Correlation

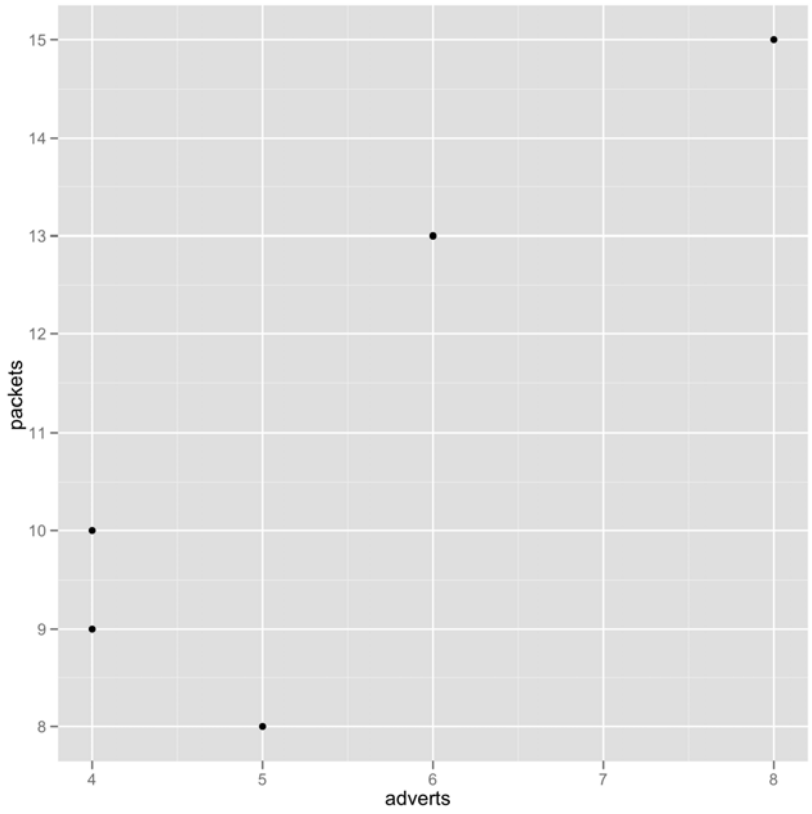
Self-test answers

- Enter the advert data and use *ggplot2* to produce a scatterplot (number of packets bought on the *y*-axis, and adverts watched on the *x*-axis) of the data.

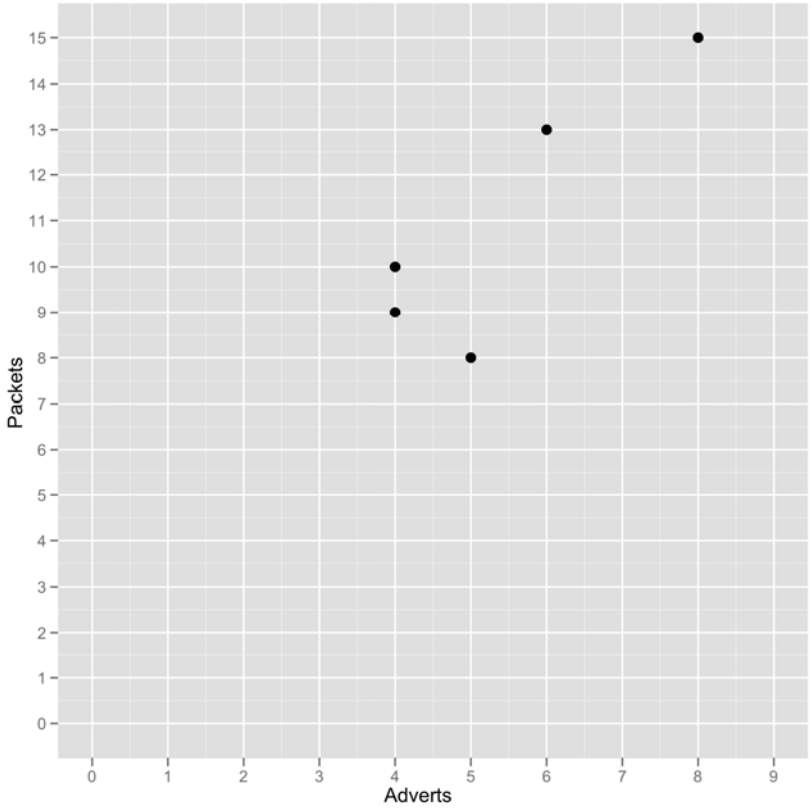
You can get a basic scatterplot by executing the following commands:

```
scatter<-ggplot(advertData, aes(adverts, packets))
scatter + geom_point(size = 3) + labs(x = "Adverts", y = "Packets")
```

Your scatterplot should come out like this:



This graph looks horrible because *ggplot2* has not scaled the axes from 0. If yours looks like this too, then, as an additional task, edit it so that the axes both start at 0. While you're at it, label the axes nicely and make the points larger. Mine ended up like this:



Ah, that's better. This is the code that I used:

```
scatter<-ggplot(advertData, aes(adverts, packets))

scatter + geom_point(size = 3) + labs(x = "Adverts", y = "Packets") +
scale_y_continuous(limits=c(0, 15), breaks=0:15) + scale_x_continuous(limits=c(0, 9),
breaks=0:9)
```

