6.5.4. Pearson's correlation coefficient ①

6.5.4.1. Assumptions of Pearson's *r* ①

Pearson's (Figure 6.5) correlation coefficient was described in full at the beginning of this chapter. Pearson's correlation requires only that data are interval (see section 1.5.1.2) for it to be an accurate measure of the linear relationship between two variables. However, if you want to establish whether the correlation coefficient is significant, then more assumptions are required: for the test statistic to be valid the sampling distribution has to be normally distributed and as we saw in Chapter 5 we assume that it is if our sample data are normally distributed (or if we have a large sample). Although typically, to assume that the sampling distribution is normal, we would want both variables to be normally distributed, there is one exception to this rule: one of the variables can be a categorical variable provided there are only two categories (in fact, if you look at section 6.5.7 you'll see that this is the same as doing a *t*-test, but I'm jumping the gun a bit). In any case, if your data are non-normal (see Chapter 5) or are not measured at the interval level then you should use a different kind of correlation coefficient or use bootstrapping.

FIGURE 6.5
Karl Pearson

6.5.4.2. Computing Pearson's *r* using R ①

That's a confusing title. We have already gone through the nuts and bolts of using R Commander and the command line to calculate Pearson's *r*. We're going to use the exam anxiety data to get some hands-on practice.

SELF-TEST

✓ Load the **Exam Anxiety.dat** file into a dataframe called *examData*.

Let's look at a sample of this dataframe:

```
   Code Revise Exam Anxiety Gender
1     1      4   40  86.298   Male
2     2     11   65  88.716 Female
3     3     27   80  70.178   Male
4     4     53   80  61.312   Male
5     5      4   40  89.522   Male
6     6     22   70  60.506 Female
7     7     16   20  81.462 Female
8     8     21   55  75.820 Female
9     9     25   50  69.372 Female
10   10     18   40  82.268 Female
```

The first issue we have is that some of the variables are not numeric (**Gender**) and others are not meaningful numerically (**code**). We have two choices here. The first is to make a new dataframe by selecting only the variables of interest) – we discovered how to do this in section 3.9.1. The second is to specify this subset within the *cor()* command itself. If we choose the first method then we should execute:

```
examData2 <- examData[, c("Exam", "Anxiety", "Revise")]
cor(examData2)
```

The first line creates a dataframe (*examData2*) that contains all of the cases, but only the variables **Exam**, **Anxiety** and **Revise**. The second command creates a table of Pearson correlations between these three variables (note that Pearson is the default so we don't need to specify it and because there are no missing cases we do not need the *use* command).

Alternatively, we could specify the subset of variables in the *examData* dataframe as part of the *cor()* function:

```
cor(examData[, c("Exam", "Anxiety", "Revise")])
```

The end result is the same, so it's purely down to preference. With the first method it is a little easier to see what's going on, but as you gain confidence and experience you might find that you prefer to save time and use the second method.

```
              Exam    Anxiety     Revise
Exam     1.0000000 -0.4409934  0.3967207
Anxiety -0.4409934  1.0000000 -0.7092493
Revise   0.3967207 -0.7092493  1.0000000
```

Output 6.1: Output for a Pearson's correlation

Output 6.1 provides a matrix of the correlation coefficients for the three variables. Each variable is perfectly correlated with itself (obviously) and so $r = 1$ along the diagonal of the table. Exam performance is negatively related to exam anxiety with a Pearson correlation coefficient of $r = -.441$. This is a reasonably big effect. Exam performance is positively related to the amount of time spent revising, with a coefficient of $r = .397$, which is also a reasonably big effect. Finally, exam anxiety appears to be negatively related to the time spent revising, $r = -.709$, which is a substantial effect size. In psychological terms, this all means that as anxiety about an exam increases, the percentage mark obtained in that exam decreases. Conversely, as the amount of time revising increases, the percentage obtained in the exam increases. Finally, as revision time increases, the student's anxiety about the exam decreases. So there is a complex interrelationship between the three variables.

