5

Introducing Statistics in R

5.1 Getting started doing statistics in R

We would like to emphasize what we started with in Chapter 4: we have a very fundamental rule for data analysis. *Never* start an analysis with statistical tests. *Always* start an analysis with a picture.

Why? As we noted, you should have in your head an *Expected* pattern in your data and a picture is a good way to see whether your data meet these expectations. Plot it first. Make the picture that should tell you the answer. Now, what do you do after you plot the data? Our philosophy, translated into a workflow, is as follows.

Once you've made a picture, you embark on translating the hypothesis you are testing into a statistical model and this model into R language. If you've made an informative picture that reveals the relationships among variables central to your hypothesis, this should be easy (or become easier with experience).

Once you've specified your model in R and made R build it, we feel that it is vital *not* to then interpret the results. It is, instead, vital to assess first the assumptions of your model. For example, a two-sample *t*-test may assume equal variance in the two groups, and ANOVA and regression assume normally distributed residuals, amongst other things. By assessing

Getting Started with R Second Edition. Andrew Beckerman, Dylan Childs, & Owen Petchey: Oxford University Press (2017).
DOI 10.1093/oso/9780198787839.001.0001

if these assumptions are met, you are ensuring that the results returned by the modelling are reliable. If the assumptions are not met, then the predictions from, or interpretation of, the model is compromised. You will not be making reliable inference.

Only once you have assessed the assumptions should you begin to interpret the output of the statistical modelling. It is here that we interpret the test statistic and associated *p-value*. The final step is to integrate the modelling results into your original figure—a process some of you may know as adding predicted or fitted lines (or points) to your graph.

This chapter is a long one and covers four simple statistical tests in discrete chunks. Take breaks as you work through the sections. In each, we follow this workflow: (1) plot your data, (2) build your model, (3) check your assumptions, (4) interpret your model, and (5) replot the data *and* the model. Along the way, we aim to reinforce several principles from Chapters 1–4, taking advantage of the tools and skills you developed with ***dplyr*** and ***ggplot2***. Our aim with these examples is to show you how easy interpreting statistics can be if you can make an informative picture before you embark on the analysis. Just to whet your appetite, Chapters 6 and 7 expand the types of tests, introducing the two-way ANOVA and ANCOVA in Chapter 6 and then the generalized linear model in Chapter 7.

5.1.1 GETTING READY FOR SOME STATISTICS

If you have never actually used a *t-test*, χ^2 *contingency table analysis*, *linear regression* or *one-way ANOVA*, or never taken a course in statistics that introduced these tools, now is a good time to go and acquire a bit of knowledge. Our instructions will certainly teach you some statistics, but our goal is to teach you how to use (get) R to do these. We are *assuming* that you understand when and for what type of data you may require these tools. We are *assuming* that you understand what types of *hypotheses* can be tested with these different methods.

5.2 χ^2 contingency table analysis

We wish to make two points with this introduction to the χ^2 ('chi-squared') contingency table analysis. First, always plot your data first. Did you expect that? Second, make every attempt to understand the hypothesis that you are testing, both biologically and statistically.

The χ^2 contingency table analysis is an analysis of count data. It is essentially a test of association among two or more categorical variables. The need for this sort of analysis arises when you have a set of observations, or events, and for each of these you can *classify* them by more than one categorical variable with, for example, two levels (for example, sex, male/female, and life stage, juvenile/mature). Introducing some data will help explain the basics.

5.2.1 THE DATA: LADYBIRDS

Let's assume that we have collected data on the frequency of black and red ladybirds (*Adalia bipunctata*) in industrial and rural habitats. Can you already see the two grouping variables, each with two levels?

Why might we have collected such data? It has long been thought that industrial and rural habitats, by virtue of the amount of pollution in them, provide different backgrounds against which many insects sit. Contrasting backgrounds to the insect can be bad if contrast is associated with predation risk. Here, we are interested in whether dark morphs are more likely to reside on dark (industrial) backgrounds. These data are available at `http://www.r4all.org/the-book/datasets`; if you have already downloaded the zipped-up file of all datasets, you already have it. The data file is `ladybirds_morph_colour.csv`.

By performing a χ^2 contingency table analysis, we are testing the null hypothesis that there is no association between ladybird colours and their habitat. The alternative is that there is. The test does not allow us to specify the direction of this association, but that is something we will see from the

graph we make before performing the test. Biologically, we are trying to answer the question of whether some feature of the habitat is associated with the frequencies of the different colour morphs. Note how generic this statement is—there is no specification of the features of the habitat, or of the direction of potential association.

Now we know what questions we're asking, let's read in the data and check their structure. Remember to set up a new script, clear R's brain, define the packages you want to use, set the working directory, and get the data:

```
# My First Chi-square contingency analysis

# Clear the decks
rm(list = ls())

# libraries I always use.
library(dplyr)
library(ggplot2)

# import the data
lady <- read.csv("ladybirds_morph_colour.csv")

# Check it out
glimpse(lady)
```

```
## Observations: 20
## Variables: 4
## $ Habitat      (fctr) Rural, Rural, Rural, Rural, Rural...
## $ Site         (fctr) R1, R2, R3, R4, R5, R1, R2, R3, R...
## $ morph_colour (fctr) black, black, black, black, black...
## $ number       (int) 10, 3, 4, 7, 6, 15, 18, 9, 12, 16,...
```

Looking at the output of **glimpse**(lady), we see that the dataset is quite *tidy*, with 20 rows, each of which is a location at which ladybirds were observed. The Habitat variable shows whether the observation was in an *Industrial* or *Rural* location, the Site variable is a unique identifier for the location, the morph_colour variable is which morph colour was observed (red or black), and the number variable is the number of individual ladybirds of that morph colour that were observed.

(Note that it is a good idea to be consistent with the case of variable names, rather than having some with an upper-case first letter and others with a lower-case first letter. We like to keep you on your toes, though.)

There are various ways we could analyse these data to answer our question. We're going to take a quite simple approach (for learning purposes): a χ^2 contingency table analysis, to test if the frequencies of the two colours of ladybirds differ between the two habitat types.

The first thing you may notice is that we first need to calculate these totals... This is a chance to make use of your ***dplyr***-ninja skills. Before you do this, think about what we are aiming for... four numbers, each of which is the sum of the observations of each colour in each habitat. Make sure you understand that.

5.2.2 ORGANIZING THE DATA FOR PLOTTING AND ANALYSIS

If you understand our aim in summarizing the data, you probably realize that you can use the **group_by**() and **summarise**() functions in the ***dplyr*** package:

```
totals <- lady %>%
  group_by(Habitat, morph_colour) %>%
    summarise(total.number = sum(number))
```

Making a figure of these data is easy. And this is the *only* time we advocate using a bar chart to summarize your raw data (see below). Here's how to do it:

```
ggplot(totals, aes(x = Habitat, y = total.number,
       fill = morph_colour)) +
  geom_bar(stat = 'identity', position = 'dodge')
```

5.2.3 NEW **ggplot**() DETAIL

This figure we've made (Figure 5.1) needs some explanation, doesn't it? How can it be that we just told you never to make a bar chart, and here we are making a bar chart? In the case of count data like this, each bar is one

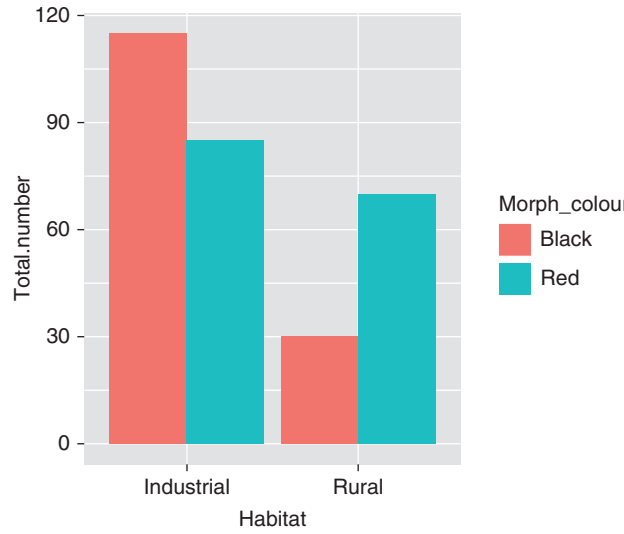

Figure 5.1 Total numbers of red and black ladybirds collected in industrial and rural habitats. Default R colours.

point, we're not summarizing a distribution, and the counts are ratio data (they have a 'real' zero). It *is* a logical way to view these data, particularly as they have (have you noticed?) no 'variation' to examine. In this situation, we think a bar chart starting at zero is actually a good option.