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

Assuming you have set your working directory to be the location of the file, you can load the data by executing this command:

```
examData = read.delim("Exam Anxiety.dat",  header = TRUE)
```

Alternatively, if you want to select the file using a dialog box you could execute:

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

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

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

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

The outputs will be:

```
> cor.test(examData$Revise, examData$Exam)

        Pearson's product-moment correlation

data:  examData$Revise and examData$Exam
t = 4.3434, df = 101, p-value = 3.343e-05
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
```

```
 0.2200938 0.5481602
sample estimates:
      cor
0.3967207

> cor.test(examData$Anxiety, examData$Revise)

        Pearson's product-moment correlation

data:  examData$Anxiety and examData$Revise
t = -10.1111, df = 101, p-value < 2.2e-16
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 -0.7938168 -0.5977733
sample estimates:
       cor
-0.7092493
```

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

No it didn't. Well, it might have done, but we can't tell this from a correlation coefficient. This is because although we found a significant relationship between creativity and position, we cannot infer causality from a correlation coefficient.

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

Execute these commands:

```
adverts<-c(5,4,4,6,8)

packets<-c(8,9,10,13,15)

cor.test(adverts, packets)
```

The output will be:

```
        Pearson's product-moment correlation

data:  adverts and packets
t = 3.0732, df = 3, p-value = 0.05443
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 -0.0479747  0.9914236
sample estimates:
     cor
0.871165
```

This value is the same as that calculated in the book.

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

Here are the code and output for the Pearson correlation between exam performance and exam anxiety:

```
bootR<-function(examData2,i) cor(examData2$Exam[i], examData2$Anxiety[i], use =
"complete.obs")

boot_pearson<-boot(examData2, bootR, 2000)

boot_pearson

boot.ci(boot_pearson)
```

```
ORDINARY NONPARAMETRIC BOOTSTRAP

Call:
boot(data = examData2, statistic = bootR, R = 2000)
```

```
Bootstrap Statistics :
      original      bias    std. error
t1* -0.4409934 0.002317646  0.06352735

> boot.ci(boot_pearson)

BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
Based on 2000 bootstrap replicates

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

Intervals :
Level      Normal              Basic
95%   (-0.5678, -0.3188 )   (-0.5752, -0.3288 )

Level     Percentile            BCa
95%   (-0.5532, -0.3068 )   (-0.5582, -0.3120 )
Calculations and Intervals on Original Scale
```

Here are the code and output for the Pearson correlation between exam revision and exam anxiety:

```
bootR<-function(examData2,i) cor(examData2$Revise[i], examData2$Anxiety[i], use =
"complete.obs")

boot_pearson<-boot(examData2, bootR, 2000)

boot_pearson

boot.ci(boot_pearson)


ORDINARY NONPARAMETRIC BOOTSTRAP

Call:
boot(data = examData2, statistic = bootR, R = 2000)

Bootstrap Statistics :
      original      bias    std. error
t1* -0.7092493 0.005252507   0.1126909

> boot.ci(boot_pearson)

BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
Based on 2000 bootstrap replicates

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

Intervals :
Level      Normal              Basic
95%   (-0.9354, -0.4936 )   (-0.9773, -0.5467 )

Level     Percentile            BCa
95%   (-0.8718, -0.4412 )   (-0.8519, -0.3415 )
Calculations and Intervals on Original Scale
```

Here are the code and output for the Pearson correlation between exam performance and exam revision:

```
bootR<-function(examData2,i) cor(examData2$Revise[i], examData2$Exam[i], use =
"complete.obs")

boot_pearson<-boot(examData2, bootR, 2000)

boot_pearson

boot.ci(boot_pearson)

ORDINARY NONPARAMETRIC BOOTSTRAP

Call:
boot(data = examData2, statistic = bootR, R = 2000)


Bootstrap Statistics :
     original       bias    std. error
t1* 0.3967207 -0.004676362  0.06795331

> boot.ci(boot_pearson)

BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
Based on 2000 bootstrap replicates
```

```
CALL :
boot.ci(boot.out = boot_pearson)

Intervals :
Level      Normal              Basic
95%   ( 0.2682,  0.5346 )   ( 0.2749,  0.5404 )

Level     Percentile            BCa
95%   ( 0.2530,  0.5186 )   ( 0.2596,  0.5227 )
Calculations and Intervals on Original Scale
```

Here are the code and output for the Spearman correlation between exam performance and exam anxiety:

```
bootRho<-function(examData2,i) cor(examData2$Exam[i], examData2$Anxiety[i], use =
"complete.obs", method = "spearman")

boot_spearman<-boot(examData2, bootRho, 2000)

boot_spearman

boot.ci(boot_spearman)


ORDINARY NONPARAMETRIC BOOTSTRAP

Call:
boot(data = examData2, statistic = bootRho, R = 2000)

Bootstrap Statistics :
      original      bias    std. error
t1* -0.4046141 0.002405364  0.08018193

> boot.ci(boot_spearman)

BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
Based on 2000 bootstrap replicates

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

Intervals :
Level      Normal              Basic
95%   (-0.5642, -0.2499 )   (-0.5717, -0.2554 )

Level     Percentile            BCa
95%   (-0.5538, -0.2375 )   (-0.5523, -0.2340 )
Calculations and Intervals on Original Scale
```

