Non-parametric tests

Self-test answers

- Carry out some analyses to test for normality and homogeneity of variance in these data.

To get the outputs in the book execute the following commands:

```
by(drugData[,c(2:3)], drug, stat.desc, basic=FALSE, norm=TRUE)
leveneTest(drugData$sundayBDI, drugData$drug, center = "mean")
leveneTest(drugData$wedsBDI, drugData$drug, center = "mean")
```

- Compute the change in BDI scores from Sunday to Wednesday and then compute normality tests for this change score separately for the alcohol and ecstasy groups.

To compute the change in BDI scores execute:

```
drugData$BDIchange<-drugData$wedsBDI-drugData$sundayBDI
```

This creates a variable called *BDIchange* in the *drugData* dataframe:

```
     drug sundayBDI wedsBDI BDIchange
1  Ecstasy        15      28        13
2  Ecstasy        35      35         0
3  Ecstasy        16      35        19
4  Ecstasy        18      24         6
5  Ecstasy        19      39        20
6  Ecstasy        17      32        15
7  Ecstasy        27      27         0
8  Ecstasy        16      29        13
9  Ecstasy        13      36        23
10 Ecstasy        20      35        15
11 Alcohol        16       5       -11
12 Alcohol        15       6        -9
13 Alcohol        20      30        10
14 Alcohol        15       8        -7
15 Alcohol        16       9        -7
16 Alcohol        13       7        -6
17 Alcohol        14       6        -8
18 Alcohol        19      17        -2
19 Alcohol        18       3       -15
20 Alcohol        18      10        -8
```

We can get some descriptive statistics (and normality tests) for the change score in each group by executing:

```
by(drugData$BDIchange, drugData$drug, stat.desc, basic = FALSE, norm = TRUE)
```

- Use the *subset()* function to create separate dataframes for the different drugs called *alcoholData* and *ecstasyData*.

```
alcoholData<-subset(drugData, drug == "Alcohol")
ecstasyData<-subset(drugData, drug == "Ecstacy")
```

- See whether you can enter the data in Table 15.3 into **R** (you don't need to enter the ranks). Then conduct some exploratory analysis on the data.

When the data are collected using different participants in each group, we need to input the data using a factor variable. So, the data editor will have two columns of data. The first column is a coding variable (called something like **Soya**) which, in this case, will have four levels. We can create this variable using the *gl()* function by executing:

```
Soya<-gl(4, 20, labels = c("No Soya", "1 Soya Meal", "4 Soya Meals", "7 Soya Meals"))
```

This command creates a variable called **Soya**, containing four blocks of 20 rows of data: the first block will be labelled *No Soya*, the second block *1 Soya Meal*, and so on. The second column will have values for the dependent variable (**Sperm**). We can, therefore, create these variables by executing:

```
Sperm<-c(0.35, 0.58, 0.88, 0.92, 1.22, 1.51, 1.52, 1.57, 2.43, 2.79, 3.40, 4.52, 4.72,
6.90, 7.58, 7.78, 9.62, 10.05, 10.32, 21.08, 0.33, 0.36, 0.63, 0.64, 0.77, 1.53, 1.62,
1.71, 1.94, 2.48, 2.71, 4.12, 5.65, 6.76, 7.08, 7.26, 7.92, 8.04, 12.10, 18.47, 0.40,
0.60, 0.96, 1.20, 1.31, 1.35, 1.68, 1.83, 2.10, 2.93, 2.96, 3.00, 3.09, 3.36, 4.34,
5.81, 5.94, 10.16, 10.98, 18.21, 0.31, 0.32, 0.56, 0.57, 0.71, 0.81, 0.87, 1.18, 1.25,
1.33, 1.34, 1.49, 1.50, 2.09, 2.70, 2.75, 2.83, 3.07, 3.28, 4.11)
```

Finally, we can tie these variables together in a dataframe called *soyaData* by executing:

```
soyaData<-data.frame(Sperm, Soya)
```

To get the outputs in the book execute:

```
by(soyaData$Sperm, soyaData$Soya, stat.desc, basic=FALSE, norm=TRUE)
leveneTest(soyaData$Sperm, soyaData$Soya, center = "mean")
```

- Use *ggplot2* to draw a boxplot of these data.

```
ggplot(soyaData, aes(Soya, Sperm)) + geom_boxplot() +
  labs(y = "Sperm Count", x = "Number of Soya Meals Per Week")
```

- Using what you know about inputting data, try to enter these data into **R** and run some exploratory analyses.