Correlation coefficients are effect sizes, so we can interpret these values without really needing to worry about *p*-values (and as I have tried to drum into you, because *p*-values are related to sample size, there is a lot to be said for not obsessing about them). However, if you are the type of person who obsesses about *p*-values, then you can use the *rcorr()*

function instead and *p* yourself with excitement at the output it produces. First, make sure you have loaded the *Hmisc* package by executing:

```
library(Hmisc)
```

Next, we need to convert our dataframe into a matrix using the *as.matrix()* command. We can include only numeric variables so, just as we did above, we need to select only the numeric variables within the *examData* dataframe. To do this, execute:

```
examMatrix<-as.matrix(examData[, c("Exam", "Anxiety", "Revise")])
```

Which creates a matrix called *examMatrix* that contains only the variables **Exam**, **Anxiety**, and **Revise** from the *examData* dataframe. To get the correlation matrix we simply input this matrix into the *rcorr()* function:[4]

```
rcorr(examMatrix)
```

As before, I think that the method above makes it clear what we're doing, but more experienced users could combine the previous two commands into a single one:

```
rcorr(as.matrix(examData[, c("Exam", "Anxiety", "Revise")]))
```

Output 6.2 shows the same correlation matrix as Output 6.1, except rounded to 2 decimal places. In addition, we are given the sample size on which these correlations are based, and also a matrix of *p*-values that corresponds to the matrix of correlation coefficients above. Exam performance is negatively related to exam anxiety with a Pearson correlation coefficient of $r = -.44$ and the significance value is less than .001 (it is approximately zero). This significance value tells us that the probability of getting a correlation coefficient this big in a sample of 103 people if the null hypothesis were true (there was no relationship between these variables) is very low (close to zero in fact). Hence, we can gain confidence that there is a genuine relationship between exam performance and anxiety. Our criterion for significance is usually .05 (see section 2.6.1) so we can say that all of the correlation coefficients are significant.

```
          Exam Anxiety Revise
Exam      1.00   -0.44   0.40
Anxiety -0.44    1.00  -0.71
Revise    0.40   -0.71   1.00

n= 103


P
         Exam Anxiety Revise
Exam           0       0
Anxiety  0             0
Revise   0     0
```

Output 6.2

It can also be very useful to look at confidence intervals for correlation coefficients. Sadly, we have to do this one at a time (we can't do it for a whole dataframe or matrix). Let's look at the correlation between exam performance (**Exam**) and exam anxiety (**Anxiety**). We can compute the confidence interval *using cor.test()* by executing:

```
cor.test(examData$Anxiety, examData$Exam)
```

[4] The *ggm* package also has a function called *rcorr()*, so if you have this package installed, **R** might use that function instead, which will produce something very unpleasant on your screen. If so, you need to put *Hmisc::* in front of the commands to make sure **R** uses *rcorr()* from the *Hmisc* package (R's Souls' Tip 3.4):

```
Hmisc::rcorr(examMatrix)
Hmisc::rcorr(as.matrix(examData[, c("Exam", "Anxiety", "Revise")]))
```

Note that we have specified only the variables because by default this function produces Pearson's *r* and a 95% confidence interval. Output 6.3 shows the resulting output; it reiterates that the Pearson correlation between exam performance and anxiety was –.441, but tells us that this was highly significantly different from zero, $t(101) = -4.94$, $p < .001$. Most important, the 95% confidence ranged from –.585 to – .271, which does not cross zero. This tells us that in all likelihood, the population or actual value of the correlation is negative, so we can be pretty content that exam anxiety and exam performance are, in reality, negatively related.

```
        Pearson's product-moment correlation

data:  examData$Anxiety and examData$Exam
t = -4.938, df = 101, p-value = 3.128e-06
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 -0.5846244 -0.2705591
sample estimates:
       cor
-0.4409934
```

Output 6.3

SELF-TEST

✓ Compute the confidence intervals for the relationships between the time spent revising (**Revise**) and both exam performance (**Exam**) and exam anxiety (**Anxiety**).

6.5.4.3. Using R^2 for interpretation ①

Although we cannot make direct conclusions about causality from a correlation, we can take the correlation coefficient a step further by squaring it. The correlation coefficient squared (known as the **coefficient of determination**, R^2) is a measure of the amount of variability in one variable that is shared by the other. For example, we may look at the relationship between exam anxiety and exam performance. Exam performances vary from person to person because of any number of factors (different ability, different levels of preparation and so on). If we add up all of this variability (rather like when we calculated the sum of squares in section 2.4.1) then we would have an estimate of how much variability exists in exam performances. We can then use R^2 to tell us how much of this variability is shared by exam anxiety. These two variables had a correlation of −0.4410 and so the value of R^2 will be $(-0.4410)^2 = 0.194$. This value tells us how much of the variability in exam performance is shared by exam anxiety.