The **ggplot**() syntax has some new things to which to pay attention. First, we see a new part of the aesthetic, fill = morph_colour. This is used when the 'geometry' is something like a bar, and can be filled. Incidentally, colour = morph_colour with a bar alters the *outline* of the bar, not the fill colour. Remember that if you want to avoid future frustration.

Second, **geom_bar**() has some arguments that are worthy of explanation. stat = 'identity' tells ggplot *not* to try and calculate anything from the data. **ggplot**() *can* do stuff if you let it, but in this case we want **ggplot**() to use *just what we've given it to use*. position = 'dodge' is not a reference to any games or US cities. It is a request to put the two bars in each Habitat group (e.g. black and red counts) next to each other. If you don't use the position = 'dodge' option you'll end up with a stacked barplot.

5.2.4 FIXING THE COLOURS

The colours in Figure 5.1 were chosen by R, and are far from ideal. Black is red, and red is blue!!! Let's make black colour correspond to black ladybirds, and red to red ones. That is, we customize the colours used to represent groups of data.

When working with discrete groups (as in the variables Habitat and colour_morph), ***ggplot2*** has built-in scale_ functions ending with _manual. We'll provide way more info about scale_'s in Chapter 8. Here we use scale_fill_manual; it takes an argument called values that is a set of colours to use:

```
ggplot(totals, aes(x = Habitat, y = total.number,
      fill = morph_colour)) +
  geom_bar(stat = 'identity', position = 'dodge') +
  scale_fill_manual(values = c(black = "black", red = "red"))
```

Lovely (Figure 5.2)! Note that, in this case, we specify values = c(black = "black", red = "red"). This seems rather obvious, perhaps, but we are

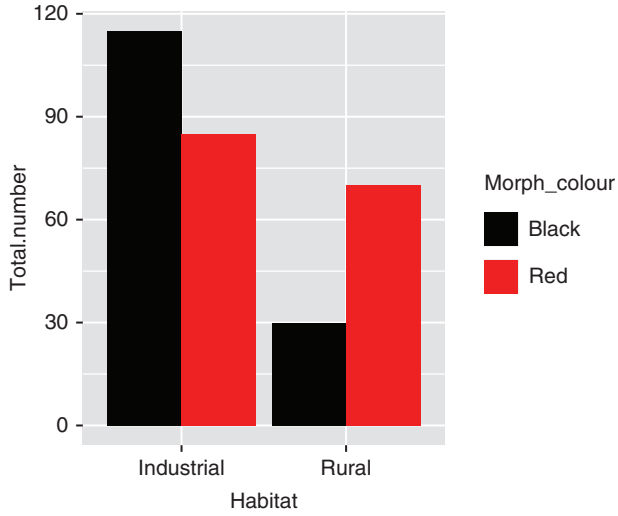

Figure 5.2 Total numbers of red and black ladybirds collected in industrial and rural habitats. Colours manually specified.

assiging the colours to the appropriate morph_colour. If, for example, the morphs of the ladybirds were large and small, and we wanted to use 'black' for large and 'red' for small morphs, we would use values = c(large = "black", small = "red").

5.2.5 INTERPRETING THE GRAPH (GUESS THE ANSWER BEFORE WE DO STATS)

Right . . . now back to biology. At this point, we can ask ourselves whether we think the null hypothesis—no association between colour morphs and habitat, the same relative abundance of black and red morphs in each habitat—is true or not. Look at Figure 5.2, and decide for yourself.

It certainly looks like the black morph is more common in the industrial habitat, relative to the red morph. Can you see this? Make sure you can—if you are confused, remember that a χ^2 test compares the frequencies/proportions, not the absolute numbers. This suggests that the morphs are *not* equally distributed among the habitats. We are expecting to *reject* the null hypothesis.

5.2.6 MAKING THE χ^2 TEST

To actually test this hypothesis, we need to do two things. Step 2 is to use the function **chisq.test**(). But, to make this function do a two-way contingency test, we must give it a matrix of the total counts. Although they can look superfically similar, a matrix is different from a data frame. At present, because all ***dplyr*** functions take and give back a data frame, we have all four totals in a single column of a three-column data frame:

```
totals

## Source: local data frame [4 x 3]
## Groups: Habitat [?]
##
##      Habitat morph_colour total.number
```

```
##        (fctr)        (fctr)         (int)
## 1 Industrial         black           115
## 2 Industrial           red            85
## 3       Rural         black            30
## 4       Rural           red            70
```

We introduce one trick here that transforms this data frame into the matrix we need for the χ^2 test: the function **xtabs**(). If you have used Excel, you may be familiar with a pivot table for doing cross-tabulation. **xtabs**() is cross(x)-tabulation:

```
lady.mat <- xtabs(number ~ Habitat + morph_colour,
                  data = lady)
lady.mat

##             morph_colour
## Habitat      black red
##   Industrial   115  85
##   Rural         30  70
```

xtabs() requires a simple formula and a data frame. The data frame in this situation is the raw data from above, lady (although it would also work just fine with the summarized data, totals). The formula reads, '*Please cross-tabulate* the number column of counts in the totals data frame by the Habitat and morph_colour variables.' This carries out exactly the same calculation as the **group_by**() and **summarise**() functions did to make totals, but now we end up with a matrix. Nice.

Now the χ^2 test:

```
chisq.test(lady.mat)

##
##  Pearson's Chi-squared test with Yates' continuity
##  correction
##
## data:  lady.mat
## X-squared = 19.103, df = 1, p-value = 1.239e-05
```

This rather sparse output provides all the information we need in order to present our 'result'. According to the test, there is a very small

probability ($p = 0.00001239$) that the pattern we see is consistent with the null hypothesis; that is, if there really were no association between colour morph and habitat, and we carried out the sampling process again and again, we would get such a similar (or more extreme) result only once every 10,000 or so samples. This is a pretty good indication that it probably isn't a chance result. The result allows us to reject the null hypothesis and conclude that there is some association. Our figure suggests that black morphs are more frequent in industrial habitats, while red morphs are more frequent in rural habitats.

What might we report in a manuscript? 'We tested the hypothesis that there is an association between colour morphs of ladybirds and industrial and rural habitats. Ladybird colour morphs are not equally distributed in the two habitats ($\chi^2 = 19.1, df = 1, p < 0.001$), with black morphs being more frequent in industrial habitats and red morphs more frequent in rural habitats (Figure 5.1).'

By the way, if you are wondering what that 'Yates' continuity correction' is all about, it refers to a little adjustment of the χ^2 test that makes it a bit more reliable when the counts are small. For those of you who are familiar with the mechanics of the χ^2 test (row sums, column sums, observed and expected values), you will be pleased to know that all of these are accessible to you, simply by assigning the values returned by **chisq.test**() to a name and then looking at these:

```
lady.chi <- chisq.test(lady.mat)
names(lady.chi)

## [1] "statistic" "parameter" "p.value"   "method"
## [5] "data.name" "observed"  "expected"  "residuals"
## [9] "stdres"

lady.chi$expected

##             morph_colour
## Habitat         black       red
##   Industrial 96.66667 103.33333
##   Rural      48.33333  51.66667
```

5.2.7 FROM DATA TO STATISTICS: AN OVERVIEW

We had two main points to make with this introduction to the χ^2 contingency table analysis: to remind you to plot your data first and to implore you to think about how to translate the biological question you are asking into a statistical hypothesis. Hopefully you've grasped the importance of each, and we have reinforced some basic details about the χ^2 contingency table analysis. Most importantly, this example should have shown you the ease with which you can gain understanding of your data using R. You now have a script/recipe for processing, graphing, and analysing data such as these . . . you have saved your script, yes?

