STQD6134: Business Analytics

Linear Regression: Supply and Demand

The sub sections

- Understanding linear regression
- Checking model assumptions
- Using a simple linear regression
- Refining data for simple linear regression
- Introducing multiple linear regression

Understanding linear regression

In a business context, analysts use linear regression in diverse endeavors such as evaluating trends, making revenue estimates, analyzing the impact of price or supply changes, and assessing risk.

The lm() function

The principal tool used for linear regression in R is the `lm()` function. Refer to its help page by typing `?lm` into your console. You will see the function has 13 arguments. For the most part, you will be using just two arguments, `formula` and `data`.

Start by loading the data into the `adverts` variable. By using the `str()` function that you learned in `Chapter 2`, *Data Cleaning*, you will see that this dataset consists of 172 observations:

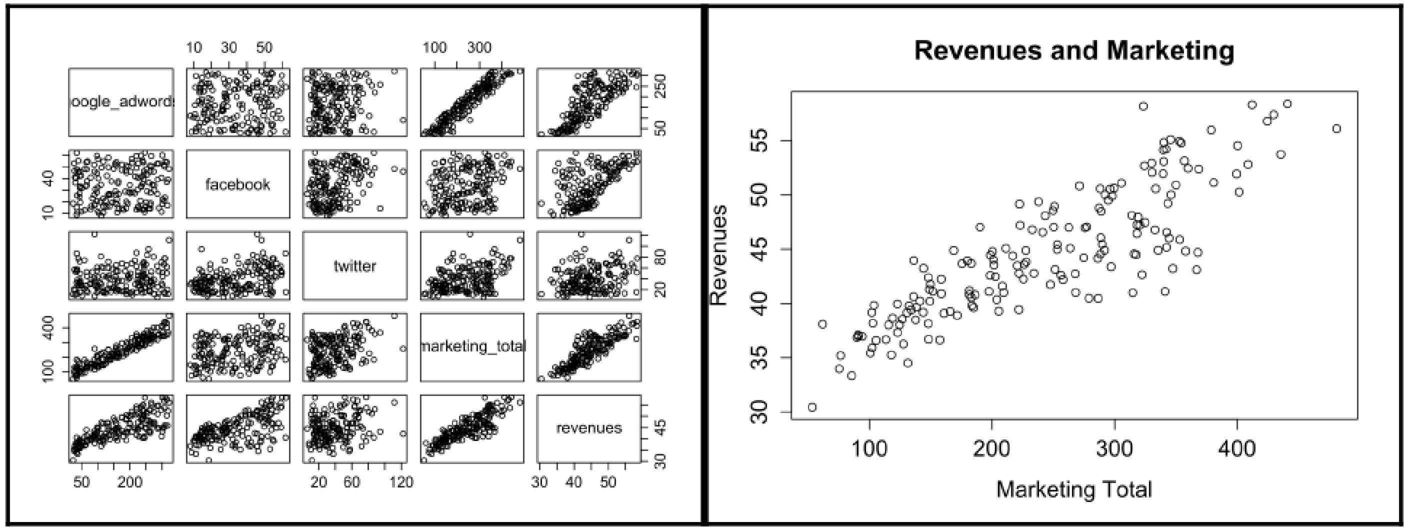
```
adverts <- read.csv("./data/Ch4_marketing.csv"); str(adverts)
```

Each observation has five variables that provide expenditures on three different advertising methods, total marketing cost, and revenue.

By using the `pairs()` function that you learned in `Chapter 3`, *Exploratory Data Analysis*, you will notice a very strong correlation between the `marketing_total` and `revenues` variables. Create a scatterplot to look at this relationship more closely:

```
pairs(adverts)
plot(adverts$marketing_total, adverts$revenues, ylab = "Revenues",
     xlab = "Marketing Total", main = "Revenues and Marketing")
```

We will get a pairs plot and scatterplot as follows:



It is easy to imagine a straight line extending from the lower-left of the scatterplot to the upper-right, running straight through the data. This is a primary indication that the relationship might lend itself to linear regression.

Simple linear regression

You run an **simple linear regression** (**SLR**) by formulating a model with your data. The formulation generally consists of a response variable (`Y`) and a predictor variable (`X`), linked by a tilda (~). You can think of this formulation as either *Y as a function of X* or *Y regressed on X*. The data needs to be in a data frame. A generic example in R would look similar to the following:

```
model <- lm(Y ~ X, data = dataset)
```

It is now time for you to build an SLR. The first line of the code runs the regression and saves the `lm` object to a variable called `model1`. The second line displays the contents of the `model1` object to the console:

```
model1 <- lm(revenues ~ marketing_total, data = adverts)
model1
```

The output is as follows:

```
Call:
lm(formula = revenues ~ marketing_total, data = adverts)

Coefficients:
    (Intercept)  marketing_total
       32.00670          0.05193
```

This output informs you about the following two things:

- `Call` summarizes the model. It reminds you that you are using the `lm()` model, the formulation used (`revenues ~ marketing_total`), and the `adverts` dataset.
- `Coefficients` provides you with two important numbers: an `Intercept` of `32.00670`, and a slope (`marketing_total`) with a value of `0.05193`.

Interpreting your models: Models are full of numbers. Without the ability to interpret them, they provide little use to business leaders. In the case of your first SLR, you will articulate the output using the following statements:

- Revenue increases by $51.93 for every $1,000 increase in total marketing
- Revenue is $32,007 when total marketing expenditure is $0

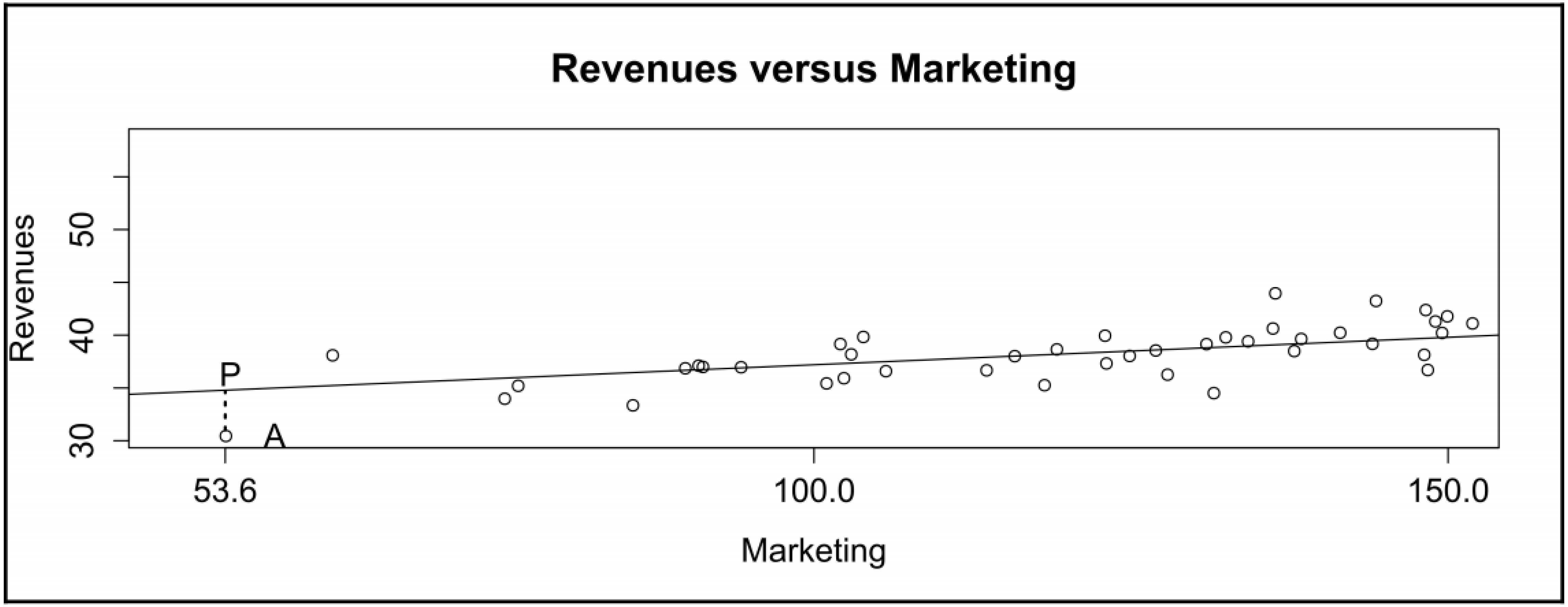
The `Intercept` value is important for model execution, but may not have as much interpretability. In your case, this number sets the baseline level for the model.