If we convert this value into a percentage (multiply by 100) we can say that exam anxiety shares 19.4% of the variability in exam performance. So, although exam anxiety was highly correlated with exam performance, it can account for only 19.4% of variation in exam scores. To put this value into perspective, this leaves 80.6% of the variability still to be accounted for by other variables.

You'll often see people write things about R^2 that imply causality: they might write 'the variance in *y accounted for* by *x*', or 'the variation in one variable *explained* by the other'. However, although R^2 is an extremely useful measure of the substantive importance of an effect, it cannot be used to infer causal relationships. Exam anxiety might well share 19.4% of the variation in exam scores, but it does not necessarily cause this variation.

We can get **R** to compute the coefficient of determination by remembering that "^2" means 'squared' in **R**-speak. Therefore, for our *examData2* dataframe (see earlier) if we execute:

```
cor(examData2)^2
```

instead of:

```
cor(examData2)
```

then you will see be a matrix containing r^2 instead of r (Output 6.4).

```
             Exam   Anxiety    Revise
Exam    1.0000000 0.1944752 0.1573873
Anxiety 0.1944752 1.0000000 0.5030345
Revise  0.1573873 0.5030345 1.0000000
```

Output 6.4

Note that for exam performance and anxiety the value is 0.194, which is what we calculated above. If you want these values expressed as a percentage then simply multiply by 100, so the command would become:

```
cor(examData2)^2 * 100
```

6.5.5. Spearman's correlation coefficient ①

Spearman's correlation coefficient (Spearman, 1910), r_s, is a non-parametric statistic and so can be used when the data have violated parametric assumptions such as non-normally distributed data (see Chapter 5). You'll sometimes hear the test referred to as Spearman's rho (pronounced 'row', as in 'row your boat gently down the stream'). Spearman's test works by first ranking the data (see section 15.4.1), and then applying Pearson's equation (equation (6.3)) to those ranks.

I was born in England, which has some bizarre traditions. One such oddity is the World's Biggest Liar competition held annually at the Santon Bridge Inn in Wasdale (in the Lake District). The contest honours a local publican, 'Auld Will Ritson', who in the nineteenth century was famous in the area for his far-fetched stories (one such tale being that Wasdale turnips were big enough to be hollowed out and used as garden sheds). Each year locals are encouraged to attempt to tell the biggest lie in the world (lawyers and politicians are apparently banned from the competition). Over the years there have been tales of mermaid farms, giant moles, and farting sheep blowing holes in the ozone layer. (I am thinking of entering next year and reading out some sections of this book.)

Imagine I wanted to test a theory that more creative people will be able to create taller tales. I gathered together 68 past contestants from this competition and asked them where they were placed in the competition (first, second, third, etc.) and also gave them a creativity questionnaire (maximum score 60). The position in the competition is an ordinal variable (see section 1.5.1.2) because the places are categories but have a meaningful order (first place is better than second place and so on). Therefore, Spearman's correlation coefficient should be used (Pearson's r requires interval or ratio data). The data for this study are in the file **The Biggest Liar.dat**. The data are in two columns: one labelled **Creativity** and one labelled **Position** (there's actually a third variable in there but we will ignore it for the time being). For the **Position** variable, each of the categories described above has been coded with a numerical value. First place has been coded with the value 1, with positions being labelled 2, 3 and so on.

The procedure for doing a Spearman correlation is the same as for a Pearson correlation except that we need to specify that we want a Spearman correlation instead of Pearson,

which is done using *method* = *"spearman"* for *cor()* and *cor.test()*, and *type* = *"spearman"* for *rcorr()*. Let's load the data into a dataframe and then create a dataframe by executing:

```
liarData = read.delim("The Biggest Liar.dat",  header = TRUE)
```

or if you haven't set your working directory, execute this command and use the dialog box to select the file:

```
liarData = read.delim(file.choose(),  header = TRUE)
```

SELF-TEST

✓ See whether you can use what you have learned so far to compute a Spearman's correlation between **Position** and **Creativity**.