5.3 Two-sample *t*-test

As above, we wish to make three points with this introduction to the two-sample *t*-test. First, always plot your data. Second, check model assumptions—vital for reliable interpretation. Finally, R makes all of this easy.

The two-sample *t*-test is a comparison of the means of two groups of numeric values. It is appropriate when the sample sizes in each group are small. However, it does make some assumptions about the data being analysed. The standard two-sample, Students *t*-test assumes that the data in each group are normally distributed and that their variances are equal. We will come back to these issues shortly but, immediately, you might think about the graph you want to make of these data, showing the distributions. Perhaps two histograms?

5.3.1 THE *t*-TEST DATA

First, some data. Here we will analyse ozone levels in gardens distributed around a city. The gardens were either to the west of the city centre or to the east. The data are ozone concentrations in parts per hundred million (pphm). Ozone in excess of 8 pphm can damage lettuce plants—they bolt and get filled with latex and are yucky to eat. We are interested in whether there is a difference in the average ozone concentration between gardens in the east and the west.

As above, let's read in the data and check their structure. The `ozone.csv` data can be found in the collection of datasets at `http://www.r4all.org/the-book/datasets` —you've probably already downloaded it. Remember to set up a new script, clear R's brain, define the packages you want to use, set the working directory, and get the data.

Following the convention we established above, we assign the data to a name (object), called ozone. Once the data are into R (using read.csv), examine them using one of our suggested tools, such as **glimpse**(). Refer to the example script above to organize the first few lines:

```
glimpse(ozone)

## Observations: 20
## Variables: 3
## $ Ozone           (dbl) 61.7, 64.0, 72.4, 56.8, 52.4, 4...
## $ Garden.location (fctr) West, West, West, West, West, ...
## $ Garden.ID       (fctr) G1, G2, G3, G4, G5, G6, G7, G8...
```

The output from **glimpse**(ozone) reveals that our data are in a data frame format with three columns and that Garden.location is a factor with two levels. R has assumed (because they are not numbers) that the data in the Garden.location column should be treated as a factor—a code denoting a categorical grouping variable within the dataset.

5.3.2 THE FIRST STEP: PLOT YOUR DATA

The first step in an analysis of data is . . . making a figure! We can see that there are two different locations of gardens: East and West. To initiate a comparison of ozone levels between East and West gardens, we might consider a plotting method that allows us to see the central tendency of the data and the variability in the data, for each location. Since there are only two groups, a good tool for this is the histogram.

Recall that in Chapter 4 we introduced the histogram *and* facets, a tool to break data into groups during the plotting process. Let's take advantage of this. Feel free to first check back in Chapter 4.

If we stack the histograms on top of each other, we should be able to (a) see whether the means seem different and (b) assess whether the data in each location appear to be normal and have similar variance. We thus accomplish the process of visualizing our hypothesis and evaluating some assumptions of a two-sample *t*-test, all in one effort:

```
ggplot(ozone, aes(x = Ozone)) +
  geom_histogram(binwidth = 10) +
  facet_wrap(~ Garden.location, ncol = 1) +
  theme_bw()
```

The resulting Figure 5.3 provides visual representations of the distributions of the data in each sample, and from these it seems reasonable to say that the assumption of normality and equality of variance has been met. Let us assume that it has. Of course, R has functions to statistically evaluate normality and equality of variance. And we bet you can guess what ncol = 1 means . . .

The second reason for plotting the data like this was to provide some indication of whether our null hypothesis—that there is no difference in the mean ozone levels between the two groups—is true or not. A cursory

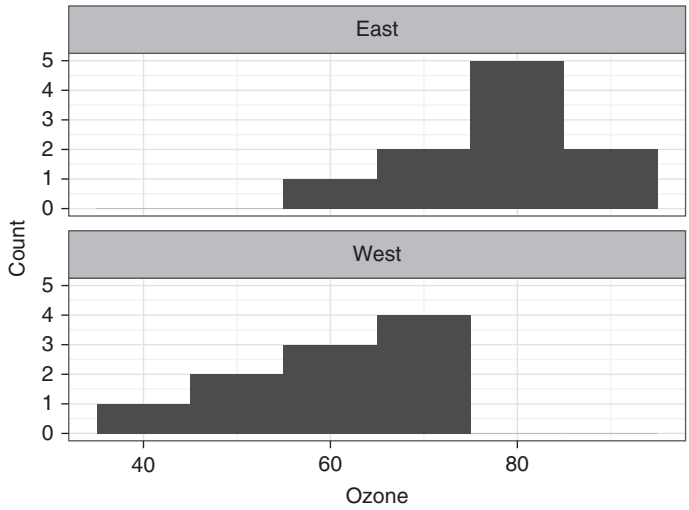

Figure 5.3 Histograms of ozone levels in gardens in East and West garden locations.

glance at the figure we've produced might suggest that maybe we will be able to reject the null hypothesis, i.e. the peaks of the two histograms are at different locations on the *x*-axis. However, there is also considerable overlap between the two histograms. We need some stats. Perhaps at this point you might generate some code, using ***dplyr*** (e.g. **group_by**() and **summarise**()) to calculate the means and standard errors of the ozone levels in each location? The previous chapters have equipped you to do this.

5.3.3 THE TWO-SAMPLE *t*-TEST ANALYSIS

We now have an informative figure and have made an informal assessment of whether we might reject the null hypothesis. If you've not done this, go do it! To do the *t*-test, use the **t.test**() function in R. You may even want to read the help file! Add the following to your script:

```
# Do a t.test now....
t.test(Ozone ~ Garden.location, data = ozone)
```

First, let's look at the arguments we give to the **t.test**() function: we provide a 'formula' and a data argument. The data argument specifies the data frame in which the variables reside. We know that. The formula is a representation of our hypothesis: do ozone levels (Ozone) vary as a function of location (Garden.location)? That is how the ~ makes us read the formula.

Let us spend a bit of time reviewing the output as a way to reinforce aspects of our workflow. We will call this the anatomy of the R output:

```
##
##  Welch Two Sample t-test
##
## data:  Ozone by Garden.location
## t = 4.2363, df = 17.656, p-value = 0.0005159
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##   8.094171 24.065829
## sample estimates:
## mean in group East mean in group West
##              77.34              61.26
```

The first line of the output tells us the statistical test performed: a Welch two-sample *t*-test. This is what we expected, apart from the word 'Welch': this is a clear indication that R is doing something slightly unexpected, and we should not ignore such an alarm bell. We'll cover this later, however.

Next we see that the data that have been used are declared—this is a good way to confirm that you've analysed the data you wanted to analyse. The next line provides the traditional *t*-test statistic, degrees of freedom, and *p*-value. We assume you know what these mean. However, let us use the next few lines of output to make a few clear points about the two-sample *t*-test. The output declares the alternative hypothesis to us: that the true difference in means is not equal to 0. This should help you understand more of what you've been doing. If the difference between two things is 0 then they are the same, and we have no grounds for rejecting the null hypothesis.

Next in the output is a 95% confidence interval. This interval is around *the difference between the two means*. Keep in mind that the difference would be 0 if the means were the same. The fact that this interval does not include 0 provides an answer to our initial question. We can conclude that they are probably different. This falls in line with the test statistic and associated *p*-value. Finally, the output provides the means in each group.