Here are the code and output for the Spearman correlation between exam revision and exam anxiety:

```
bootRho<-function(examData2,i) cor(examData2$Revise[i], examData2$Anxiety[i], use =
"complete.obs", method = "spearman")

boot_spearman <-boot(examData2, bootRho, 2000)

boot_spearman

boot.ci(boot_spearman)


ORDINARY NONPARAMETRIC BOOTSTRAP

Call:
boot(data = examData2, statistic = bootRho, R = 2000)


Bootstrap Statistics :
      original      bias    std. error
t1* -0.6219694 0.003764904   0.0819895

> boot.ci(boot_spearman)

BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
Based on 2000 bootstrap replicates

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

Intervals :
Level      Normal              Basic
95%   (-0.7864, -0.4650 )   (-0.7982, -0.4773 )
```

```
Level     Percentile            BCa
95%   (-0.7666, -0.4458 )   (-0.7588, -0.4319 )
Calculations and Intervals on Original Scale
```

Here are the code and output for the Spearman correlation between exam performance and exam revision:

```
bootRho<-function(examData2,i) cor(examData2$Revise[i], examData2$Exam[i], use =
"complete.obs", method = "spearman")

boot_spearman <-boot(examData2, bootRho, 2000)

boot_spearman

boot.ci(boot_spearman)
```

```
ORDINARY NONPARAMETRIC BOOTSTRAP


Call:
boot(data = examData2, statistic = bootRho, R = 2000)


Bootstrap Statistics :
     original       bias    std. error
t1* 0.3498948 -0.003656983  0.09103305
> boot.ci(boot_spearman)
BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
Based on 2000 bootstrap replicates

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

Intervals :
Level      Normal              Basic
95%   ( 0.1751,  0.5320 )   ( 0.1865,  0.5423 )

Level     Percentile            BCa
95%   ( 0.1575,  0.5133 )   ( 0.1490,  0.5097 )
```

- Carry out a Pearson correlation on **time** and **gender**.

```
cor.test(catData$time, catData$gender)
```

The output is shown in the chapter.

- Carry out a Pearson correlation on **time** and **recode**.

```
cor.test(catData$time, catData$recode)
```

The output is:

```
        Pearson's product-moment correlation

data:  catData$time and catData$recode
t = -3.1138, df = 58, p-value = 0.002868
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 -0.576936 -0.137769
sample estimates:
       cor
-0.3784542
```

- Use the *subset()* function to compute the correlation coefficient between exam anxiety and exam performance in men and women.

To create separate dataframes for males and females we can use the subset command as follows:

```
maleExam<-subset(examData, Gender == "Male", select= c("Exam", "Anxiety"))
femaleExam<-subset(examData, Gender == "Female", select= c("Exam", "Anxiety"))
```

In both cases we have created new dataframes that contain only male data (*maleExam*) and female data (*femaleExam*). We have done this by taking the *examData* dataframe and selecting cases based on the logical operator *Gender == "Male"* or *Gender == "Female"*; note that **Gender** is a string variable, containing the words 'Male' and 'Female' (remember the capital letter), which is why we have specified the condition in the way we have. Finally, we use the select command to extract the numeric variables of interest (i.e., **Exam** and **Anxiety**). These dataframes can then be fed into the *cor()* function to get the correlation coefficients:

```
cor(maleExam)
cor(femaleExam)
```

Execute these commands, and the output for males will look like this:

```
              Exam    Anxiety
Exam     1.0000000 -0.5056874
Anxiety -0.5056874  1.0000000
```

For females, the output is as follows:

```
              Exam    Anxiety
Exam     1.0000000 -0.3813845
Anxiety -0.3813845  1.0000000
```

The book chapter has some interpretation of these findings and suggestions for how to compare the coefficients for males and females.

Oliver Twisted

Please Sir, can I have some more … covariance and variance?

There are two functions very similar to *cor()* that can be used for computing variance and covariances; these are *var()* and *cov()*. These variables take the following general form:

```
var(x, y, na.rm = FALSE, use = "string")
cov(x, y, use = "string", method = c("pearson", "kendall", "spearman"))
```

We use both these functions much as we do *cor()*.

- *x*: This is a numeric variable or dataframe.
- *y*: This is another numeric variable (does not need to be specified if *x*, above, is a dataframe).
- *use*: This is a character string that specifies how missing values are handled. The strings can be: (1) *"everything"*, which will mean that **R** will output an NA instead of a correlation coefficient for any correlations involving variables containing missing values; (2) *"all.obs"*, which will use all observations and, therefore, returns an error message if there are any missing values in the data; (3) *"complete.obs"*, in which correlations are computed from only cases that are complete for all variables; or (4) *"pairwise.complete.obs"*, in which correlations between pairs of variables are computed for cases that are complete for those two variables.
- *method*: This enables you to specify whether you want *"pearson"*, *"spearman"* or *"kendall"* covariances. If you want more than one type you can specify a list using the *c()* function; for example, *c("pearson", "spearman")* would produce both types of covariances.
- *na.rm*: This determines whether missing values should be removed (TRUE) or if they should be included (FALSE), the default.