To enter the data execute:

```
Start<-c(63.75, 62.98, 65.98, 107.27, 66.58, 120.46, 62.01, 71.87, 83.01, 76.62)
Month1<-c(65.38, 66.24, 67.70, 102.72, 69.45, 119.96, 66.09, 73.62, 75.81, 67.66)
Month2<-c( 81.34, 69.31, 77.89, 91.33, 72.87, 114.26, 68.01, 55.43, 71.63, 68.60)
dietData<-data.frame(Start, Month1, Month2)
```

To get exploratory analysis execute:

```
stat.desc(dietData, basic = FALSE, norm = TRUE)
```

Oliver Twisted

Please Sir, can I have some more … Jonck?

Jonckheere's test is based on the simple, but elegant, idea of taking a score in a particular condition and counting how many scores in subsequent conditions are smaller than that score. So, the first step is to order your groups in the way that you expect your medians to change. If we take the soya example from Chapter 13, then

we expect sperm counts to be highest in the no soya meals condition, and then decrease in the following order: 1 meal per week, 4 meals per week, 7 meals per week. So, we start with the no meals per week condition, and we take the first score and ask 'How many scores in the next condition are bigger than this score?'.

You'll find that this is easy to work out if you arrange your data in ascending order in each condition. Table 1 shows a convenient way to lay out the data. Note that the sperm counts have been put in ascending order and the groups have been ordered in the way that we expect our medians to decrease. So, starting with the first score in the no soya meals group (this score is 0.35), we look at the scores in the next condition (1 soya meal) and count how many are greater than 0.35. It turns out that all 19 of the 20 scores are greater than this value, so we place the value of 19 in the appropriate column and move onto the next score (0.58) and do the same. When we've done this for all of the scores in the no-meals group, we go back to the first score (0.35) again, but this time count how many scores are bigger in the next but one condition (the 4 soya meals condition). It turns out that 18 scores are bigger so we register this in the appropriate column and move onto the next score (0.58) and do the same until we've done all of the scores in the 7 soya meals group. We basically repeat this process until we've compared the first group with all subsequent groups.

At this stage we move onto the next group (1 soya meal). Again, we start with the first score (0.33) and count how many scores are bigger than this value in the subsequent group (the 4 soya meals group). In this case there all 20 scores are bigger than 0.33, so we register this in the table and move onto the next score (0.36). Again, we repeat this for all scores and then go back to the beginning of this group (i.e. back to the first score of 0.33) and repeat the process until this category has been compared with all subsequent categories. When all categories have been compared with all subsequent categories in this way, we simply add up the counts as I have done in the table. These sums of counts are denoted by U_{ij}.

Table 1: **Data to show Jonckheere's test for the soya example**

No Soya Meals				1 Soya Meal			4 Soya Meals		7 Soya Meals
Sperm	How many scores are bigger in the…			Sperm	How many scores are bigger in the…		Sperm	How many scores are bigger in the…	Sperm
	1 Meal	4 Meals	7 Meals		4 Meals	7 Meals		7 Meals	
0.35	19	20	18	0.33	20	18	0.40	18	0.31
0.58	18	19	16	0.36	20	18	0.60	16	0.32
0.88	18	18	13	0.63	18	16	0.96	13	0.56
0.92	15	18	13	0.64	18	16	1.20	12	0.57
1.22	15	15	12	0.77	18	15	1.31	11	0.71
1.51	15	14	7	1.53	14	7	1.35	9	0.81
1.52	15	14	7	1.62	14	7	1.68	7	0.87
1.57	14	14	7	1.71	13	7	1.83	7	1.18
2.43	10	11	6	1.94	12	7	2.10	6	1.25
2.79	9	11	4	2.48	11	6	2.93	3	1.33
3.40	9	6	1	2.71	11	5	2.96	3	1.34
4.52	8	5	0	4.12	6	0	3.00	3	1.49
4.72	8	5	0	5.65	4	0	3.09	2	1.50
6.90	6	3	0	6.76	3	0	3.36	1	2.09
7.58	4	3	0	7.08	3	0	4.34	0	2.70
7.78	4	3	0	7.26	3	0	5.81	0	2.75
9.62	2	3	0	7.92	3	0	5.94	0	2.83
10.05	2	3	0	8.04	3	0	10.16	0	3.07
10.32	2	2	0	12.10	1	0	10.98	0	3.28
21.08	0	0	0	18.47	0	0	18.21	0	4.11
Total (U_{ij})	**193**	**187**	**104**		**195**	**122**		**111**	