Now, back to the word 'Welch'. A quick look in the help files (`?t.test`) or on Wikipedia reveals that this method allows one of the assumptions of the standard two-sample *t*-test to be relaxed—that of equal variance. While you may have made the effort to assess the assumption of equal variance above, and found the variances relatively similiar, you now know that there are options for when this assumption is not met!

You might have once been taught to test for equality of variance in a two-sample *t*-test. It isn't necessary to do this when using the Welch version—and, actually, we don't think it is ever a very good idea—but if feel you must make a formal test, there are several functions you could use.

For example, the var.test() function has the same structure as the *t*-test function:

```
var.test(Ozone ~ Garden.location, data = ozone)
```

5.3.4 *t*-TEST SUMMARY

A two-sample *t*-test is often assumed to be easy. If you can collect data in a manner that allows you to test a fundamental hypothesis using *t*-tests, this is certainly a good thing. But don't let the workflow of a good analysis slip just because it is simply a comparison of two groups: always plot your data, evaluate assumptions, and only then interpret your analysis results.

5.4 Introducing . . . linear models

The previous two sections focused on relatively simple questions, and we continue doing that here, but moving into models known as 'general linear models'. General linear models are a class of model that includes regression, multiple regression, ANOVA, and ANCOVA. All of these are fundamentally linear models. These models share a common framework for estimation (least squares) and a common set of assumptions, centred around the idea of normally distributed residuals. The next two examples introduce the assessment of these assumptions as part of our *Plot -> Model -> Check Assumptions -> Interpret -> Plot Again* workflow.

A moment to anticipate confusion. Don't confuse the *general* linear model with the *generalized* linear model, known as the GLM. We introduce the GLM, where key assumptions about normality are relaxed, in Chapter 7.

Perhaps the best way forward is just to get stuck in to the next two examples. We will be using several new functions, but the most important one is lm(), which, by the looks of it, is a tool to fit linear models.

5.5 Simple linear regression

The example we are going to use is one that asks whether plant growth rates vary with soil moisture content. The underlying prediction is that more moisture will likely allow higher growth rates (we like simple examples). Two features of these data are important. The first is that we have a clear relationship specified between the two variables, one that is easily visualized by plotting the response (dependent) variable—plant growth rate—against the explanatory (independent) variable—soil moisture content. The second is that the explanatory variable is a continuous, numeric variable. It doesn't have categories.

5.5.1 GETTING AND PLOTTING THE DATA

The dataset is available from the same place as all the previous ones; you already have it. It is called `plant.growth.rate.csv`. Go ahead and get these data, start a linear models script, clear the decks, make the ***dplyr*** and ***ggplot2*** packages available, and import the data.

If all goes to plan, you should be able to see the data via **glimpse**() or any of the other tools we introduced in Chapters 2 and 3. We'll call the data frame plant_gr:

```
glimpse(plant_gr)

## Observations: 50
## Variables: 2
## $ soil.moisture.content (dbl) 0.4696876, 0.5413106, 1.6...
## $ plant.growth.rate     (dbl) 21.31695, 27.03072, 38.98...
```

We can see that the structure of the data is as expected, with two continuous variables. Making a figure of these data involves producing a scatterplot, using **geom_point**(). Not much to it, really. Feel free to change colours or point sizes, just to practice some of the basic ***ggplot2*** tools we've

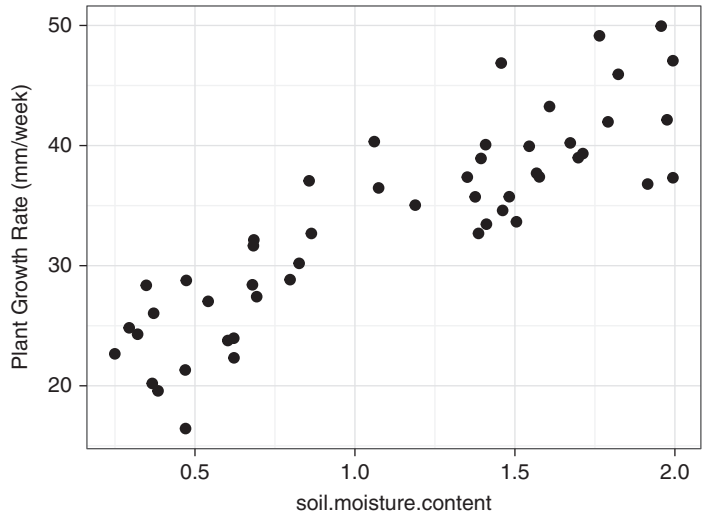

Figure 5.4 An exploratory graph needn't be beautiful. Clarity and speed are priorities.

introduced. We've added one annotation—introducing units of mm/week to the growth rate data (Figure 5.4):

```
ggplot(plant_gr,
       aes(x = soil.moisture.content, y = plant.growth.rate)) +
          geom_point() +
          ylab("Plant Growth Rate (mm/week)") +
          theme_bw()
```

5.5.2 INTERPRETING THE FIGURE: BIOLOGICAL INSIGHT

Looking at the resulting Figure 5.4, there are a few things that you, a budding data scientist and interpreter of all things data, can do. First, you should realize that the gradient (i.e. the slope) is positive. Indeed, the more moisture in the soil, the higher the plant growth rate. Good. Biology seems to be working.

Second, you may not realize this, but you can actually pre-empt the statistical analysis by guestimating the slope and intercept. Check it out. *Roughly speaking*, the growth rate varies between 20 and 50 mm/week.

Roughly speaking, the soil moisture varies between 0 and 2. *Thus*, the gradient is, *roughly speaking*, 30/2 = 15. And the intercept is somewhere, *roughly speaking*, between 15 and 20 mm/week. Gooood. This is always a good idea. Try and examine your data *before* you do the analysis. Push yourself even further by figuring out, in advance, the degrees of freedom for error that you expect. Hint: it's the number of data points minus the number of parameters (also called coefficients) estimated.

5.5.3 MAKING A SIMPLE LINEAR REGRESSION HAPPEN

This next step is rather straightforward. We use the function **lm**() to fit the model. The **lm**() function is very much like xtabs and t.test; it needs a formula and some data:

```
model_pgr <- lm(plant.growth.rate ~ soil.moisture.content,
                data = plant_gr)
```

This reads, 'Fit a linear model, where we hypothesize that plant growth rate is a function of soil moisture content, using the variables from the plant_gr data frame.' Nice. And. Simple.

5.5.4 ASSUMPTIONS FIRST

Now, we know you want to rush ahead and find out whether our a priori estimates of the intercept and gradient/slope were correct. And whether, as aeons of biology would suggest, growth rate increases with soil moisture content. But WAIT, we say. Do not rush excellence.

First, check the assumptions of the linear model. ***ggplot2*** needs a little help with this, as it doesn't know what linear models are. Help comes from ***ggfortify***, and its **autoplot**() function, which, when given a linear model created by **lm**(), produces four very useful figures (Figure 5.5). For those of you familiar with base graphics or with the previous version of the book, these are the same four pictures that **plot**() produces when it is provided with an **lm**() model. We suggest that after installing ***ggfortify***, you add **library**(ggfortify) to your list of libraries at the start of every script.