Let's input the advert data from the book chapter by executing the following commands:

```
adverts<-c(5,4,4,6,8)
packets<-c(8,9,10,13,15)
advertData<-data.frame(adverts, packets)
```

If we want to compute the covariance between these two variables we execute:

```
cov(adverts, packets)
```

which gives us the following output:

```
[1] 4.25
```

This is the value calculated in equation (6.2).

If we want to compute the variance of a variable we execute:

```
var(adverts)
```

which gives us the following output:

```
[1] 2.8
```

This value could be calculated manually from equation (6.1).

If you want the covariances and variances to be computed at the same time, then enter the dataframe into *cov()*:

```
cov(advertData)
```

This gives the following output:

```
         adverts packets
adverts     2.80    4.25
packets     4.25    8.50
```

The value of the covariance between the two variables is 4.25, which is the same value as was calculated from equation (6.2). The covariance within each variable is the same as the variance for each variable (so the variance for the number of adverts seen is 2.8, and the variance for the number of packets bought is 8.5). These variances can be calculated manually from equation (6.1).

Please Sir, can I have some more … functions?

If you want to compare two independent *r*s (i.e. *r*s measuring the same thing in two different samples) then probably the best way to do it is to write yourself a function that describes equation (6.11) in the book. The reason why writing a function is a good idea is that, having executed the function once, you can use it over and over again within your current **R** session. This saves you having to write out the equation every time you want to compare two correlations.

If you access the **DSUR Correlations.R** file from the companion website you will see that I have written the function or you. The function looks like this:

```
zdifference<-function(r1, r2, n1, n2)
{zd<-(atanh(r1)-atanh(r2))/sqrt(1/(n1-3)+1/(n2-3))
        p <-1 - pnorm(abs(zd))
        print(paste("Z Difference: ", zd))
        print(paste("One-Tailed P-Value: ", p))
        }
```

This looks pretty horrendous, but before you run off and find a pencil with which to stab yourself in the head we'll try to break it down into its constituent parts:

- *zdifference*: This is the name I have given to the function. If we want to use it in the future, we therefore would type *zdifference()* just like we do with built in functions.

- *function(r1, r2, n1, n2)*: This defines what values are *input* to the function. To calculate the difference between correlations we need to know four things: the correlations (which I have called *r1* and *r2*, but I could have given then any label I liked), and the sample sizes on which they are based (which I have called *n1* and *n2*). When we use the function we will have to input these four values. So, to calculate the difference for the example in the book we would simply execute *zdifference(-0.506, -0.381, 52, 51)*.
- *{}*: Anything between these brackets is the function itself. It contains a list of instructions that tell **R** what to do.
- *zd<-(atanh(r1)-atanh(r2))/sqrt(1/(n1-3)+1/(n2-3))*: The first instruction is to create an object called *zd*, which is simply equation (6.11) from the book. We use a built-in function, *atanh()*, to get the *z* of each correlation coefficient. The rest is simply inputting the sample sizes (*n1* and *n2*) into the equation.
- *p <-1 - pnorm(abs(zd))*: This creates an object, *p*, which is the one-tailed probability of *zd* (which we have already computed). We obtain this using a built in function *pnorm()*, which is applied to the absolute value of *zd* (i.e., ignore the plus or minus sign); note that we use the function *abs()* to get the absolute value of *zd*.
- *print(paste("Z Difference: ", zd))*: This command prints *zd* (the test statistic for the difference between the correlations). To make the output nicer we use the *paste()* command to 'paste' together a text string explaining what the statistic is, and the value of the statistic itself.
- *print(paste("One-Tailed P-Value: ", p))*: This command is the same as above but prints the value of *p* to the screen with an explanation of what the value is.

Once you have executed the above function you can compute differences between correlations by simply referring to *zdifference()* (remembering to include the necessary information for *r1*, *r2*, *n1* and *n2*). For example, to test the difference between the correlation for exam anxiety and performance in males and females (see the book chapter), we would execute:

```
zdifference(-0.506, -0.381, 52, 51)
tdifference(-0.441, -0.709, 0.397, 103)
```

The output will look like this:

```
[1] "Z Difference:  -0.768709306290097"
[1] "One-Tailed P-Value:  0.221032949510287"
```

These values match those computed in the chapter.

Please Sir, can I have some more … comparing of correlations?