The test statistic, *J*, is simply the sum of these counts:

$$J = \sum_{i<j}^{k} U_{ij}$$

which for these data is simply:

$$J = \sum_{i<j}^{k} U_{ij} = 193 + 187 + 104 + 195 + 122 + 111$$
$$= 912$$

For small samples there are specific tables to look up critical values of *J*; however, when samples are large (anything bigger than about 8 per group would be large in this case) the sampling distribution of *J* becomes normal with a mean and standard deviation of:

$$\bar{J} = \frac{N^2 - \sum n_k^2}{4}$$

$$\sigma_j = \sqrt{\frac{1}{72}\left[N^2(2N+3) - \sum n_k^2(2n_k+3)\right]}$$

in which *N* is simply the total sample size (in this case 80) and n_k is simply the sample size of group *k* (in each case in this example this will be 20 because we have equal sample sizes). So, we just square each group's sample size and add them up, then subtract this value from the total sample size squared. We then divide the result by 4. Therefore, we can calculate the mean for these data as:

$$\begin{aligned}\bar{J} &= \frac{80^2 - \left(20^2 + 20^2 + 20^2 + 20^2\right)}{4} \\ &= \frac{4800}{4} \\ &= 1200\end{aligned}$$

The standard deviation can similarly be calculated using the sample sizes of each group and the total sample size:

$$\begin{aligned}&\sigma_j \\ &= \sqrt{\frac{1}{72}\left\{80^2\left((2\times 80)+3\right) - \left[20^2\left((2\times 20)+3\right) + 20^2\left((2\times 20)+3\right) + 20^2\left((2\times 20)+3\right) + 20^2\left((2\times 20)+3\right)\right]\right\}} \\ &= \sqrt{\frac{1}{72}\{6400(163) - [400(43) + 400(43) + 400(43) + 400(43)]\}} \\ &= \sqrt{\frac{1}{72}\{1043200 - 68800\}} \\ &= \sqrt{1353333} \\ &= 116.33\end{aligned}$$

We can use the mean and standard deviation to convert *J* to a *z*-score (see Chapter 1) using the standard formulae:

$$z = \frac{x - \bar{x}}{s} = \frac{J - \bar{J}}{\sigma_j} = \frac{912 - 1200}{116.33} = -2.476$$

This *z* can then be evaluated using the critical values in the Appendix of the book. This test is always one-tailed because we have predicted a trend to use the test. So we're looking at *z* being above 1.65 (when ignoring the sign) to be significant. In fact, the sign of the test tells us whether the medians ascend across the groups (a positive *z*) or descend across the groups (a negative *z*) as they do in this example!

Does it matter how I order my groups?

I have just showed how to use the test when the groups are ordered by descending medians (i.e. we expect sperm counts to be highest in the no soya meals condition, and then decrease in the order 1 meal per week, 4 meals per week, 7 meals per week; so we ordered the groups: no soya, 1 meal, 4 meals and 7 meals). Certain books will tell you to order the groups in ascending order (i.e. start with the group that you expect to have the lowest median). For the soya data this would mean arranging the groups in the opposite order to how I did above; that is, 7 meals, 4 meals, 1 meal and no meals. The purpose of this section is to show you what happens if we order the groups the opposite way around!

The process is similar to that used in the previous section, only now we start with start with the 7 meals per week condition, and we take the first score and ask 'How many scores in the next condition are bigger than this score?' You'll find that this is easy to work out if you arrange your data in ascending order in each condition. Table 2 shows a convenient way to lay out the data. Note that the sperm counts have been ordered in ascending order and the groups have been ordered in the way that we expect our medians to increase. So, starting with the first score in the 7 soya meals group (this score is 0.31), we look at the scores in the next condition (4 soya meals) and count how many are greater than 0.31. It turns out that all 20 scores are greater than this value, so we place the value of 20 in the appropriate column and move on to the next score (0.32) and do the same. When we've done this for all of the scores in the 7 meals group, we go back to the first score (0.31) again, but this

time count how many scores are bigger in the next but one condition (the 1 soya meal condition). It turns out that all 20 scores are bigger, so we register this in the appropriate column and move on to the next score (0.32) and do the same until we've done all of the scores in the 7 meals group. We basically repeat this process until we've compared the first group to all subsequent groups.

