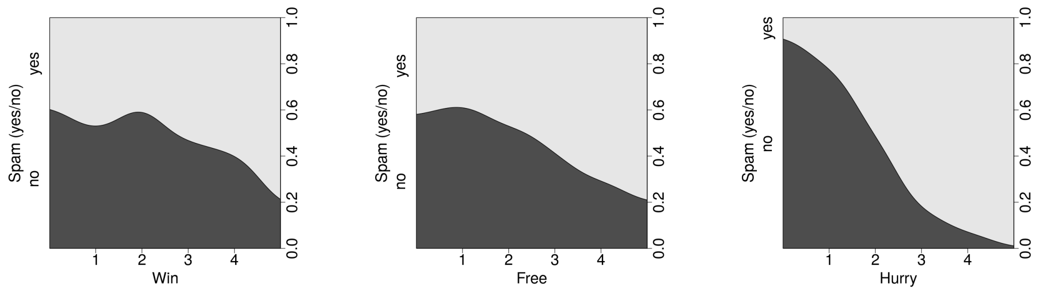


Figure 1.8: Conditional density plots showing predicted response versus each predictor for the spam filter example.

Another diagnostic measure commonly used for assessing the model's prediction accuracy is to see how well it predicts the responses in the input dataset. To do this, the predicted probabilities are converted to a binary response, and these results are compared with the actual response in the dataset. This is analogous to computing residuals in situations where the response is numerical. The only difference is that a binary response is either predicted "correctly," or "incorrectly." Thus, we can compute the proportion of correct predictions for the input dataset and get a quantitative measure of accuracy. For the spam filter example, if we choose a threshold probability of 0.5, the model correctly classifies 78.5% of the cases in the input dataset. In a practical application setting, the best practice would be to use a test dataset that is separate and independent of the input (or, training) dataset, to assess the model's prediction accuracy. This is analogous to the strategies used in supervised machine learning, were a model is developed using one set of data, and tested using a similar but independent dataset.

Inference methods

For convenience, we repeat the general form of the logistic regression model given in equation (1.25)

$$\ln\left[\frac{\hat{p}}{1-\hat{p}}\right] = \hat{z} = b_0 + b_1x_1 + b_2x_2 + \ldots + b_kx_k \tag{1.33}$$

where $x_1, \ldots, x_k$ denote k predictors, $\hat{p}$ is the predicted response (a probability), and $b_0, b_1, \ldots, b_k$ are regression coefficients to be determined by fitting the model to the input data. Some inference strategies from multilinear regression can be applied to the logit $\hat{z}$, since it is linearly related to the predictors.

Typically, the first inference question of interest is whether the model, taken as a whole, is a significant predictor of the response. For this we use a significance test that hypothesizes the true slope coefficient of all the predictors is 0. In other words

$$H_0 : \beta_1 = 0 = \beta_2 = \ldots = \beta_k \tag{1.34}$$

Notice that we continue to use the β-symbol to denote slope parameters when they correspond to the underlying population as a whole. The fitted model provides sample-based estimates for their values, respectively denoted by $b_1, \ldots, b_k$. To show the overall model is a significant predictor, we must demonstrate strong evidence that $\beta_i \neq 0$ for at least one i in $\{1, 2, \ldots, k\}$. This is done by computing a test-statistic based on the estimated model coefficients, and then using it to compute a P-value.

One standard way of computing this type of test statistic is by comparing the ratio of the likelihood of the full model to that of the reduced model. This is known as the **Likelihood Ratio Test**. Typically, the corresponding test statistic is taken to be

$$\Lambda = -2\ln\left[\frac{L(\text{reduced})}{L(\text{full})}\right] \tag{1.35}$$

where L denotes the likelihood function defined in (1.26). In the numerator, it is evaluated for the reduced model (with all the β's set to 0), and in the denominator it is evaluated using the full set of estimated model coefficients. The resulting Λ follows a χ^2 distribution with k degrees of freedom. Thus, it can be used to compute a P-value, and to infer an appropriate conclusion for our significance test.