To obtain the correlation coefficient for a pair of variables we can execute:

```
cor(liarData$Position, liarData$Creativity, method = "spearman")
```

Note that we have simply specified the two variables of interest, and then set the method to be a Spearman correlation. The output of this command will be:

```
[1] -0.3732184
```

If we want a significance value for this correlation we could either use *rcorr()* by executing (remembering that we have to first convert the dataframe to a matrix):

```
liarMatrix<-as.matrix(liarData[, c("Position", "Creativity")])
rcorr(liarMatrix)
```

or simply use *cor.test()*, which has the advantage that we can set a directional hypothesis. I predicted that more creative people would tell better lies. Doing well in the competition (i.e., telling better lies) actually equates to a lower number for the variable **Position** (first place = 1, second place = 2 etc.), so we're predicting a negative relationship. High scores on **Creativity** should equate to a lower value of **Position** (because a low value means you did well!). Therefore, we predict that the correlation will be less than zero, and we can reflect this prediction by using *alternative* = *"less"* in the command:

```
cor.test(liarData$Position,  liarData$Creativity,  alternative  =  "less",
method = "spearman")
```

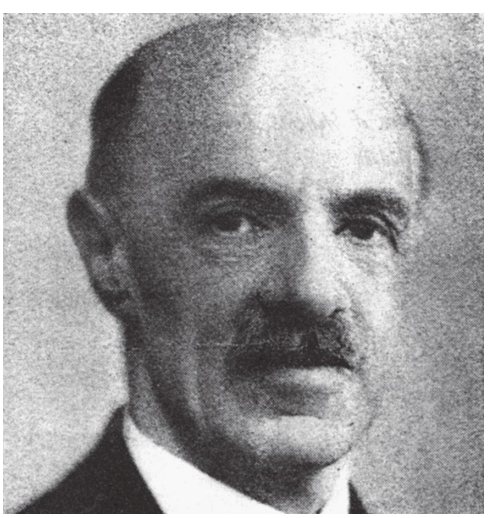

FIGURE 6.6 Charles Spearman, ranking furiously

```
        Spearman's rank correlation rho
data:  liarData$Position and liarData$Creativity
S = 71948.4, p-value = 0.0008602
alternative hypothesis: true rho is less than 0
sample estimates:
       rho
-0.3732184
```

Output 6.5

Output 6.5 shows the output for a Spearman correlation on the variables **Creativity** and **Position**. The output is very similar to that of the Pearson correlation (except that confidence intervals are not produced – if you want one see the section on bootstrapping): the correlation coefficient between the two variables is fairly large (−.373), and the significance value of this coefficient is very small ($p < .001$). The significance value for this correlation coefficient is less than .05; therefore, it can be concluded that there is a significant relationship between creativity scores and how well someone did in the World's Biggest Liar competition. Note that the relationship is negative: as creativity increased, position decreased. Remember that a low number means that you did well in the competition (a low number such as 1 means you came first, and a high number like 4 means you came fourth). Therefore, our hypothesis is supported: as creativity increased, so did success in the competition.

SELF-TEST

✓ Did creativity cause success in the World's Biggest Liar competition?

6.5.6. Kendall's tau (non-parametric) ①

Kendall's tau, τ, is another non-parametric correlation and it should be used rather than Spearman's coefficient when you have a small data set with a large number of tied ranks. This means that if you rank all of the scores and many scores have the same rank, then Kendall's tau should be used. Although Spearman's statistic is the more popular of the two coefficients, there is much to suggest that Kendall's statistic is actually a better estimate of the correlation in the population (see Howell, 1997: 293). As such, we can draw more accurate generalizations from Kendall's statistic than from Spearman's. To carry out Kendall's correlation on the World's Biggest Liar data simply follow the same steps as for Pearson and Spearman correlations but use *method* = *"kendall"*:

```
cor(liarData$Position, liarData$Creativity, method = "kendall")
```

```
cor.test(liarData$Position, liarData$Creativity, alternative = "less",
method = "kendall")
```

The output is much the same as for Spearman's correlation.

```
        Kendall's rank correlation tau
data:  liarData$Position and liarData$Creativity
z = -3.2252, p-value = 0.0006294
alternative hypothesis: true tau is less than 0
sample estimates:
       tau
-0.3002413
```

Output 6.6

You'll notice from Output 6.6 that the actual value of the correlation coefficient is closer to zero than the Spearman correlation (it has increased from −.373 to −.300). Despite the difference in the correlation coefficients we can still interpret this result as being a highly significant relationship (because the significance value of .001 is less than .05). However, Kendall's value is a more accurate gauge of what the correlation in the population would be. As with the Pearson correlation, we cannot assume that creativity caused success in the World's Best Liar competition.