Residual

The intercept and slope provide the parameters for a prediction equation:

*Revenue = 32.0067 + (0.05193 * marketing_total)*

Residual


Revenues versus Marketing
Revenues
Marketing
30
40
50
53.6
100.0
150.0
P
A

Checking model assumptions

- Linearity
- Independence
- Normality
- Equal variance

Linearity

Linearity assumption: The relationship between the predictor and response variables is linear.

In an SLR situation, a quick way to determine linearity is to plot the variables with a scatterplot. Earlier, you saw a strong correlation and linearity between the `marketing_total` and `revenues` variables. A straight line would run through the data. This appears to meet the linearity assumption. Things get more complex when using more than one predictor variable, but you can see this assumption more easily when plotting only one predictor variable.

Independence

Independence assumption: The relationship between variables is independent of one another.

This is probably the most difficult assumption to test. You can typically handle it using common sense along with an understanding of the data and its collection method. A good example of independence is observations in the marketing data. They all come from a different location. It is reasonable to assume that the marketing budgets from one location have no impact on another.

Normality

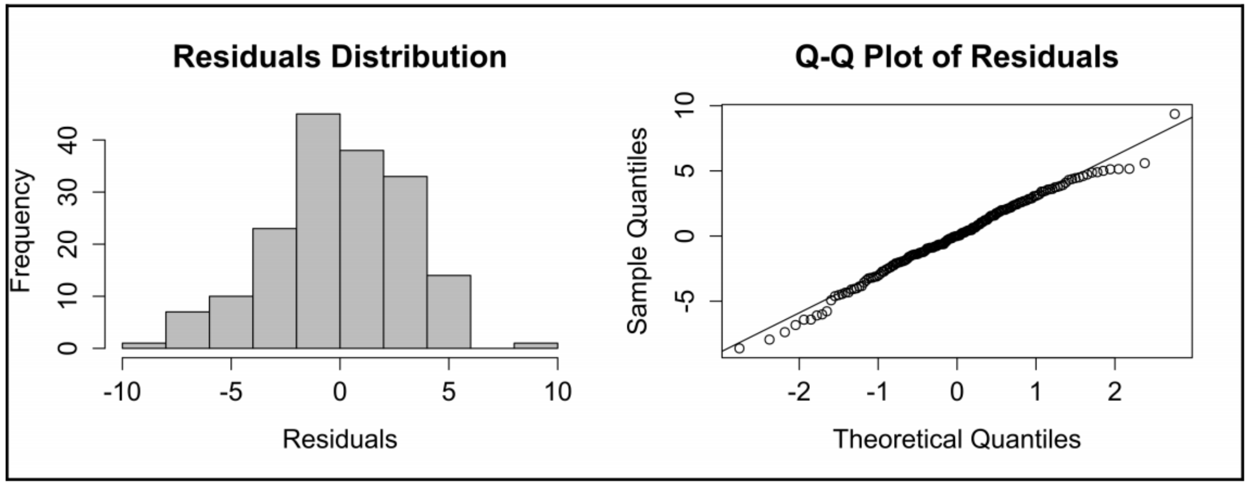
Normality assumption: The residuals form a normal distribution around the regression line with a mean value of zero.

A histogram provides an excellent way to view the distribution of the residuals. Another plot to check the normality is the normal quantile, or Q-Q, plot created using the `qqnorm()` function. It tests for normality by plotting the data by quantiles, comparing how they actually appear versus how they would appear in a theoretical normal plot.

The following code creates a two-panel plot with a histogram in the left panel and the Q-Q plot in the right. The `$` operator is used to extract the residuals from the `model1` object:

```
par(mfrow = c(1, 2))
hist(model1$residuals, xlab = "Residuals", col = "gray",
     main = "Residuals Distribution")
qqnorm(model1$residuals, main = "Q-Q Plot of Residuals")
qqline(model1$residuals)
```

The histogram shows residuals that look normally distributed. Meanwhile, the closer the residuals hug the diagonal line in the Q-Q plot, the more normal the distribution, as shown in the following diagrams:



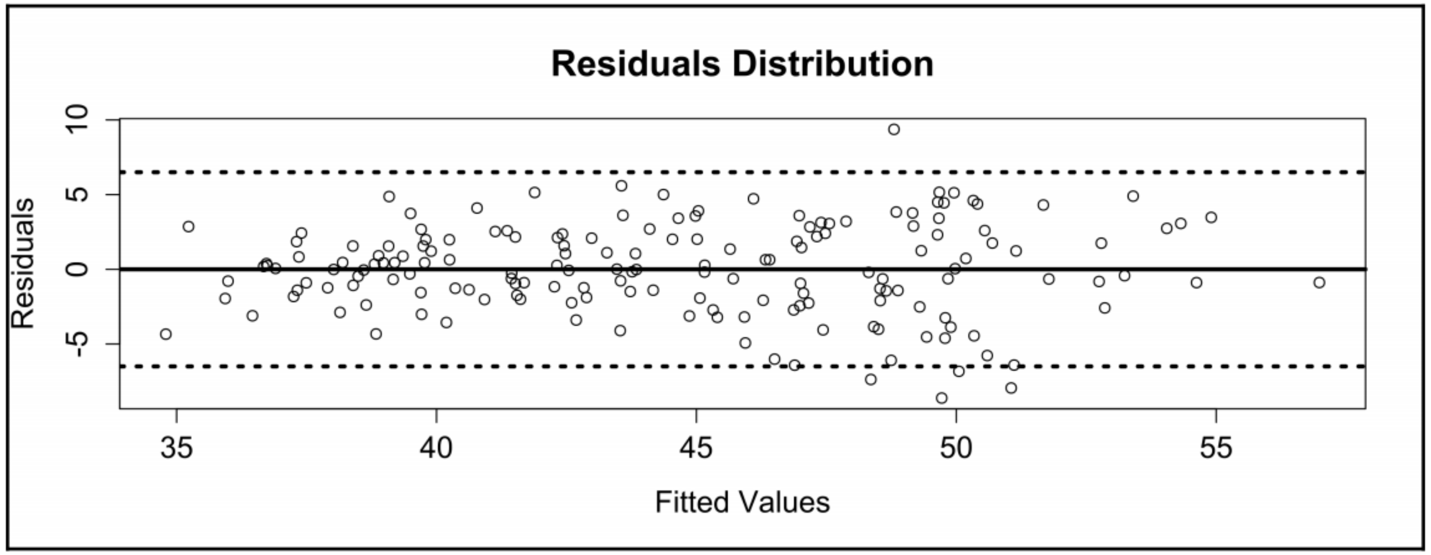
Equal variance

Equal variance assumption: Residuals form a random pattern distributed around a mean of zero.