Observe that equation (1.35) can also be written as

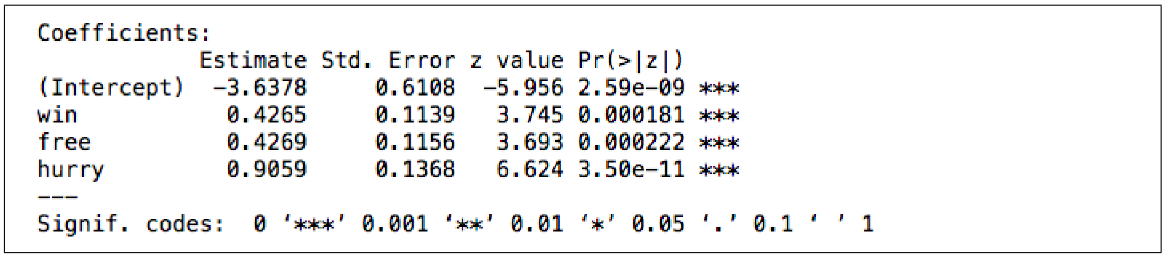
$$\begin{aligned}\Lambda &= -2\ln[L(\text{reduced})] + 2\ln[L(\text{full})] \\ &= \text{Deviance(reduced)} - \text{Deviance(full)}\end{aligned}$$

It follows that our test statistic is simply the difference between the deviances of the two models. Intuitively, this makes sense because we expect a model with a good fit to have much smaller deviance than an "intercept-only" model. Thus, Λ increases as the fit of the model gets better, resulting in lower P-values, and higher probability of inferring significance of the model as a whole.

Another common goal of inference procedures is to test the significance of individual predictors. As usual, for the i^{th} predictor we would hypothesize $\beta_i = 0$, and then compute a test statistic based on the corresponding b_i estimated by our model. A common way of doing this is by computing the **Wald** statistic, which is calculated in a very similar way to a z-score

$$w = \frac{b_i - 0}{SE(b_i)} \tag{1.36}$$

The $SE(b_i)$ in the denominator is obtained from software, and the standard normal distribution is used for computing the P-value. This is known as the Wald test of significance for the parameter β_i. Regression summary output from R typically displays the Wald statistic and P-value for each model parameter. Here is an excerpt from the email spam filter example that shows this (`z value` refers to the Wald statistic)



```
Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  -3.6378     0.6108  -5.956 2.59e-09 ***
win           0.4265     0.1139   3.745 0.000181 ***
free          0.4269     0.1156   3.693 0.000222 ***
hurry         0.9059     0.1368   6.624 3.50e-11 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

In this example we observe that at a 5% significance level ($\alpha = 0.05$), all three variables are significant, according to the Wald test.

A closely related inference procedure is the Wald confidence interval. It provides an interval estimate for the value of the parameter β_i. The computational procedure is very similar to that of confidence intervals based on the standard normal distribution

$$\text{Wald} \;\; CI = [b_i \;\pm\; z^* \times SE(b_i)] \tag{1.37}$$

Here b_i is the fitted estimate for β_i, z^* is the critical z-value for the chosen confidence level, and $SE(b_i)$ is the standard error of the estimate (obtained from software). For example, a 95% Wald confidence interval for the variable `free` in the software output shown above is

$$\text{Wald} \;\; CI_{\texttt{free}} = [0.4269 \;\pm\; 1.96 \times 0.1156] = [0.2003, 0.6535] \tag{1.38}$$

Thus, assuming the conditions are met, we could infer with 95% confidence that the true value of the slope coefficient of `free` lies between 0.2003 and 0.6535. Recall, this slope coefficient models a linear relationship for the log odds of the predicted response. Therefore, the confidence interval says that the $\underline{\text{log odds}}$ of an email being spam increases by a value that lies between 0.2003 and 0.6535, for each 1 unit increase in the variable `free`. In practice, we often exponentiate these values to determine the effect on the odds ratio of the predicted response

$$e^{CI} = [e^{[0.2003}, \ e^{0.6535}] = [1.2218, 1.9222]$$

~~The interval suggests the odds ratio of an email being spam increases by a factor that lies between 1.2218 and 1.9222, for each 1 unit increase in the variable `free`.~~

(I think this last statement is wrong. Fariba? Huong? What are your thoughts? Maybe it is better to replace it with:
"The interval suggests the odds ratio of an email being spam lies between 1.2218 and 1.9222, for each 1 unit increase in the variable `free`."

What I am trying to suggest is that the odds ratio does not increase – but the odds do increase.)

(Yet to complete:
*** More examples showing how to do all the stuff described here.**
*** Homework exercises.)**