Table 2: Data to show Jonckheere's test for the soya example

7 Soya Meals				4 Soya Meals			1 Soya Meal		No Soya Meals
Sperm	How many scores are bigger in the…			Sperm	How many scores are bigger in the…		Sperm	How many scores are bigger in the…	Sperm
	4 Meals	1 Meal	No Meals		1 Meal	No Meals		No Meals	
0.31	20	20	20	0.40	18	19	0.33	20	0.35
0.32	20	20	20	0.60	18	18	0.36	19	0.58
0.56	19	18	19	0.96	15	16	0.63	18	0.88
0.57	19	18	19	1.20	15	16	0.64	18	0.92
0.71	18	16	18	1.31	15	15	0.77	18	1.22
0.81	18	15	18	1.35	15	15	1.53	13	1.51
0.87	18	15	18	1.68	13	12	1.62	12	1.52
1.18	17	15	16	1.83	12	12	1.71	12	1.57
1.25	16	15	15	2.10	11	12	1.94	12	2.43
1.33	15	15	15	2.93	9	10	2.48	11	2.79
1.34	15	15	15	2.96	9	10	2.71	11	3.40
1.49	14	15	15	3.00	9	10	4.12	9	4.52
1.50	14	15	15	3.09	9	10	5.65	7	4.72
2.09	12	11	12	3.36	9	10	6.76	7	6.90
2.70	11	10	11	4.34	8	9	7.08	6	7.58
2.75	11	9	11	5.81	7	7	7.26	6	7.78
2.83	11	9	10	5.94	7	7	7.92	4	9.62
3.07	8	9	10	10.16	2	2	8.04	4	10.05
3.28	7	9	10	10.98	2	1	12.10	1	10.32
4.11	6	9	9	18.21	1	1	18.47	1	21.08
Total (U_{ij})	**289**	**278**	**296**		**204**	**212**		**209**	

At this stage we move on to the next group (the 4 soya meals). Again, we start with the first score (0.40) and count how many scores are bigger than this value in the subsequent group (the 1 meal group). In this case there are 18 scores bigger than 0.40, so we register this in the table and move on to the next score (0.60). Again, we repeat this for all scores and then go back to the beginning of this group (i.e. back to the first score of 0.40) and repeat the process until this category has been compared to all subsequent categories. When all categories have been compared to all subsequent categories in this way, we simply add up the counts as I have done in the table. These sums of counts are denoted by U_{ij}.

As before, the test statistic J is simply the sum of these counts:

$$J = \sum_{i<j}^{k} U_{ij}$$

which for these data is simply:

$$J = \sum_{i<j}^{k} U_{ij} = 289 + 278 + 296 + 204 + 212 + 209$$
$$= 1488$$

As I said earlier, for small samples there are specific tables to look up critical values of *J*; however, when samples are large (anything bigger than about 8 per group would be large in this case) the sampling distribution of *J* becomes normal with a mean and standard deviation of:

$$\bar{J} = \frac{N^2 - \sum n_k^2}{4}$$
$$\sigma_j = \sqrt{\frac{1}{72}\left[N^2(2N+3) - \sum n_k^2(2n_k+3)\right]}$$

in which *N* is simply the total sample size (in this case 80) and n_k is simply the sample size of group *k* (in each case in this example this will be 20 because we have equal sample sizes). So, we just square each group's sample size and add them up, then subtract this value from the total sample size squared. We then divide the result by 4. Therefore, we can calculate the mean for these data as we did earlier:

$$\bar{J} = \frac{80^2 - \left(20^2 + 20^2 + 20^2 + 20^2\right)}{4}$$
$$= \frac{4800}{4}$$
$$= 1200$$

The standard deviation can similarly be calculated using the sample sizes of each group and the total sample size (again this is the same as earlier):

$$\sigma_j$$
$$= \sqrt{\frac{1}{72}\{80^2((2\times 80)+3) - [20^2((2\times 20)+3) + 20^2((2\times 20)+3) + 20^2((2\times 20)+3) + 20^2((2\times 20)+3)]\}}$$
$$= \sqrt{\frac{1}{72}\{6400(163) - [400(43) + 400(43) + 400(43) + 400(43)]\}}$$
$$= \sqrt{\frac{1}{72}\{1043200 - 68800\}}$$
$$= \sqrt{1353333}$$
$$= 116.33$$