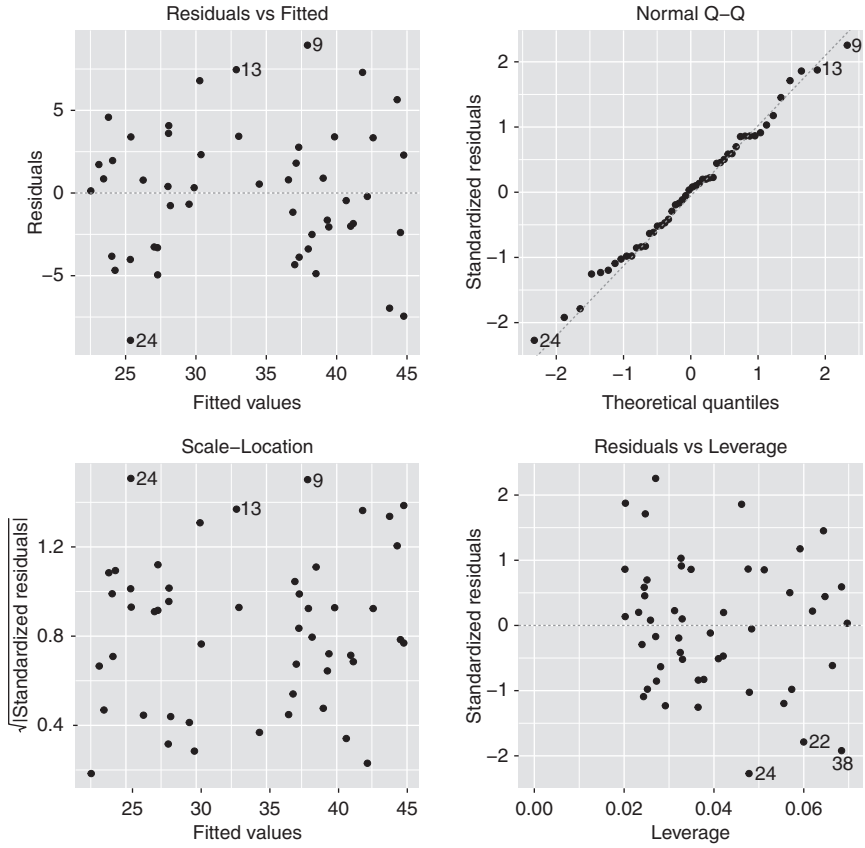

Figure 5.5 Nothing looks too bad in the diagnostic plots.

Here's what you need to do to produce the plots now:

```
library(ggfortify)
autoplot(model_pgr, smooth.colour = NA)
```

What, pray tell, do these plots mean? We actually hope you know what some of them are. They are all based around the residuals—errors around the fitted line. We assume you *do* know what residuals are, and perhaps their relationship to sums of squares and mean square errors. We'll revisit the **smooth . colour** = **NA** argument below.

Here we go:

1. *Top left*. This panel is about the 'systematic part' of the model; it tells us whether a line is appropriate to fit to the data. If things have gone wrong, hump-shapes or valleys will be apparent. These would mean that the structure of your model was wrong. For example, fitting a straight line didn't work. Lots of people suggest you look at this to evaluate the assumption of equal variance—homo- vs heteroskedasticity. But there is a better (bottom left) for this.
2. *Top right*. This evaluates the assumption of normality of the residuals. The dots are the residuals, and the dashed line the expectation under the normal distribution. This is a *much* better tool than making a histogram of the residuals, especially with small sample sizes . . . like less than 100.
3. *Bottom left*. This evaluates the assumption of equal variance. The *y*-axis is a standardized (all positive) indicator of the variation. Linear models assume that the variance is constant over all predicted values of the response variable. There should be no pattern. But there might be one if, for example, the variance increases with the mean, as it might with count data (see Chapter 7).
4. *Bottom right*. This evaluates leverage, a tool not only to detect influential data points, ones that move the gradient more than might be expected, but also to detect outliers. Your friends, mentors, or supervisor might think this is important. If they do, speak with them . . .

If you still feel confused, don't know what these mean, or have relatively little experience with assessing the assumptions the of a general linear model, head for the stats books such as those of Crawley (2005 and 2012), Faraway (2014), and Dalgaard (2008) (see Appendix 2, 'Further Reading', for details of these books).

Overall, the take-home message from this example, and these plots, is that everything is just fine. There is no pattern in either of the left-hand

plots. The nomal-distribution assumption is clearly met. And there are no points exerting high influence. Of course, you can test for normality (e.g. by the Kruskal–Wallis or Anderson–Darling test), but we leave that to you, should you wish to or be told to pursue these tests (you might have guessed by now that we're not a fan of them).

As promised, we will tell you about **smooth.colour** = **NA**. In the absence of this argument, the default presentation of the diagnostic plots includes a 'wiggly line' fitted by locally weighted regression. The `= NA` suppresses the line. Many people find this line useful, but we have found that they also tend not to look at the data points. Furthermore, the lines apparently talk to them and tell them there are problems when there really are none. We find, in a nutshell, that omitting these lines is quite beneficial. When there are problems, you can see them without the (un)helpful line (see Chapter 7).

5.5.5 NOW THE INTERPRETATION

Phew. Now we are ready. Ready to see whether we can reject the null hypothesis that soil moisture has no effect on plant growth rate. Ready to see if our guesses were right, to assess the real estimate of plant growth at zero soil moisture (the intercept) and the change of growth rate with soil moisture (the gradient/slope).

We do all of this using two tools, tools we will use for every general (and generalized) linear model from here on in and out: **anova**() and **summary**().

Now, wait a minute, you say. **anova**()? Yes. **anova**(). Repeat after us: **anova**() does not perform an ANOVA. Or, more accurately, it does not carry out the type of ANOVA that compares means. Again? OK. You get it. **anova**() produces a classic table in statistics, the sums-of-squares table. It provides the overall F-value for the model, representing the ratio of variance explained by the explanatory variables to the leftover variance. It also produces an estimate of R^2 and the adjusted R^2.

summary() is far less contentious. It produces a table of the estimates of the coefficients of the line that is 'the model': an intercept and a slope. Of course, there is a bunch of other stats stuff along with that too. But let's see how they all come together.

First the **anova**() table:

```
anova(model_pgr)

## Analysis of Variance Table
##
## Response: plant.growth.rate
##                       Df  Sum Sq Mean Sq F value    Pr(>F)
## soil.moisture.content  1 2521.15 2521.15  156.08 < 2.2e-16
## Residuals             48  775.35   16.15
##
## soil.moisture.content ***
## Residuals
## ---
## Signif. codes:
## 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

If you've had any course in stastistics (we hope you have), this should be familiar. The ANOVA table is a classic assessment of the hypothesis, presenting the *F*-value, degrees of freedom, and *p*-value associated with the explanatory variables in the model (in this case, the model has one explanatory variable).

We can see here a rather large *F*-value, indicating that the error variance is small relative to the variance attributed to the explanatory variable. This, with the single degree of freedom, leads to the tiny *p*-value. As we noted in the χ^2 example, if there really was *no* relationship between plant growth rate and soil moisture, and we were to carry out the sampling process again and again, we would get such a large *F*-value fewer than one in a million or more samples. This is a pretty good indication that the pattern we are seeing probably isn't a chance result.

And now the **summary**() table:

```
summary(model_pgr)

##
## Call:
```

```
## lm(formula = plant.growth.rate ~ soil.moisture.content,
   data = plant_gr)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -8.9089 -3.0747  0.2261  2.6567  8.9406
##
## Coefficients:
##                        Estimate Std. Error t value Pr(>|t|)
## (Intercept)              19.348      1.283   15.08   <2e-16
## soil.moisture.content    12.750      1.021   12.49   <2e-16
##
## (Intercept)            ***
## soil.moisture.content ***
## ---
## Signif. codes:
## 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.019 on 48 degrees of freedom
## Multiple R-squared:  0.7648, Adjusted R-squared:  0.7599
## F-statistic: 156.1 on 1 and 48 DF,  p-value: < 2.2e-16
```

The estimates provided in the summary table (the first column) correspond to the estimates, in a linear regression, of the intercept and slope associated with the explanatory variable. Can you guess which is the intercept . . . More seriously, make sure you understand that the gradient/slope is associated with the explanatory variable, in this case soil moisture, which is also the x-axis of our figure, the values of which are associated with differences in plant growth rate.

Recalling our guesses from the figure we made, we've done pretty well. *Roughly speaking*, we predicted a gradient/slope of 30/2 = 15 vs 12.7 estimated by least squares, and an intercept between 15 and 20 mm/week vs 19.34 estimated by least squares. Making that figure makes our statistics confirmatory of our understanding of our data. Did R also give you the correct degrees of freedom for error?

The t- and p-values provide tests of whether, for example the gradient/slope is different from zero, and we can see that it is. In fact, we might even report something like the following:

> Soil moisture had a positive effect on plant growth. For each unit increase in soil moisture, plant growth rate increased by 12.7 mm/week (slope = 12.7, $t = 12.5$, d.f. = 48, $p < 0.001$).

In case you were wondering, it is also be fine to report the *F*-value, degrees of freedom, and the *p*-value from **anova**(). They test exactly the same thing in this example. This isn't generally true of general linear models, though.

5.5.6 FROM STATS BACK TO FIGURE

The final step in our workflow involves translating the model we have fitted back onto our figure of the raw data. In this simple linear regression example, ***ggplot2*** comes to the rescue. For more complicated models, we need to do something different (see Chapters 6 and 7), but for now, let's introduce one more feature of ***ggplot2***: the capacity to add regression lines (Figure 5.6):