If you want to compare dependent *r*s then writing a function is again the way to go. And again, I have written it for you in the file **DSUR Correlations.R**. The function looks like this:

```
tdifference<-function(rxy, rxz, rzy, n)
{       df<-n-3
td<-(rxy-rzy)*sqrt((df*(1 + rxz))/(2*(1-rxy^2-rxz^2-rzy^2+(2*rxy*rxz*rzy))))
p <-pt(td, df)
print(paste("t Difference: ", td))
print(paste("One-Tailed P-Value: ", p))
}
```

As before, let's break this function down:

- *tdifference*: This is the name I have given to the function. If we want to use it in the future, we therefore would type *tdifference()* just like we do with built-in functions.
- *function(rxy, rxz, rzy, n)*: This defines what values are *input* to the function. To calculate the difference between dependent correlations we need to know four things: the correlations between all combinations of variables (which I have called *rxy*, *rxz*, *rzy* to match the equation in the book), and the sample size on which they are based (which I have called *n*). When we use the function we will have to input these four values. So, to calculate the difference for the example in the book we would simply execute *tdifference(-0.441, -0.709, 0.397, 103)*.
- *{}*: Anything between these brackets is the function itself. It contains a list of instructions that tell **R** what to do.
- *df<-n-3*: This computes an object called df (degrees of freedom) which is the sample size minus 3. I have computed this value because N-3 gets used a few times for various things.
- *td<-(rxy-rzy)*sqrt((df*(1 + rxz))/(2*(1-rxy^2-rxz^2-rzy^2+(2*rxy*rxz*rzy))))*: The second instruction creates an object called *td*, which is simply equation (6.12) from the book. Note that this command basically takes the values that we input into the equation and places them in equation (6.12) to compute the *t*-value for the difference between correlations.
- *p <-pt(td, df)*: This creates an object, *p*, which is the one-tailed probability of *td* (which we have already computed). We obtain this using a built in function *pt()*, which is applied to *td* with its associated degrees of freedom, which we computed earlier (*df*).
- *print(paste("t Difference: ", td))*: This command prints *td* (the test statistic for the difference between the correlations). To make the output nicer we use the *paste()* command to 'paste' together a text string explaining what the statistic is, and the value of the statistic itself.
- *print(paste("One-Tailed P-Value: ", p))*: This command is the same as above but prints the value of *p* to the screen with an explanation of what the value is.

Once you have executed the above function you can compute differences between dependent correlations by simply referring to *tdifference()* (remembering to include the necessary information for *rxy*, *rxz*, *rzy* and *n*). For example, to test the difference between the correlation for exam anxiety and performance in males and females (see the book chapter), we would execute:

```
tdifference(-0.441, -0.709, 0.397, 103)
```

The output will look like this:

```
[1] "t Difference:  -5.09576822523987"
[1] "One-Tailed P-Value:  8.21913727738006e-07"
```

The value of *t* corresponds to the value calculated in the chapter.

Labcoat Leni's real research

Why do you like your lecturers?

Problem

Chamorro-Premuzic, T., et al. (2008). *Personality and Individual Differences*, *44,* 965–976.

As students you probably have to rate your lecturers at the end of the course. There will be some lecturers you like and others that you hate. As a lecturer I find this process horribly depressing (although this has a lot to do with the fact that I tend focus on negative feedback and ignore the good stuff). There is some evidence that students tend to pick courses of lecturers whom they perceive to be enthusastic and good communicators. In a fascinating study, Tomas Chamorro-Premuzic and his colleagues (Chamorro-Premuzic, Furnham, Christopher, Garwood, & Martin, 2008) tested a

slightly different hypothesis, which was that students tend to like lecturers who are like themselves. (This hypothesis will have the students on my course who like my lectures screaming in horror.)

First of all, the authors measured students' own personalities using a very well-established measure (the NEO-FFI) which gives rise to scores on five fundamental personality traits: Neuroticism, Extroversion, Openness to experience, Agreeableness and Conscientiousness. They also gave students a questionnaire that asked them to have each of a list of characteristics. For example, they would be given the description 'warm: friendly, warm, sociable, cheerful, affectionate, outgoing' and asked to rate how much they wanted to see this in a lecturer from −5 (they don't want this characteristic at all) through 0 (the characteristic is not important) to +5 (I really want this characteristic in my lecturer). The characteristics on the questionnaire all related personality characteristics measured by the NEO-FFI. As such, the authors had a measure of how much a student had each of the five core personality characteristics, but also a measure of how much they wanted to see those same characteristics in their lecturer.

In doing so, Tomas and his colleagues could test whether, for instance, extroverted students want extrovert lecturers. The data from this study (well, for the variables that I've mentioned) are in the file **Chamorro-Premuzic.dat**. Run some Pearson correlations on these variables to see if students with certain personality characteristics want to see those characteristics in their lecturers. What conclusions can you draw?

Solution

We can run this analysis by loading the data:

```
personalityData = read.delim("Chamorro-Premuzic.dat",  header = TRUE)
```

and then just pretty much running a Pearson correlation on all the variables. I am going to use the *rcorr()* function because it is able to calculate correlation coefficients for multiple pairs of variables and also their corresponding *p*-values.