You can see this by plotting the residuals against the fitted (predicted) values of the model:

```
plot(model1$fitted.values, model1$residuals, ylab = "Residuals",
     xlab = "Fitted Values", main="Residuals Distribution")
abline(0, 0, lwd = 3); abline(h = c(-6.5, 6.5), lwd = 3, lty = 3)
```

The code once again uses the `$` operator to extract the residuals. It is also extracts the fitted values from `model1`. The `abline()` function was used to create reference lines of a mean around zero and for a visual indictor of equal variance:



Assumption wrap-up

"A rule of thumb is that if things look OK, then they probably are OK."

– Anonymous

You built a simple model. Modeling is easy, but it requires skill to know that it is appropriate for the data. Notice the amount of time you spent making sure the model met the assumptions of linear regression. Some assumptions are more important than others, as follows:

- All linear regression operations and results are very sensitive to even small departures from independence.
- All operations are sensitive to moderate departures from equal variance.
- Some operations deal well with departures from normality. One exception is that prediction intervals are quite sensitive to departures from normality.

Using a simple linear regression

Interpreting model output

```
summary(model1)
```

The following is the output:

```
Call:
lm(formula = revenues ~ marketing_total, data = adverts)

 Residuals:
    Min      1Q  Median      3Q     Max
-8.6197 -1.8963 -0.0006  2.1705  9.3689


Coefficients:
                 Estimate Std. Error t value Pr(>|t|)
(Intercept)     32.006696   0.635590   50.36   <2e-16 ***
marketing_total  0.051929   0.002437   21.31   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 3.054 on 170 degrees of freedom
Multiple R-squared:  0.7277,  Adjusted R-squared:  0.7261
F-statistic: 454.2 on 1 and 170 DF,  p-value: < 2.2e-16
```

- `Residuals` provide the Tukey five-number summary of the residuals (you learned this in `Chapter 3`, *Exploratory Data Analysis*). This particular output is consistent with what you saw in your residual analysis, namely that the residuals distribution is relatively normal around a mean of zero.
- `Coefficients` are the same as you saw before, but now you also see the standard errors, t-values, and p-values. Recall the concept of the null hypothesis from `Chapter 3`, *Exploratory Data Analysis*. The p-value is a measure of how surprised you would be to see the result in the null hypothesis that the intercept and slope equal zero. The p-value in each case is substantially less than the statistical benchmark of 0.05, meaning that if either of them were zero, you would be very surprised at this result.
- `Adjusted R-squared` tells you how much error your model explains. In this case, your model accounts for nearly 73% of the error. The goodness of this sort of percentage varies from domain to domain.
- `F-statistic` (and its associated `p-value`) indicate whether your overall model works. In the case of SLR, with a single predictor, the `p-value` associated with the slope is sufficient for this. `F-statistic` is more useful with **multiple linear regression** (**MLR**).

Predicting unknown outputs with an SLR

```
newdata <- data.frame(marketing_total = 460)
predict.lm(model1, newdata, interval = "predict")
```

It will return the output as follows:

```
       fit      lwr      upr
1 55.89403 49.75781 62.03025
```

```
predict.lm(model1, newdata, level = 0.99, interval = "predict")
```

The output is as follows:

```
       fit      lwr      upr
1 55.89403 47.79622 63.99184
```

You can also use `predict.lm()` to examine multiple input values of `marketing_total`:

```
newdata = data.frame(marketing_total = c(450, 460, 470))
predict.lm(model1, newdata, interval = "predict")
```

The following is the output:

```
       fit      lwr      upr
1 55.37474 49.24653 61.50295
2 55.89403 49.75781 62.03025
3 56.41332 50.26873 62.55791
```

Working with big data using confidence intervals

You will find the term *big data* used in different ways. One definition is *a dataset that is too big for processing given current techniques or hardware* (*Sicular, 2013*). Computer scientists have dealt with big data by developing algorithms, such as MapReduce, and frameworks, such as Hadoop. These distribute the data to multiple processors where they bin and count data by category (Map) and then assemble the various outputs into an aggregated result (Reduce).

Statisticians, on the other hand, developed sampling techniques as one way of dealing with similar problems. Imagine that you have a dataset with billions of observations, and that processing such a dataset on your machine would take too long. Under certain conditions, you might take smaller samples of this data, process these samples, and then try to infer things about the larger dataset.

When you use a sample from a larger population, you cannot be certain that your model output is correct. It is just an estimate of some unknown true parameter associated with the larger population. Like predictions, confidence intervals can help mitigate this problem. For example, take a random sample of 30% of your marketing data:

```
set.seed(4510)
market_sample <- sample_frac(adverts, 0.30, replace = FALSE)
```

Set a randomization seed to allow you to get repeatable results. Then use the `sample_frac()` function from the `dplyr` library. You can pass a fraction as a parameter. Setting `replace = FALSE` pulls samples without putting them back. Then run `lm()`:

```
samp_model <- lm(revenues ~ marketing_total, data = market_sample)
samp_model
```

The output is as follows:

```
Call:
lm(formula = revenues ~ marketing_total, data = market_sample)
Coefficients:
    (Intercept)  marketing_total
       31.08243          0.05531
```

The coefficients for the `Intercept` and `marketing_total` are close to the values you saw in `model1`, but not quite the same. If you were to take a different sample, then you would get slightly different coefficients again. You can use the `confint()` function to gain insights about population estimates while accounting for sampling uncertainty:

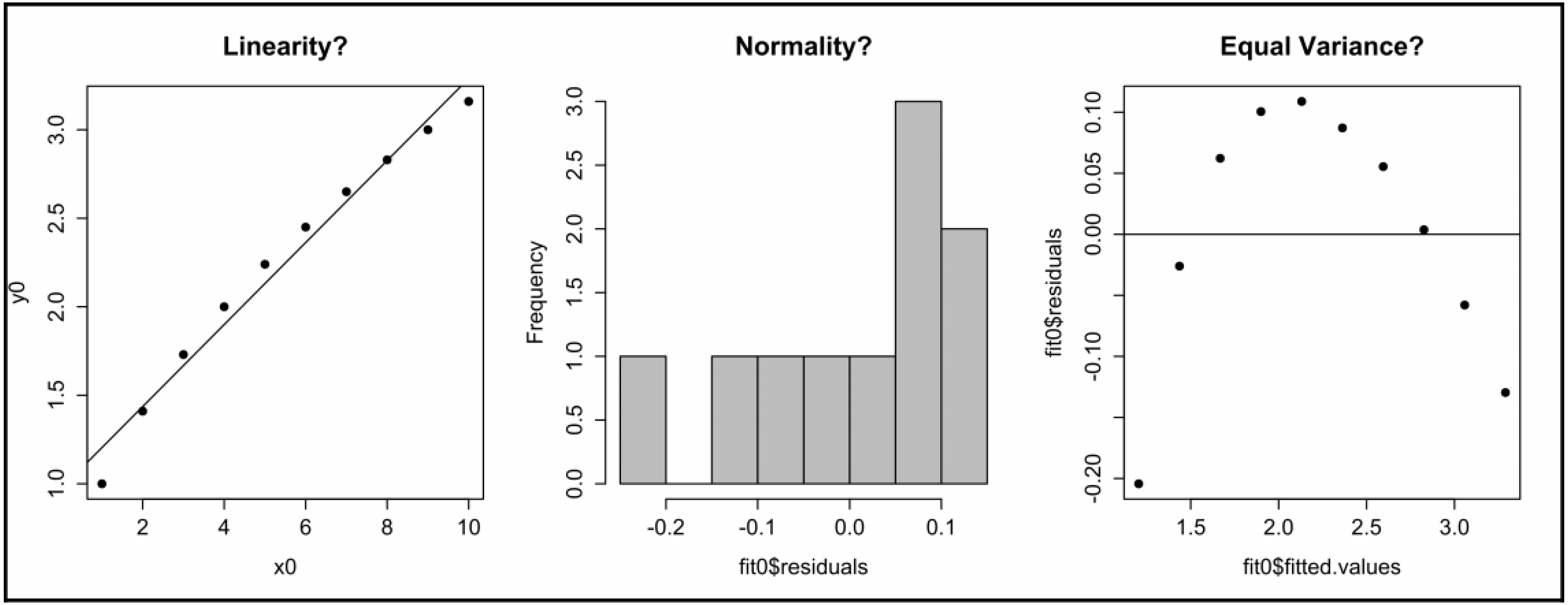
```
confint(samp_model)
```

The output is shown here:

```
                      2.5 %      97.5 %
(Intercept)     28.49785446 33.66700509
marketing_total  0.04615628  0.06445785
```

What this shows is that if you take 100 different random samples from the population, then 95 of the 100 samples would estimate the slope of `marketing_total` between the values `0.04615628` and `0.06445785`. Indeed, the slope of `marketing_total` in the marketing dataset is `0.05193`, which falls in this interval. You now have a tool that can help you analyze a big data dataset and have some confidence in the estimates you calculate.

Refining data for a simple linear regression



```
x0 <- c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
y0 <- c(1.00, 1.41, 1.73, 2.00, 2.24,
        2.45, 2.65, 2.83, 3.00, 3.16)
fit0 <- lm(y0 ~ x0)

par(mfrow = c(1, 3))
plot(x0, y0, pch = 19, main = "Linearity?"); abline(fit0)
hist(fit0$residuals, main = "Normality?", col = "gray")
plot(fit0$fitted.values, fit0$residuals,
     main = "Equal Variance?", pch = 19); abline(h = 0)
```

Transforming data

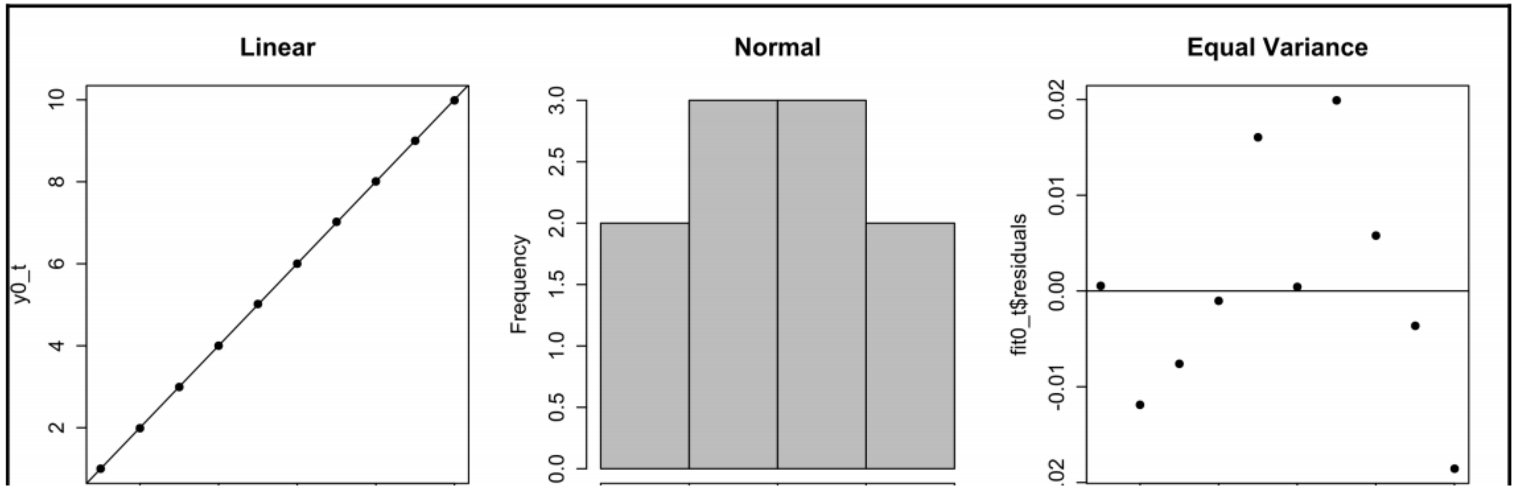
A transformation is an operation to fix data so that it meets model needs. In order to run an SLR, data should satisfy the LINE assumptions. There are some general rules of thumb available to you depending on the assumption(s) that the data fails to satisfy:

- **Not independent**: You may need to transform your independent variables into a time series and apply a time series analysis. This approach is in `Chapter 6`, *Time Series Analysis*.
- **Not linear**: You might consider transforming the independent (input) variable.
- **Not normal or unequal variance**: You might consider transforming the response variable (the dependent variable).