```
ggplot(plant_gr, aes(x = soil.moisture.content,
           y = plant.growth.rate)) +
  geom_point() +
  geom_smooth(method = 'lm') +
  ylab("Plant Growth Rate (mm/week)") +
  theme_bw()
```

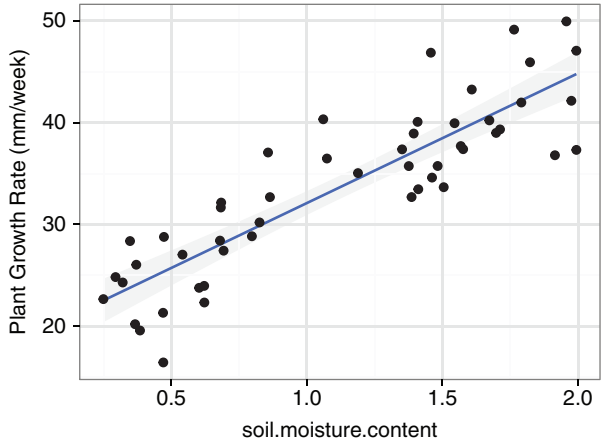

Figure 5.6 ***ggplot2*** provides the functionality to add the fitted values and standard error of the fit to a figure when it is a simple one-explanatory-variable model.

The extra, new feature we've introduced is the layer **geom_smooth**(method = "lm"), which is ***ggplot2***-speak for 'shove a linear-model fitted line, and the standard error of the fit, using flash transparent grey, onto my graph'. It is lovely and easy, isn't it?

Top cautionary tips. This is very handy. And the **geom_smooth**(method = "lm") tool is outstanding for this very simple example of a single explanatory variable. It is also outstanding for data exploration–i.e. *before* you make models, as it works brilliantly with **facet_wrap**(). But . . . don't expect it to work correctly with more complicated models. We will, as promised, introduce the safe, robust method for adding model fits to your figure. Be patient . . . we know you almost cannot wait.

OK. So, we've introduced, via a simple linear regression, some very classic tools for making inferences. We learned how to fit a linear model with **lm**(). We looked at four plots (produced by the **autoplot**() function) that tell us about how well model assumptions are met. We learned how to assess what **lm**() does using **anova**(), which doesn't fit an ANOVA, and **summary**(), which returns our estimates of the slope/gradient and intercept. Phew. And one more ***ggplot2*** trick. Nice.

Are you ready for one more example? Sure you are. Take a break if you need. We did . . .

5.6 Analysis of variance: the one-way ANOVA

The final example in this chapter is a one-way ANOVA. The one-way ANOVA is as simple as the previous example, but we will see one change in the variables in the data frame: the explanatory variable is no longer continuous. It is a factor, or categorical variable.

You've experienced these types of variables already, using the `compensation.csv` data frame and, specifically, the Grazing variable, which had two levels, Grazed and Ungrazed. We continue that idea here, but with more levels. The dataset we are going to use centres on water fleas, also known more 'scientifically' as *Daphnia* spp., and their parasites, which have lots of cool names.

5.6.1 GETTING AND PLOTTING THE DATA

The question we are asking focuses on water flea growth rates and has two parts. First, we are asking generally whether parasites alter growth rates. Second, because it is a well-replicated and designed experiment, we can also ask whether each of three parasites reduces growth, compared with a control, no parasite treatment. We are going to use the same *Plot -> Model -> Check Assumptions -> Interpret -> Plot Again* workflow. Let's get started. Grab the `Daphniagrowth.csv` data from the same place as you get all the datasets, set up your script, make ***dplyr***, ***ggplot2***, and ***ggfortify*** available, clear the decks and check out the data... phew. This step is a lot easier now, isn't it? We are going to call the data *daphnia*:

```
glimpse(daphnia)

## Observations: 40
## Variables: 3
## $ parasite    (fctr) control, control, control, control...
## $ rep         (int) 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 1, 2...
## $ growth.rate (dbl) 1.0747092, 1.2659016, 1.3151563, 1....
```

We can see that the data frame has three variables, two of which we want to use for figure making, growth.rate and parasite. The other, rep, indicates the replication in each treatment level. Making a figure of these data needs a bit of thinking before jumping in. Recall Chapter 4, where we introduced the box-and-whisker plot as a quick and effective tool for viewing variation in a response variable as a function of a grouping, categorical variable? That's probably the start we want to make (Figure 5.7):

```
ggplot(daphnia, aes(x = parasite, y = growth.rate)) +
  geom_boxplot() +
  theme_bw()
```

This looks excellent, and it's just what we expected. Save for the fact that the parasite names are *huge* and mashed together like sardines. *Not. Good. Need. Quick. Fix.* Chapter 8 is going to provide a series of extra tools to manipulate the axis attributes. We are going to introduce here a more bizarre,

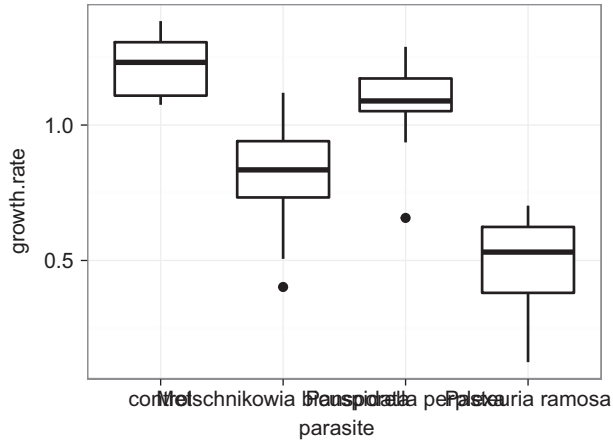

Figure 5.7 Daphnia's parasites alter daphnia growth rate.

and fun, trick. It's called coordinate flipping. Having made the graph in a way that works perfectly for **geom_boxplot**(), we just are going to switch the *x*- and *y*-axes using **coord_flip**(). This may seem novel to some of you, but it can be a very good way to visualize categorical data:

```
ggplot(daphnia, aes(x = parasite, y = growth.rate)) +
  geom_boxplot() +
  theme_bw() +
  coord_flip()
```

That's better. It's not perfect, but we can certainly read the names now AND we can start the process of second-guessing the statistics!

The first thing we can note is that there is substantial variation in the daphnia growth rates among the four treatments. Second, we can see that the control treatment produces the highest growth rate, about 1.2 mm/day. Third, we can see that *P. perplexa* is closest and perhaps lower, *M. bicuspidata* (try saying that first name!) is next lowest, and *P. ramosa* is definitely lower. Or we believe so. The point is, we can see that there is likely to be a parasite treatment effect overall (question 1), and an ordering in the growth rates, with parasites generally driving down the growth rate, and *P. ramosa* < *M. bicuspidata* < *P. perplexa* (question 2).

You can go even further if you wish, estimating the average growth rate for each treatment (looks like about 1.2 for the control treatment), and the difference of the parasite treatments from the control treatment (i.e. the effects of each parasite on growth rate). You could also figure out the degrees of freedom for the treatment and the degrees of freedom for error. It's very good practice to do this now, and to always do it, and then check R reports what you expect.

With that in mind, let's construct our linear model that does a one-way ANOVA. Don't freak out. The function that performs a one-way ANOVA is . . . **lm**().

5.6.2 CONSTRUCT THE ANOVA

This should be familiar. It has exactly the same structure as the linear regression model, except that 'we know' that the explanatory variable is a categorical variable:

```
model_grow <- lm(growth.rate ~ parasite, data = daphnia)
```

5.6.3 CHECK THE ASSUMPTIONS

That was, like, the shortest section ever. And here we are again, ready to evaluate the set of assumptions associated with linear models. They are the

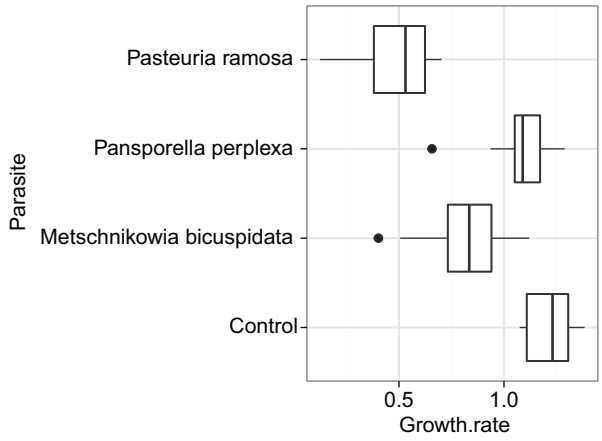

Figure 5.8 Perhaps a nicer graph of how daphnia's parasites alter daphnia growth rate.

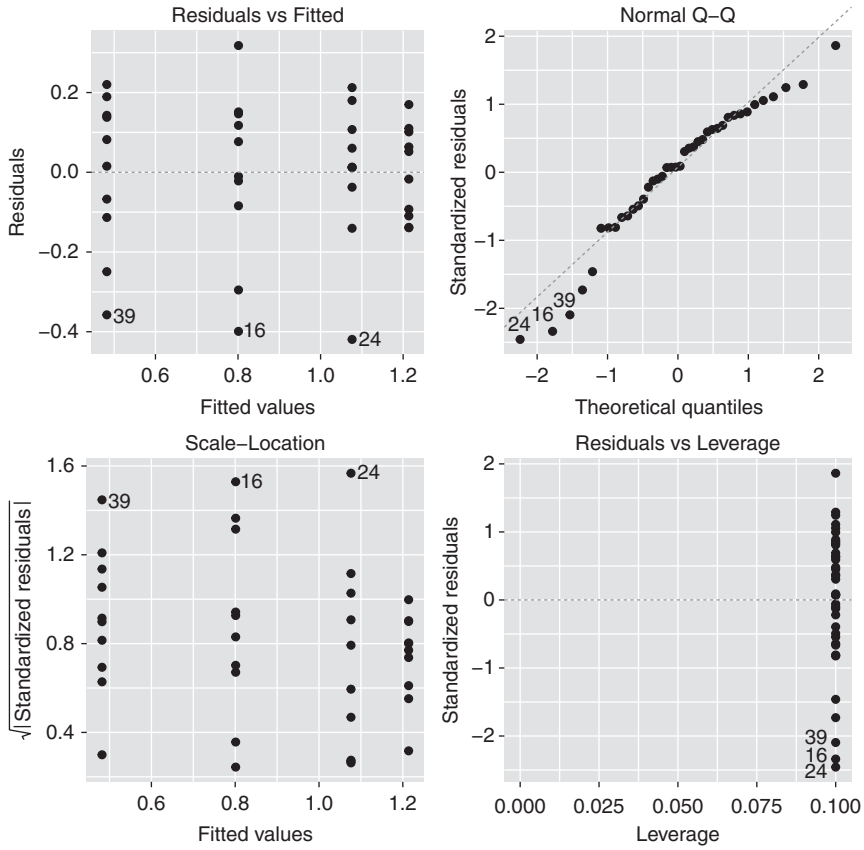

Figure 5.9 Nothing looks too bad in the diagnostic plots.

same for a regression and for a one-way ANOVA. We construct them and evaluate them in exactly the same way as above, by plotting four diagnostic graphs (Figure 5.9):

```
autoplot(model_grow, smooth.colour = NA)
```

If this didn't work, then you've not added **library**(ggfortify) to your script . . .

Briefly, these figures suggest everything is probably fine. You may be uncomfortable with the Q–Q plot, upper right, evaluating the normality of the residuals. You can trust us that this pattern is within the expected

bounds of variation found in samples of the normal distribution. If there is a 'real' departure from normality, it isn't too excessive, so our all-important *p*-values should be OK. Or, you could read more, simulate stuff, and prove to yourself that it's OK.

We are going to consider them fine, and move on to interpreting the output of the two tools for making inferences from linear models, **anova**() and **summary**(). Yes, we are going to apply the function **anova**() to a one-way ANOVA.

5.6.4 MAKING AN INFERENCE FROM A ONE-WAY ANOVA

As above, we'll start with the use of **anova**(). This function, applied to our model, will provide an answer to our first question posed above: is there an effect at all of our treatments?

```
anova(model_grow)

## Analysis of Variance Table
##
## Response: growth.rate
##           Df Sum Sq Mean Sq F value    Pr(>F)
## parasite   3 3.1379 1.04597  32.325 2.571e-10 ***
## Residuals 36 1.1649 0.03236
## ---
## Signif. codes:
## 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The output looks remarkably similar to that from a regression, and it should. They are both just linear models. We can see that there is indeed evidence that the parasite treatment, comprising four levels of manipulation, has produced an effect.

It is worth considering exactly what the null hypothesis is for a one-way ANOVA: that all of the groups come from populations with the same mean. The *F*-value quantifies the ratio of the between-group variance to the within-group variance. As the former is large relative to the latter, this produces a large *F*-value, and thus a small *p*-value that allows us to reject the null hypothesis that there are no differences.

Moving along to our second question, what are the effects? This is a question that can be answered in many ways. We'll show you one, and discuss a few others. The one we'll show you involves understanding how R presents coefficients of linear models with categorical explanatory variables.

5.6.5 TREATMENT CONTRASTS

Every statistical package has to decide how to produce the information in a summary-like table. There are many ways to do this, and these 'ways' are known as contrasts. R uses a presentation method known as 'treatment contrasts'. Several other statistical programs do as well, including Genstat and ASREML. SAS does not. Minitab does not. This means that if you compare the outputs of various statistical programs, they may well be different. None are wrong. Just different.

If you don't know what contrasts are or mean, they are a way of expressing coefficients taken from statistical models, and there are many types; see Crawley (2012) and Venables and Ripley (2003). Understanding what type of contrast is used by a statistical package is required for interpreting the summary table. This is also important when you make comparisons between the results from R and other packages. We suggest that those of you who are transitioning from other packages take a good look at ?contr.treatment, as this help file will explain R's contrasts and also reveal methods for making R produce output that matches other statistical packages.