Remember that to use this function you will need the *Hmisc* package, so (if you haven't already done so) execute:

```
install.packages("Hmisc")
library(Hmisc)
```

The annoying thing about *rcorr()* is that it does not work on dataframes, so we will have to convert our *personalityData* dataframe into a matrix before we can run the correlation analysis. This can be done by excecuting the following command:

```
personalityMatrix<-as.matrix(personalityData[, c("studentN", "studentE", "studentO",
"studentA", "studentC", "lecturerN", "lecturerE", "lecturerO", "lecturerA",
"lecturerC")])
```

We can then run the correlation analysis by executing:

```
rcorr(personalityMatrix)
```

Or you could be a real smarty-pants and combine the above two steps:

```
rcorr(as.matrix(personalityData[, c("studentN", "studentE", "studentO", "studentA",
"studentC", "lecturerN", "lecturerE", "lecturerO", "lecturerA", "lecturerC")]))
```

In the output you will get three matrices. The first matrix displays the Pearson's correlation coefficients, the second displays the sample sizes, and the final matrix displays the *p*-values of the correlation coefficients. I will just include the first and last matrices here, which should look like this:

```
           studentN studentE studentO studentA studentC lecturerN lecturerE lecturerO lecturerA lecturerC
studentN       1.00    -0.34    -0.06     0.01    -0.20     0.01    -0.08    -0.02     0.10     0.00
studentE      -0.34     1.00     0.07     0.08     0.19    -0.10     0.15     0.07     0.00    -0.01
studentO      -0.06     0.07     1.00    -0.04    -0.09    -0.10     0.04     0.20    -0.16    -0.03
studentA       0.01     0.08    -0.04     1.00     0.52    -0.02     0.05     0.11     0.16     0.20
studentC      -0.20     0.19    -0.09     0.52     1.00    -0.14     0.10     0.03     0.13     0.22
lecturerN      0.01    -0.10    -0.10    -0.02    -0.14     1.00     0.00     0.04     0.04     0.26
lecturerE     -0.08     0.15     0.04     0.05     0.10     0.00     1.00     0.49     0.12     0.10
lecturerO     -0.02     0.07     0.20     0.11     0.03     0.04     0.49     1.00     0.24     0.12
lecturerA      0.10     0.00    -0.16     0.16     0.13     0.04     0.12     0.24     1.00     0.24
lecturerC      0.00    -0.01    -0.03     0.20     0.22    -0.26     0.10     0.12     0.24     1.00
```

```
P
          studentN studentE studentO studentA studentC   lecturerN lecturerE lecturerO lecturerA lecturerC
studentN            0.0000   0.2586   0.8961   0.0000    0.8945    0.1756    0.7123    0.0410    0.9560
studentE  0.0000             0.1599   0.1056   0.0001    0.0448    0.0102    0.1654    0.9315    0.8442
studentO  0.2586   0.1599             0.4613   0.0655    0.0406    0.4989    0.0000    0.0009    0.4942
studentA  0.8961   0.1056   0.4613             0.0000    0.6671    0.4118    0.0315    0.0009    0.0000
studentC  0.0000   0.0001   0.0655   0.0000              0.0046    0.0900    0.5823    0.0072    0.0000
lecturerN 0.8945   0.0448   0.0406   0.6671   0.0046              0.9745    0.4561    0.3612    0.0000
lecturerE 0.1756   0.0102   0.4989   0.4118   0.0900    0.9745              0.0000    0.0491    0.0925
lecturerO 0.7123   0.1654   0.0000   0.0315   0.5823    0.4561    0.0000              0.0000    0.0144
lecturerA 0.0410   0.9315   0.0009   0.0009   0.0072    0.3612    0.0491    0.0000              0.0000
lecturerC 0.9560   0.8442   0.4942   0.0000   0.0000    0.0000    0.0925    0.0144    0.0000
```

These tables look pretty horrendous, but there are actually a lot of correlations that we don't need. First, the table is symmetrical around the diagonal, so we can first ignore either the top diagonal or the bottom (the values are the same). The second thing is that we're interested only in the correlations between students' personalities and what they want in lecturers. We're not interested in how their own five personality traits correlated with each other (i.e. if a student is neurotic are they conscientious too?) The red rectangles above show the areas of the table that contain the information that we need to address our hypothesis. The top right red rectangle is a replica of the values reported in the original research paper (part of the authors' table is below so you can see how they reported these values – match these values to the values in your output):

		Lecturer				
		N	E	O	A	C
Student	N	.00	−.08	−.02	.10*	.00
	E	−.10*	.15*	.07	.00	−.01
	O	−.10*	.04	.20**	−.17**	.03
	A	−.02	.05	.11**	.16**	.20**
	C	−.14**	.10*	.03	.13	.22**

As for what we can conclude, well, neurotic students tend to want agreeable lecturers, $r = .10$, $p < .05$; extroverted students tend to want extroverted lecturers, $r = .15$, $p < .05$; students who are open to experiences tend to want lecturers who are open to experience, $r = .20$, $p < .01$, and don't want agreeable lecturers, $r = -.17$, $p < .01$; agreeable students want every sort of lecturer apart from neurotic and extroverted. This could be because these students are agreeable! Finally, conscientious students tend to want conscientious lecturers, $r = .22$, $p < .01$, and extroverted ones, $r = .10$, $p < .05$, but don't want neurotic ones, $r = -.14$, $p < .01$.

Smart Alex's solutions

Task 1

- A student was interested in whether there was a positive relationship between the time spent doing an essay and the mark received. He got 45 of his friends and timed how long they spent writing an essay (**hours**) and the percentage they got in the essay (**essay**). He also translated these grades into their degree classifications (**grade**): in the UK, a student can get a first-class mark (the best), an upper second, a lower second, a third, a pass or a fail (the worst). Using the data in the file **EssayMarks.dat** find out what the relationship was between the time spent doing an essay and the eventual mark in terms of percentage and degree class (draw a scatterplot too!).