We can use the mean and standard deviation to convert *J* to a *z*-score (see Chapter 1) using the standard formulae. The mean and standard deviation are the same as before, but we now have a different test statistic (it is 1491 rather than 912). So, let's see what happens when we plug this new test statistic into the equation:

$$z = \frac{x - \bar{x}}{s} = \frac{J - \bar{J}}{\sigma_j} = \frac{1488 - 1200}{116.33} = 2.476$$

Note that the *z*-score is the same value as when we ordered the groups in descending order, except that it now has a positive value rather than a negative one! This goes to prove what I wrote earlier: the sign of the test tells us whether the medians ascend across the groups (a positive *z*) or descend across the groups (a negative *z*)! Earlier we ordered the groups in descending order and so got a negative *z*, and now we ordered them in ascending order and so got a positive *z*.

Labcoat Leni's real research

Having a quail of a time?

Problem

Matthews, R. C. et al. (2007). *Psychological Science*, *18*(9), 758–762.

We encountered some research in Chapter 2 in which we discovered that you can influence aspects of male quail's sperm production through 'conditioning'. The basic idea is that the male is granted access to a female for copulation in a certain chamber (e.g. one that is coloured green) but gains no access to a female in a different context (e.g. a chamber with a tilted floor). The male, therefore, learns that when he is in the green chamber his luck is in, but if the floor is tilted then frustration awaits. For other males the chambers will be reversed (i.e. they get sex only when in the chamber with the tilted floor). During the test phase, males get to mate in both chambers; the question is: after the males have learnt that they will get a mating opportunity in a certain context, do they produce more sperm or better-quality sperm when mating in that context compared to the control context?

Mike Domjan and his colleagues predicted that if conditioning evolved because it increases reproductive fitness then males who mated in the context that had previously signalled a mating opportunity would fertilize a significantly greater number of eggs than quails that mated in their control context (Matthews, Domjan, Ramsey, & Crews, 2007). They put this hypothesis to the test in an experiment that is utter genius. After training, they allowed 14 females to copulate with two males (counterbalanced): one male copulated with the female in the chamber that had previously signalled a reproductive opportunity (**Signalled**), whereas the second male copulated with the same female but in the chamber that had not previously signalled a mating opportunity (**Control**). Eggs were collected from the females for 10 days after the mating and a genetic analysis was used to determine the father of any fertilized eggs.

The data from this study are in the file **Matthews et al. (2007).dat**. Labcoat Leni wants you to carry out a Wilcoxon signed-rank test to see whether more eggs were fertilized by males mating in their signalled context compared to males in their control context.

Solution

First of all, load in the **Matthews et al. (2007).dat** data:

```
matthewsData<-read.delim("Matthews et al. (2007).dat", header = TRUE)
```

We can have a look at the data by executing:

```
matthewsData
    Female_quail    Signalled Control
1      Quail 01        1       0
2      Quail 02        3       0
3      Quail 03        1       0
4      Quail 04        1       2
5      Quail 05        2       2
6      Quail 06        1       0
7      Quail 07        2       0
8      Quail 08        3       0
9      Quail 09        3       0
10     Quail 10        3       2
11     Quail 11        4       0
12     Quail 12        0       1
13     Quail 13        1       2
14     Quail 14        3       2
```

You will see that we have the data for different groups stored in two columns.

The *wilcox.test()* function takes the form:

```
newModel<-wilcox.test(scores group 1, scores group 2, paired = FALSE/TRUE)
```

Therefore, to compute a basic Wilcoxon test for the current data we could execute:

```
matthewsModel<-wilcox.test(matthewsData$Signalled, matthewsData$Control,  paired =
 TRUE, correct= FALSE)
```

Having left all of the default options as they are, **R** will calculate the *p*-value, using the exact approach if *N* is less than 40 and there are no ties, or the normal approximation approach if *N* is more than 40 or if there are any ties. It will also use a continuity correction. To use a normal approximation rather than exact *p*, and to get rid of the continuity correction, we can exclude *exact = FALSE* and *correct = FALSE* respectively:

```
matthewsModel<-wilcox.test((matthewsData$Signalled, matthewsData$Control, exact =
FALSE, correct= FALSE)
```

We can then view the output by executing the name of the model:

```
matthewsModel
```

```
Wilcoxon rank sum test

data:  matthewsData$Signalled and matthewsData$Control
W = 153.5, p-value = 0.00842
alternative hypothesis: true location shift is not equal to 0
```

We can see from the output above that there were a greater number of fertilized eggs from males mating in their signalled context than in the control context, $W = 153.5$, $p < .05$. In other words, conditioning (as a learning mechanism) provides some adaptive benefit in that it makes it more likely that you will pass on your genes.