Let's learn what R does, and reveal how a bit of luck and these treatment contrasts give us the answer to our second question. Lets begin by getting the summary table:

```
summary(model_grow)

##
## Call:
## lm(formula = growth.rate ~ parasite, data = daphnia)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
```

```
## -0.41930 -0.09696  0.01408  0.12267  0.31790
##
## Coefficients:
##                                  Estimate Std. Error
## (Intercept)                       1.21391    0.05688
## parasiteMetschnikowia bicuspidata -0.41275    0.08045
## parasitePansporella perplexa      -0.13755    0.08045
## parasitePasteuria ramosa          -0.73171    0.08045
##                                  t value Pr(>|t|)
## (Intercept)                       21.340  < 2e-16 ***
## parasiteMetschnikowia bicuspidata  -5.131 1.01e-05 ***
## parasitePansporella perplexa       -1.710   0.0959 .
## parasitePasteuria ramosa           -9.096 7.34e-11 ***
## ---
## Signif. codes:
## 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1799 on 36 degrees of freedom
## Multiple R-squared:  0.7293, Adjusted R-squared:  0.7067
## F-statistic: 32.33 on 3 and 36 DF,  p-value: 2.571e-10
```

Let's start slowly. Notice that there are four rows in the 'table' of coefficients (estimates), and the first row is labelled '(Intercept)'. If you look along that row, you may see a number you recognize: 1.2. Hmm. Interesting. Let's come back to that. Below that are the names of the three parasites. Hmm. Interesting. What's missing? The level of the treatment group called 'control'. But it's not missing. It is labelled '(Intercept)'.

Top Tip with ANOVA. The most important thing to get your head around with 'treatment contrasts' is the alphabet. R just loves the alphabet. In fact, it defaults to presenting things in alphabetical order. In this example, if we look at the alphabetical order of all of the treatment levels, we find that *control* < *M. bicuspidata* < *P. perplexa* < *P. ramosa*. This is the order presented in the table obtained from **summary**() and in the figure. Consistency is good.

So, in an ANOVA framework, we can assume that the word '(Intercept)' represents the first level of the alphabetically ordered treatment levels. We hope that makes sense. If it does, then you are ready for the fun bit.

Treatment contrasts report *differences* between the reference level (in this lucky case, the control) and the other levels. So, in the summary table, the numbers associated with each parasite are differences between growth

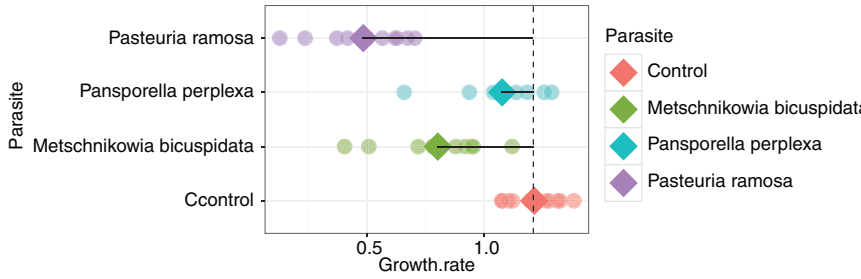

Figure 5.10 Raw growth rates, mean growth rates, and differences between control and parasite manipulations.

rates associated with that parasite and the control. This is why they are all negative. These negative distances, or contrasts, are the lengths of the black lines in Figure 5.10.

You can also get the means for each treatment level/group using ***dplyr***, **group_by**(), and **summarise**(), and then calculate the contrasts yourself:

```
# get the mean growth rates
sumDat<-daphnia %>%
  group_by(parasite) %>%
    summarise(meanGR = mean(growth.rate))

sumDat

## Source: local data frame [4 x 2]
##
##                     parasite    meanGR
##                       (fctr)     (dbl)
## 1                    control 1.2139088
## 2 Metschnikowia bicuspidata 0.8011541
## 3      Pansporella perplexa 1.0763551
## 4          Pasteuria ramosa 0.4822030
```

Now you can calculate by hand how 1.21 (the control growth rate) plus the treatment contrast for a given parasite in the **summary**() gives you the mean growth rate for that parasite. For example, using the *P. ramosa* coefficient (see the summary table above), we take 1.21 – 0.73 = 0.48, which is the mean growth rate for *P. ramosa*. Remember the alphabetical rule, know your reference level, and then watch the numbers materialize.

Back to that summary table one last time. We said we were lucky because the control ended up as the reference group. This means the *p*-values associated with the contrasts are actually useful. They tell us whether the difference between the growth rate for parasite treatment and for the control is significant, i.e. they allow us to ask whether a particular parasite is reducing the daphnia growth rate. However, if you've ever heard of the 'multiple testing problem', you might be inclined to be careful with those *p*-values. Ask a knowledgeable friend for a little guidance if that means nothing to you.

Finally, you may want to produce a figure, something like what we've done above. Aside from the segments, this is rather straightforward, and you can probably figure it out. Note how we've specified two **geom_point**() layers? The first inherits the aesthetics from the daphnia data frame. The second, however, specifies the sumDat data frame we made above, using that as a source of data to plot the mean growth rates, with a big diamond (Figure 5.11).

In the next chapter we cover what to do if you're not so lucky with the word 'control' being the first in the alphabetical listing of the treatment levels. This can be a problem, since it will mean the coefficients reported in the summary table are not so useful, as they won't be differences of treatments from control. Spoiler, the solution involves the **relevel**() function.

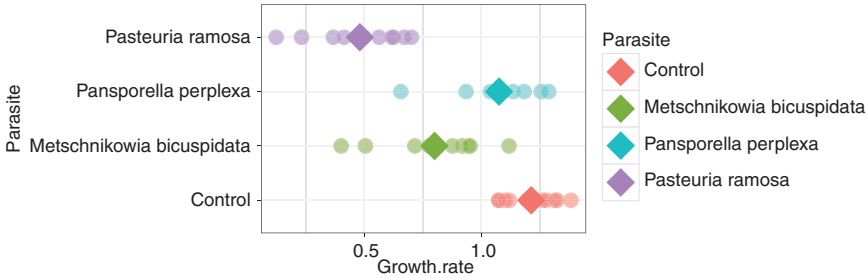

Figure 5.11 Raw growth rates and mean growth rates for each parasite manipulation.

5.7 Wrapping up

The take-home messages from this chapter are:

- Have a consistent workflow for analyses.
- Always make a graph that answers your question before touching any statistical tests.
- Interpret as much as you can, for example slopes, intercepts, contrasts, and degrees of freedom, from your graph *before doing any statistics*.
- Check if your data violate any assumptions, *before* interpreting your model results.
- Make a beautiful graph to communicate your results, and write a biologically focused (rather than statistically focused) sentence describing your results.
- R makes all this very easy.

Appendix Getting packages not on CRAN

There may be times when a package such as ***ggfortify*** may not be available on CRAN; perhaps it has not passed some test, or is having a bad day. For example, during the writing of this book, ***ggfortify*** was unavailable for a while. It had been removed from CRAN for some reason, and when we tried to install it, we got the warning shown in Figure 5.12, and the package didn't install. If this happens, you have a couple of options.

First, if the package was once on CRAN and was archived, an older version may still be available. You can probably find this by googling 'ggfortify cran' and following the links to a rather unfriendly-looking

```
> install.packages("ggfortify")
Warning in install.packages :
  package 'ggfortify' is not available (for R version 3.2.4)
```

Figure 5.12 A warning message that once appeared when we tried to install the ***ggfortify*** package.

web page that lists a few files, one of which was, at the time of writing, `gfortify_0.1.0.tar.gz`. Download this to your computer, making sure you know which folder it gets downloaded into. Then, in RStudio, click the *Install* button in *Packages* (as usual when you want to install a package). However, instead of using *Install from: Repository CRAN* choose *Install from: Package Archive File (.tgz, .tar.gz)*. Then you need to click *Browse* and find the file you just downloaded. Then click *Install*. Hopefully it will still work with the version of R you have installed.

The second option is perhaps rather simpler. Many packages are now developed in an environment/web service called GitHub, and R has an interface for accessing packages in developmental stages directly from GitHub. This is true for ***ggfortify***. Here is how you can access the package that way:

```
install.packages("devtools")
library(devtools)
install_github("sinhrks/ggfortify")
```

First you get and load a new library, ***devtools***. Then you install the package you want from a website called `github`. (`sinhrks` is the GitHub username of the package maintainer and co-author, Masaaki Horikoshi.) This solution will probably give you a development version of the package, rather than the released one. This probably won't matter; you won't notice any difference.