SELF-TEST

✓ Conduct a Pearson correlation analysis of the advert data from the beginning of the chapter.

6.5.7. Bootstrapping correlations ③

Another way to deal with data that do not meet the assumptions of Pearson's *r* is to use bootstrapping. The *boot()* function takes the general form:

```
object<-boot(data, function, replications)
```

in which *data* specifies the dataframe to be used, *function* is a function that you write to tell *boot()* what you want to bootstrap, and *replications* is a number specifying how many bootstrap samples you want to take (I usually set this value to 2000). Executing this command creates an *object* that has various properties. We can view an estimate of bias, and an empirically derived standard error by viewing *object*, and we can display confidence intervals based on the bootstrap by executing *boot.ci(object)*.

When using the *boot()* function with correlations (and anything else for that matter) the tricky bit is writing the function (R's Souls' Tip 6.2). If we stick with our biggest liar data and want to bootstrap Kendall tau, then our function will be:

```
bootTau<-function(liarData,i)cor(liarData$Position[i],liarData$Creativity[i],
use = "complete.obs", method = "kendall")
```

Executing this command creates an object called *bootTau*. The first bit of the function tells **R** what input to expect in the function: in this case we need to feed a dataframe (*liarData*) into the function and a variable that has been called *i* (which refers to a particular bootstrap sample). The second part of the function specifies the *cor()* function, which is the thing we want to bootstrap. Notice that *cor()* is specified in exactly the same way as when we did the original Kendall correlation except that for each variable we have added *[i]*, which again just refers to a particular bootstrap sample. If you want to bootstrap a Pearson or Spearman correlation you do it in exactly the same way except that you specify *method = "pearson"* or *method = "spearman"* when you define the function.

To create the bootstrap object, we execute:

```
library(boot)
boot_kendall<-boot(liarData, bootTau, 2000)
boot_kendall
```

The first command loads the *boot* package (in case you haven't already initiated it). The second command creates an object (*boot_kendall*) based on bootstrapping the *liarData* dataframe using the *bootTau* function that we previously defined and executed. The second

line displays a summary of the *boot_kendall* object. To get the 95% confidence interval for the *boot_kendall* object we execute:[5]

```
boot.ci(boot_kendall)
```

Output 6.7 shows the contents of both *boot_kendall* and also the output of the *boot.ci()* function. First, we get the original value of Kendall's tau (–.300), which we computed in the previous section. We also get an estimate of the bias in that value (which in this case is very small) and the standard error (0.098) based on the bootstrap samples. The output from *boot.ci()* gives us four different confidence intervals (the basic bootstrapped CI, percentile and BCa). The good news is that none of these confidence intervals cross zero, which gives us good reason to think that the population value of this relationship between creativity and success at being a liar is in the same direction as the sample value. In other words, our original conclusions stand.

```
ORDINARY NONPARAMETRIC BOOTSTRAP

Call:
boot(data = liarData, statistic = bootTau, R = 2000)

Bootstrap Statistics :
      original      bias    std. error
t1* -0.3002413 0.001058191    0.097663


> boot.ci(boot_kendall)

BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
Based on 2000 bootstrap replicates

CALL :
boot.ci(boot.out = boot_kendall)

Intervals :
Level      Normal              Basic
95%   (-0.4927, -0.1099 )   (-0.4956, -0.1126 )

Level     Percentile            BCa
95%   (-0.4879, -0.1049 )   (-0.4777, -0.0941 )
Calculations and Intervals on Original Scale
Warning message:
In boot.ci(boot_kendall) :
  bootstrap variances needed for studentized intervals
```

Output 6.7

SELF-TEST

✓ Conduct bootstrap analysis of the Pearson and Spearman correlations for the *examData2* dataframe.

[5] If we want something other than a 95% confidence interval we can add *conf = x*, in which *x* is the value of the confidence interval as a proportion. For example, we can get a 99% confidence interval by executing:

```
boot.ci(boot_kendall, conf = 0.99)
```