You can fix this simulated dataset by squaring the `y0` variable using the `^2` operator. The transformed variable is stored as `y0_t`:

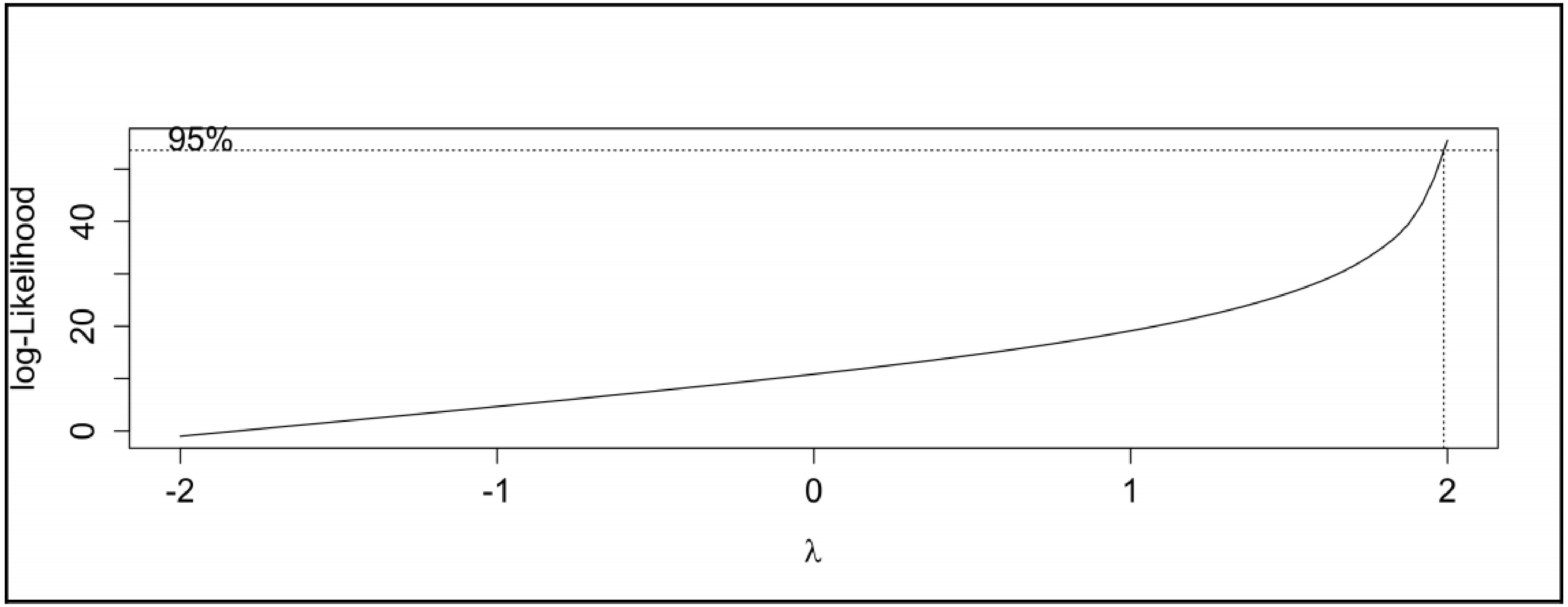
```
y0_t <- y0^2
fit0_t <- lm(y0_t ~ x0)

plot(x0, y0_t, pch = 19, main = "Linear"); abline(fit0_t)
hist(fit0_t$residuals, main = "Normal", col = "gray")
plot(fit0_t$fitted.values, fit0_t$residuals,
     main = "Equal Variance", pch = 19); abline(h = 0)
```

The output is shown as follows:



```
library(MASS)
boxcox(fit0)
```

The following is the output:

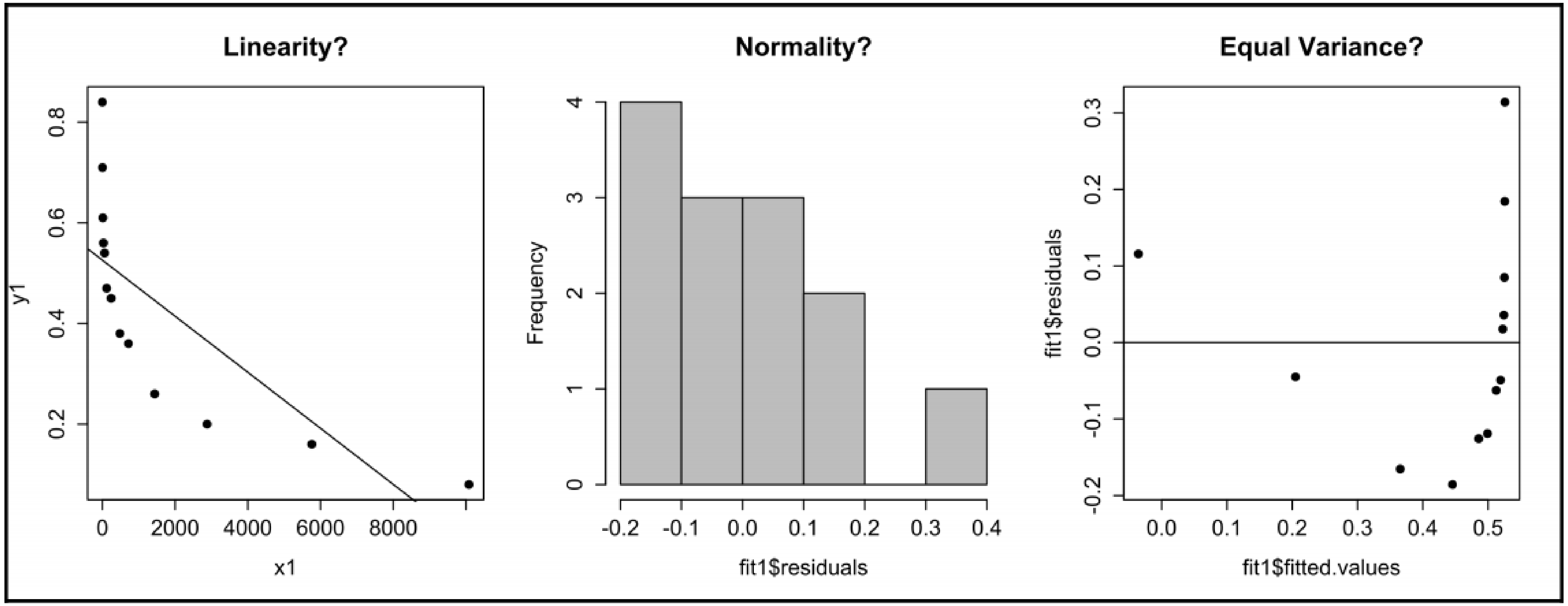


The graphical output gives you a clue about the exponent to which you will raise your response variable. The level of lambda (Î») is **2**, as shown by the vertical dotted line. This is just what you did. You raised the `y0` variable to the second power to get the fit you needed.

Now, consider an example of transforming independent (input) variables. Here are some new data and diagnostic plots:

```
x1 <- c(1, 5, 15, 30, 60, 120, 240, 480,
        720, 1440, 2880, 5760, 10080)
y1 <- c(0.84, 0.71, 0.61, 0.56, 0.54, 0.47,
        0.45, 0.38, 0.36, 0.26, 0.2, 0.16, 0.08)
fit1 <- lm(y1 ~ x1)

plot(x1, y1, pch = 19, main = "Linearity?"); abline(fit1)
hist(fit1$residuals, main = "Normality?", col = "gray")
plot(fit1$fitted.values, fit1$residuals,
     main = "Equal Variance?", pch = 19); abline(h = 0)
```

The diagnostic plots generated are as follows:

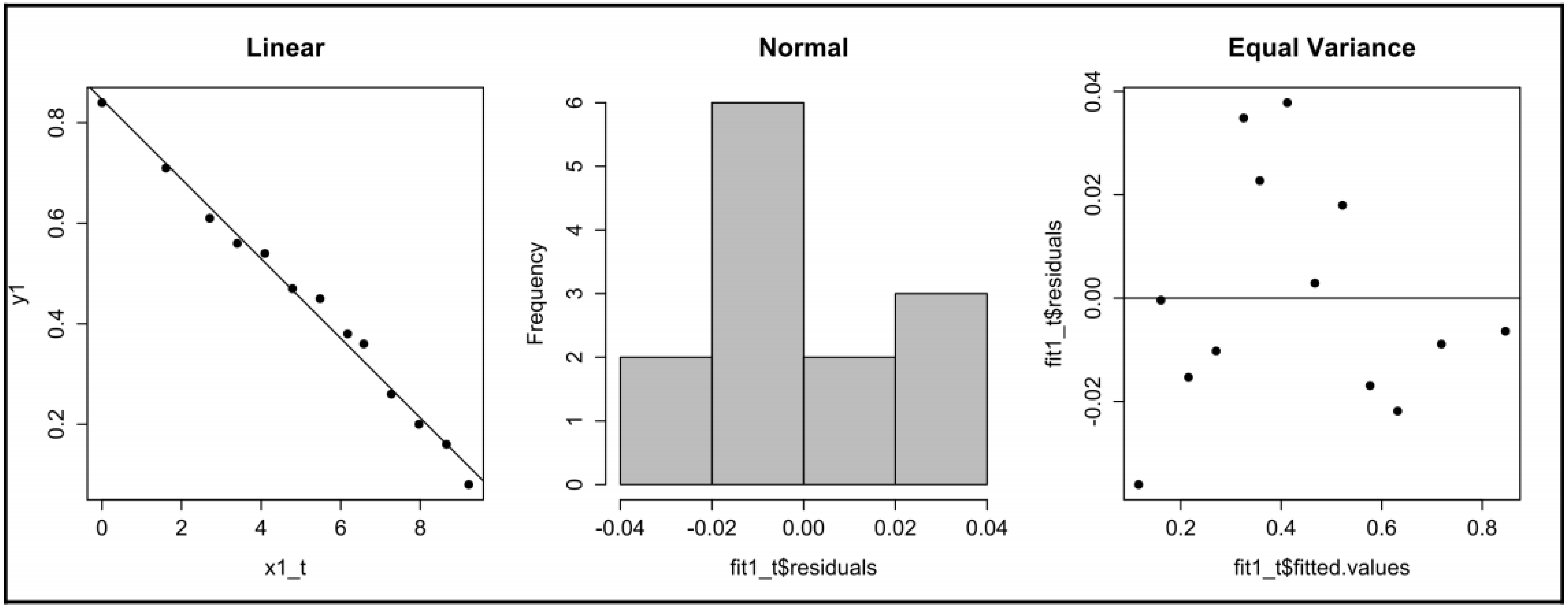


These plots show problems with the assumptions to include severe non-linearity. The rules of thumb indicate that you might consider transforming the independent variable when working with non-linear data. Unfortunately, the `boxcox()` function does not work with independent variables. If you do run this function, you will get a lambda (Î») ranging from -1.2 to -0.7. You can try transforming the response variable by raising `y0` to the power of -1.0, but it does not solve the linearity problem. This is because the problem is with the independent variable. How will you transform the independent variable?

Look at the logarithmic pattern in the scatterplot. A good place to start is often with a `log()` transformation, that is to say, transforming the independent variable by the natural logarithm function and then performing the regression:

```
x1_t <- log(x1)
fit1_t <- lm(y1 ~ x1_t)

plot(x1_t, y1, pch = 19, main = "Linear"); abline(fit1_t)
hist(fit1_t$residuals, main = "Normal", col = "gray")
plot(fit1_t$fitted.values, fit1_t$residuals,
     main = "Equal Variance", pch = 19); abline(h = 0)
```

Transforming the independent variable handles the nonlinearity. Now the data meets the other assumptions as well. Data transformations are a complex topic. There may be times when the rules of thumb are not sophisticated enough to help you or when the independent and response variables need transformations:

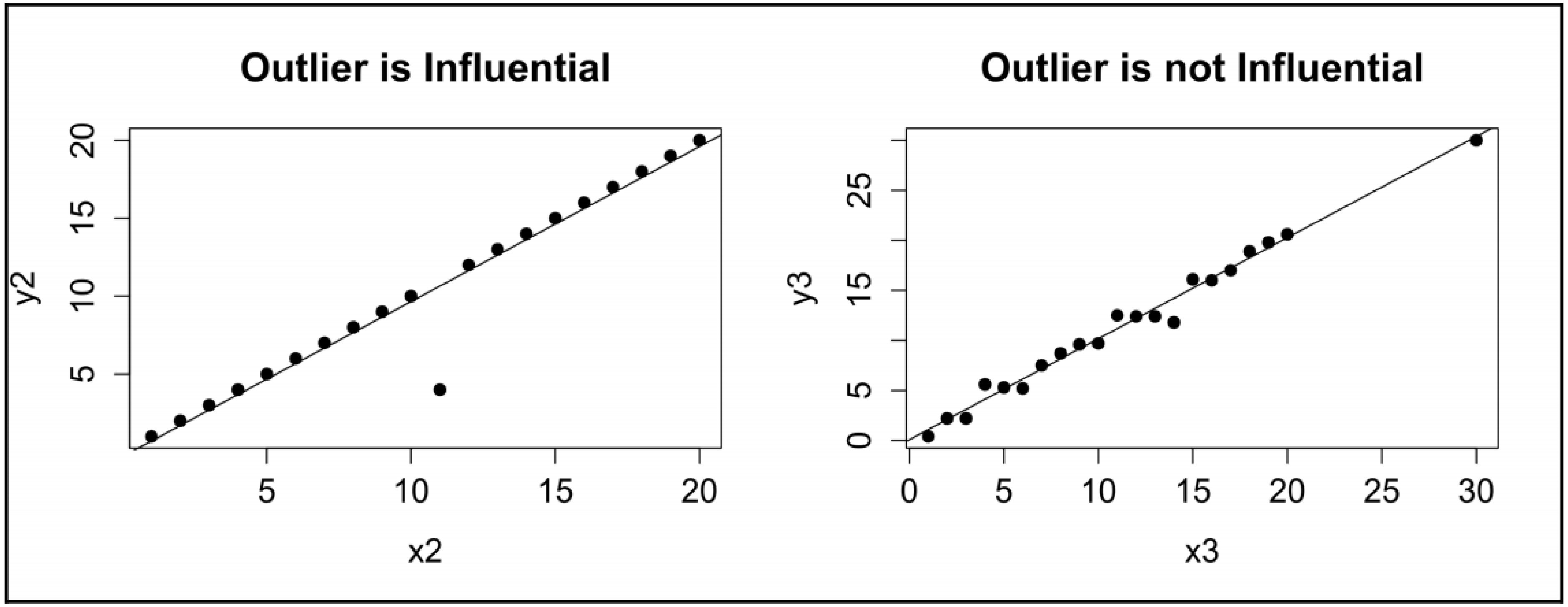


Handling outliers and influential point

An outlier is a data point that does not follow the trend of the response variables. You handle outliers by dealing with the individual data points rather than transforming the dataset mathematically. Consider this data and inspect the plots. Can you see the outliers?