We're interested in looking at the relationship between hours spent on an essay and the grade obtained. We could simply do a scatterplot of hours spent on the essay (*x*-axis) and essay mark (*y*-axis).

First of all, load the data:

```
essayData = read.delim("EssayMarks.dat",  header = TRUE)
```

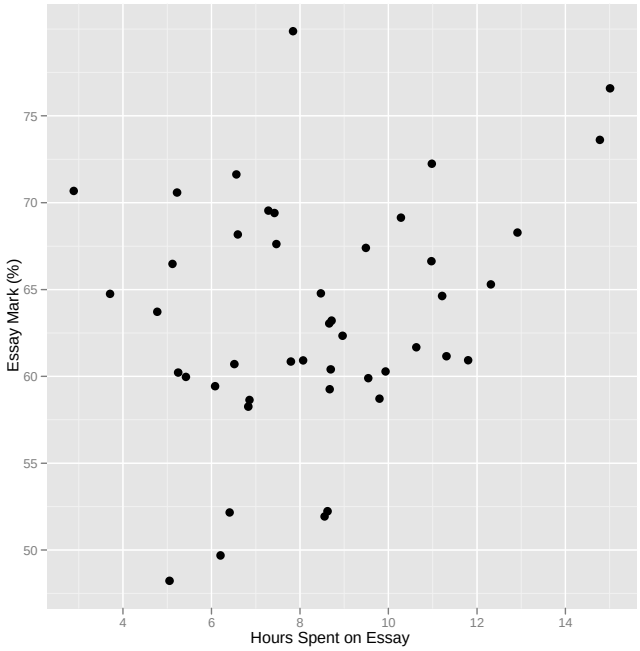
Then create a plot object called scatter:

```
scatter<-ggplot(essayData, aes(hours, essay))
```

Next add labels to your scatterplot:

```
scatter + geom_point(size = 3) + labs(x = "Hours Spent on Essay", y = "Essay Mark
(%)")
```

The resulting scatterplot should look like this:



Next, we should check whether the data are parametric using the Shapiro–Wilk test (see Chapter 5 in the textbook). To do a Shapiro–Wilk test in **R**, we use the *shapiro.test()* function. We will test the essay and numeracy variables for normality:

```
shapiro.test(essayData$essay)
```

```
Shapiro-Wilk normality test

data:  essayData$hours
W = 0.981, p-value = 0.6615
```

```
shapiro.test(essayData$hours)
```

```
 Shapiro-Wilk normality test

data:  essayData$hours
W = 0.981, p-value = 0.6615
```

The Shapiro–Wilk statistics are non-significant ($p > .05$) for both variables, which indicates that they are normally distributed. As such we can use Pearson's correlation coefficient, the result of which is:

```
cor.test(essayData$essay, essayData$hours, alternative = "greater", method =
"pearson")
```

```
 Pearson's product-moment correlation

 data:  essayData$essay and essayData$hours
 t = 1.8145, df = 43, p-value = 0.03829
 alternative hypothesis: true correlation is greater than 0
 95 percent confidence interval:
  0.01948135 1.00000000
 sample estimates:
       cor
 0.2666837
```

I chose a one-tailed test (*alternative = "greater"*) because a specific prediction was made: there would be a positive relationship; that is, the more time you spend on your essay, the better the mark you'll get. This hypothesis is supported because Pearson's $r = .27$ (a medium effect size), $p < .05$, is significant.

The second part of the question asks us to do the same analysis but when the percentages are recoded into degree classifications. The degree classifications are ordinal data (not interval): they are

ordered categories, so we shouldn't use Pearson's test statistic, but Spearman's and Kendall's ones instead:

```
cor.test(essayData$hours, essayData$grade, alternative = "less", method = "kendall")
```

```
Kendall's rank correlation tau

data:  essayData$hours and essayData$grade
z = -1.3458, p-value = 0.08918
alternative hypothesis: true tau is less than 0
sample estimates:
       tau
-0.1575566
```

```
cor.test(essayData$hours, essayData$grade, alternative = "less", method = "spearman")
```

```
Spearman's rank correlation rho

data:  essayData$hours and essayData$grade
S = 18110.93, p-value = 0.1019
alternative hypothesis: true rho is less than 0
sample estimates:
       rho
-0.1930781
```

In both cases the correlation is non-significant. There was no significant relationship between degree grade classification for an essay and the time spent doing it, $\rho = -.19$, *ns*, and $\tau = -.16$, *ns*. Note that the direction of the relationship has reversed. This has happened because the essay marks were recoded as 1 (first), 2 (upper second), 3 (lower second), and 4 (third), so high grades were represented by low numbers!

This illustrates one of the benefits of *not* taking continuous data (like percentages) and transforming them into categorical data: when you do, you lose information and often statistical power!

Task 2

- Using the **ChickFlick.dat** data from Chapter 3, is there a relationship between gender and arousal? Using the same data, is there a relationship between the film watched and arousal?