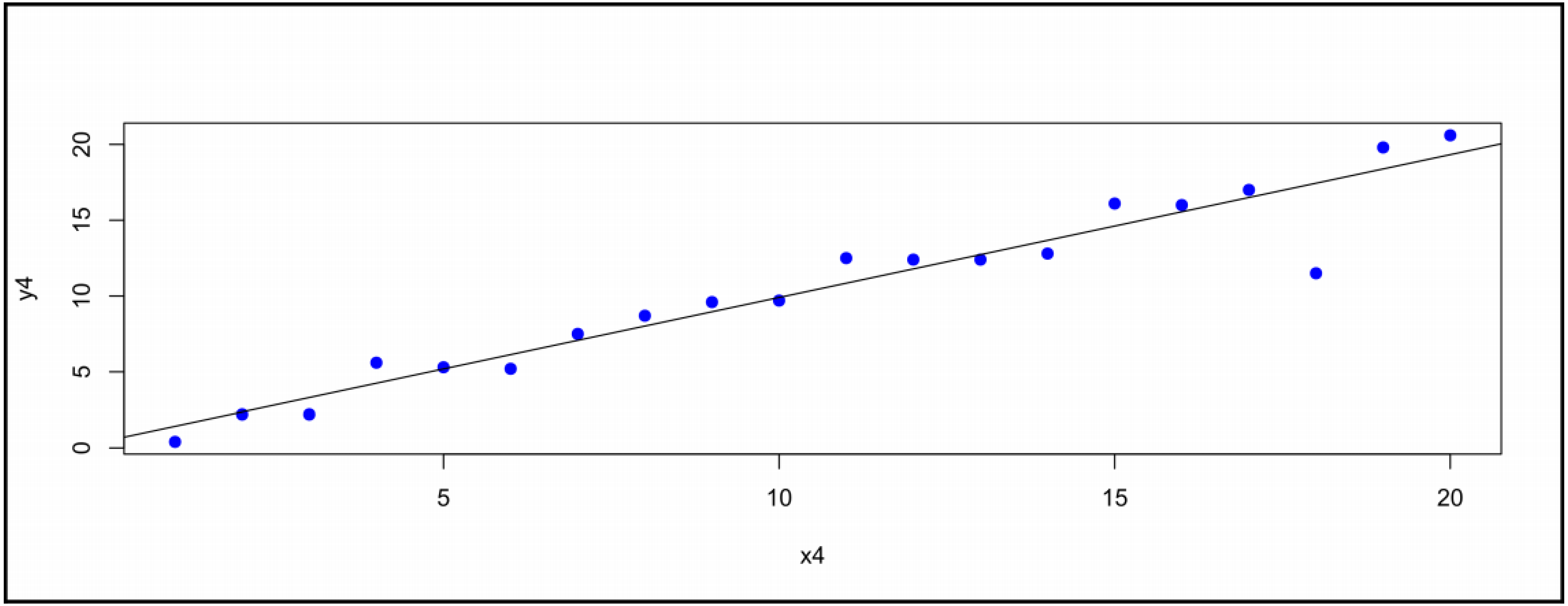
```
x2 <- 1:20
y2 <- c(1:10, 4, 12:20)
x3 <- c(1:20, 30)
y3 <- c(0.4, 2.2, 2.2, 5.6, 5.3, 5.2, 7.5, 8.7, 9.6, 9.7, 12.5,
        12.4, 12.4, 11.8, 16.1, 16, 17, 18.9, 19.8, 20.6, 30.0)
```

The plots are shown here:



If you noticed two outliers (x2, y2) = (11, 4) and (x3, y3) = (30, 30), then you are correct. The next question is, *Why is the first outlier influential while the second is not?* An **influential point** is an outlier that disproportionately alters the result of a regression model. The influence could affect the predicted responses, the estimates of the slope coefficients, or the results of the hypothesis test(s).

```
x4 <- c(1:20)
y4 <- c(0.4, 2.2, 2.2, 5.6, 5.3, 5.2, 7.5, 8.7,
        9.6, 9.7, 12.5, 12.4, 12.4, 12.8, 16.1,
        16.0, 17.0, 11.5, 19.8, 20.6)
```



The data point at (x4, y4) = (18, 11.5) is an outlier, and it is probably an influential point. Try doing regression on the dataset with this point included in the dataset, and then with this point removed. Compare the regression outputs. Is the outlier affecting the results? According to the summary table of the regression output, the outlier is influencing the results:

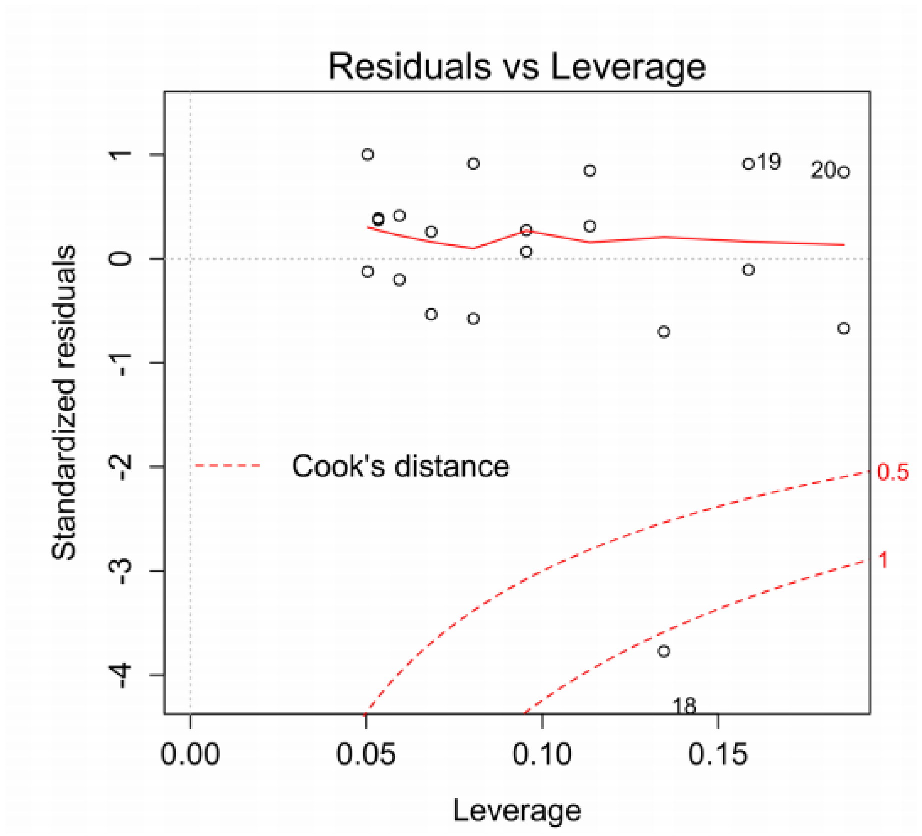
Regression output	**With outlier**	**Without outlier**
Residual (min, median, max)	`(-5.94, 0.43, 1.65)`	`(-1.49, 0.15, 1.51)`
Intercept and slope	`0.4805, 0.9423`	`0.0107, 1.0200`
Adjusted R-squared	`0.9151`	`0.9821`
F-Statistic	`205.7`	`986.1`

So how do you identify outliers and influential points? A very good way to spot outliers is visually. In `Chapter 3`, *Exploratory Data Analysis*, you learned about boxplots, histograms, and scatterplots. These simple tools help reveal patterns and anomalies in your data. Identifying influential points is a bit trickier, but there is an analytic tool called **Cook's Distance** to help you. How do you find Cook's Distance? Use the `plot()` function on the model object.