Now, both gender and the film watched are categorical variables with two categories. Therefore, we need to look at these relationships using point-biserial correlations:

```
cor.test(chickFlick$gender, chickFlick$arousal)
```

```
Pearson's product-moment correlation

data:  chickFlick$gender and chickFlick$arousal
t = -1.1291, df = 38, p-value = 0.2659
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 -0.4655482  0.1391519
sample estimates:
       cor
-0.1801672
```

```
cor.test(chickFlick$film, chickFlick$arousal)
```

```
Pearson's product-moment correlation

data:  chickFlick$film and chickFlick$arousal
t = 5.1104, df = 38, p-value = 9.4e-06
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 0.4077815 0.7922253
sample estimates:
      cor
0.6382193
```

In both cases I used a two-tailed test because no prediction was made. As you can see, there was no significant relationship between gender and arousal, $r_{pb} = -.18$, *ns*. However, there was a significant relationship between the film watched and arousal, $r_{pb} = -.64$, $p < .001$. Looking at how the groups were coded, you should see that *Bridget Jones's Diary* had a code of 1, and *Memento* had a code of 2, therefore this result reflects the fact that as film goes up (changes from 1 to 2) arousal goes up. Put another way, as the film changes from *Bridget Jones's Diary* to *Memento*, arousal increases. So, *Memento* gave rise to the greater arousal levels.

Task 3

- As a statistics lecturer I am always interested in the factors that determine whether a student will do well on a statistics course. One potentially important factor is their previous expertise with mathematics. Imagine I took 25 students and looked at their degree grades for my statistics course at the end of their first year at university: first, upper second, lower second and third class. I also asked these students what grade they got in their GCSE maths exams. In the UK GCSEs are school exams taken at age 16 that are graded A, B, C, D, E or F (an A grade is better than all of the lower grades). The data for this study are in the file **grades.csv**. Carry out the appropriate analysis to see if GCSE maths grades correlate with first-year statistics grades. ①

Let's look at these variables. In the UK, a student can get a first-class mark, an upper second, a lower second, a third, a pass or a fail. These grades are categories, but they have an order to them (an upper second is better than a lower second). In most of the UK, GCSEs are school exams taken at age 16 that are graded A, B, C, D, E or F. Again, these grades are categories that have an order of importance (an A grade is better than all of the lower grades). When you have categories like these that can be ordered in a meaningful way, the data are said to be *ordinal*. The data are not interval, because a first-class degree encompasses a 30% range (70–100%) whereas an upper second only covers a 10% range (60–70%). When data have been measured at only the ordinal level they are said to be non-parametric and Pearson's correlation is not appropriate. Therefore, the Spearman correlation coefficient is used.

As always, we need to load the data first:

```
gradesData = read.csv("grades.csv", header = TRUE)
```

We can then view the data by executing:

```
gradesData
```

The data are in two columns: one labelled **stats** and one labelled **gcse**. Each of the categories described above has been coded with a numeric value. In both cases, the highest grade (first class or A grade) has been coded with the value 1, with subsequent categories being labelled 2, 3 and so on.

To conduct a Spearman's correlation, we can execute the following command:

```
cor.test(gradesData$gcse, gradesData$stats, alternative = "greater", method =
"spearman")
```

Note that I have specified a one-tailed test by including *alternative = "greater"*. This is because earlier I predicted that better grades in GCSE maths would correlate with better degree grades for my statistics course. This hypothesis is for a positive directional relationship and so *alternative = "greater"* should be specified.

```
Spearman's rank correlation rho

data:  gradesData$gcse and gradesData$stats
S = 1418.034, p-value = 0.01121
alternative hypothesis: true rho is greater than 0
sample estimates:
      rho
```

```
0.4546021
```

The output shows the Spearman correlation on the variables **stats** and **gcse**. The output shows the correlation coefficient between the two variables (.455) and the significance value of this coefficient (.011). The significance value for this correlation coefficient is less than .05; therefore, it can be concluded that there is a significant relationship between a student's grade in GCSE maths and their degree grade for their statistics course. The correlation itself is positive: therefore, we can conclude that as GCSE grades improve, there is a corresponding improvement in degree grades for statistics. As such, the hypothesis was supported.

We could also look at Kendall's correlation by simply changing *method = "spearman"* to *method = "kendall"* in the previous code:

```
cor.test(gradesData$gcse, gradesData$stats, alternative = "greater", method =
"kendall")
```

```
Kendall's rank correlation tau

data:  gradesData$gcse and gradesData$stats
z = 2.1794, p-value = 0.01465
alternative hypothesis: true tau is greater than 0
sample estimates:
      tau
0.3539614
```

The actual value of the correlation coefficient is less than Spearman's correlation (it has decreased from .455 to .354). Despite the difference in the correlation coefficients, we can still interpret this result as being a highly significant positive relationship (because the significance value of .015 is less than .05). However, Kendall's value is a more accurate gauge of what the correlation in the population would be. As with Pearson's correlation, we cannot assume that the GCSE grades caused the degree students to do better in their statistics course.

We could report these results as follows:

- ✓ There was a positive relationship between a person's statistics grade and their GCSE maths grade, $r_s = .46$, $p < .05$.
- ✓ There was a positive relationship between a person's statistics grade and their GCSE maths grade, $\tau = .35$, $p < .05$. (Note that I've quoted Kendall's tau here.)