R tip: You do not need to learn the theory or mathematics of Cook's Distance to use it. The following is a rule of thumb you can apply to outliers:

- **Suspicious**: If a data point has a Cook's Distance > 0.5
- **Likely influential**: If a data point has a Cook's Distance > 1.0
- **Influential**: If a point has a Cook's Distance that stands out from all the others

Run the `plot({model_name})` function on the practice problem data, including the outlier.



- **Errors**: Begin by using exploratory data analysis to see if there are errors in the data. This is common and it could solve the problem.
- **Recheck assumptions**: Maybe there are non-linearity issues after all. If this is the case, you might have to change the type of model used.
- **Incorrect formulation**: There is the possibility that you did not properly formulate your regression model. Consider other formulations.
- **Delete data points**: As a last resort, consider deleting data points. Be advised that this is both controversial and potentially hazardous to your results. You should have a very good reason for deleting data. Do not remove data just because the regression model is not turning out the way you would like. If you do remove the data, make sure you describe it in your data analysis, and provide your justification for deleting it.

Introducing multiple linear regression

In the *Using a simple linear regression* section, you looked at revenue as a function of the marketing budget. You learned a great deal about the relationship by regressing the `revenues` variable on the `marketing_total` variable.

The total amount spent on marketing is the sum of `google_adwords`, `facebook`, and `twitter` marketing expenditures. Using MLR, you can examine the relationship among revenue and some or all of these component budgets.

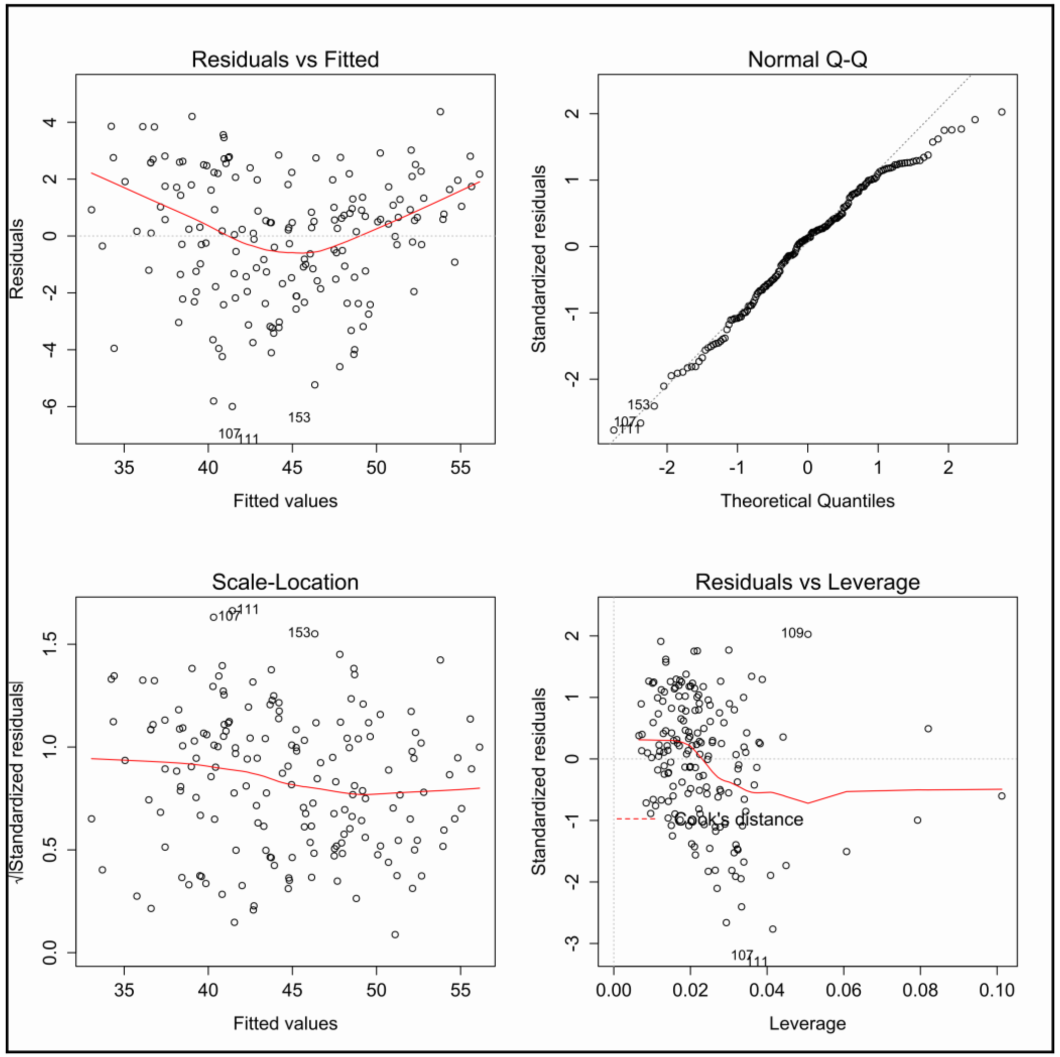
You will formulate an MLR similar to an SLR, but with more predictor terms. You can think of this formulation as *Y regressed on X1 and X2 and so forth*. The most common relationship is an additive relationship, and the + operator is used to include multiple variables. The data needs to be in a data frame. A generic example in R would look similar to the following:

```
model <- lm(Y ~ X1 + X2 + ..., data = dataset)
```

Here is the formulation after you model the revenue regressed on all three components of the marketing budget:

```
model2 <- lm(revenues ~ google_adwords + facebook + twitter,
             data = adverts)
```

- Look at linearity and equal variance from the **Residuals vs Fitted** plot
- Think about independence based on the data source and its collection
- Assess the normality assumption with the **Normal Q-Q** plot
- Identify outliers that are influential with the **Residuals vs Leverage** plot



You will see that the **Residuals vs Fitted** plot shows a slight bow, which might indicate non-linearity. You could experiment with your transformation skills and see if you can come up with something better. Running `summary(model2)` provides this output for interpretation. Use what you learned when interpreting SLR output:

```
Call:
lm(formula = revenues ~ google_adwords + facebook + twitter,
    data = adverts)
Residuals:
    Min      1Q  Median      3Q     Max

 -5.9971 -1.4566  0.2791  1.7428  4.3711
 Coefficients:
                Estimate Std. Error t value Pr(>|t|)
 (Intercept)    29.545988   0.533523   55.38   <2e-16 ***
 google_adwords  0.048384   0.001947   24.85   <2e-16 ***
 facebook        0.197651   0.011871   16.65   <2e-16 ***
 twitter         0.003889   0.008270    0.47    0.639
 ---
 Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
 Residual standard error: 2.214 on 168 degrees of freedom
 Multiple R-squared:  0.8585,  Adjusted R-squared:  0.856
 F-statistic: 339.8 on 3 and 168 DF,  p-value: < 2.2e-16
```