The authors concluded as follows: 'The present findings show that when 2 males copulated with the same female in succession, the male that received a Pavlovian CS signalling copulatory opportunity fertilized more of the female's eggs. Thus, Pavlovian conditioning increased reproductive fitness in the context of sperm competition' (p. 760).

Labcoat Leni's real research

Eggs-traordinary!

Problem

Çetinkaya, H., & Domjan, M. (2006). *Journal of Comparative Psychology*, *120*(4), 427–432.

We saw that male quail fertilized more eggs if they had been trained to be able to predict when a mating opportunity would arise. However, some quail develop fetishes. Really. In the previous example the type of compartment acted as a predictor of an opportunity to mate, but in studies where a terrycloth object acts as a sign that a mate will shortly become available, some quail start to direct their sexual behaviour towards the terrycloth object. In evolutionary terms, this fetishistic behaviour seems counterproductive because sexual behaviour becomes directed towards something that cannot provide reproductive success. However, perhaps this behaviour serves to prepare the organism for the 'real' mating behaviour.

Hakan Çetinkaya and Mike Domjan conducted a brilliant study in which they sexually conditioned male quail (Çetinkaya & Domjan, 2006). All quail experienced the terrycloth stimulus and an opportunity to mate, but for some the terrycloth stimulus immediately preceded the mating opportunity (paired group) whereas for others they experienced it 2 hours after the mating opportunity (this was the control group because the terrycloth stimulus did not predict a mating

opportunity). In the paired group, quail were classified as fetishistic or not depending on whether they engaged in sexual behaviour with the terrycloth object.

During a test trial the quail mated with a female and the researchers measured the percentage of eggs fertilized, the time spent near the terrycloth object, the latency to initiate copulation, and copulatory efficiency. If this fetishistic behaviour provides an evolutionary advantage then we would expect the fetishistic quail to fertilize more eggs, initiate copulation faster and be more efficient in their copulations.

The data from this study are in the file **Cetinkaya & Domjan (2006).dat**. Labcoat Leni wants you to carry out a Kruskal–Wallis test to see whether fetishist quail produced a higher percentage of fertilized eggs and initiated sex more quickly.

Solution

First of all, load in the data:

```
quailData<-read.delim("Cetinkaya & Domjan (2006).dat", header = TRUE)
```

The variable **Groups**, which contains text, will be imported as a factor and the order of the groups will be alphabetic: Control, Fetishistics, NonFetishistics. For reasons that will become apparent, it's useful to have the control condition as the first level, which we have already got, then NonFetishistics followed by Fetishistics. Therefore, we need to reorder the factor levels. We can do this by executing:

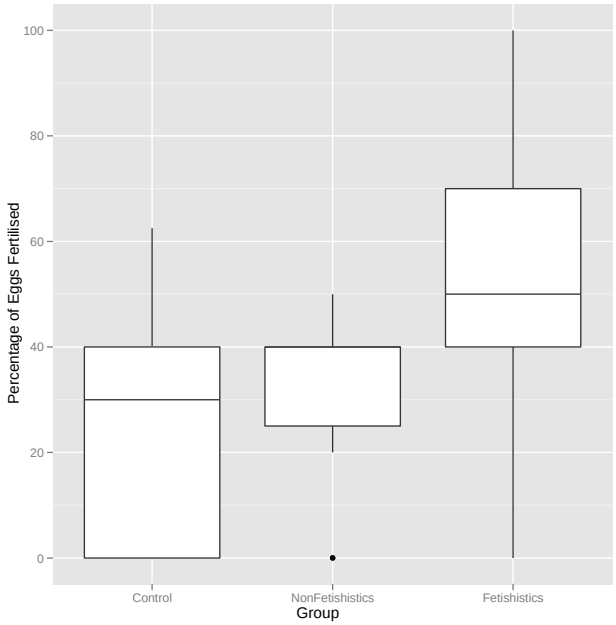
```
quailData$Groups<-factor(quailData$Groups, levels = levels(quailData$Groups)[c(1, 3,
2)])
```

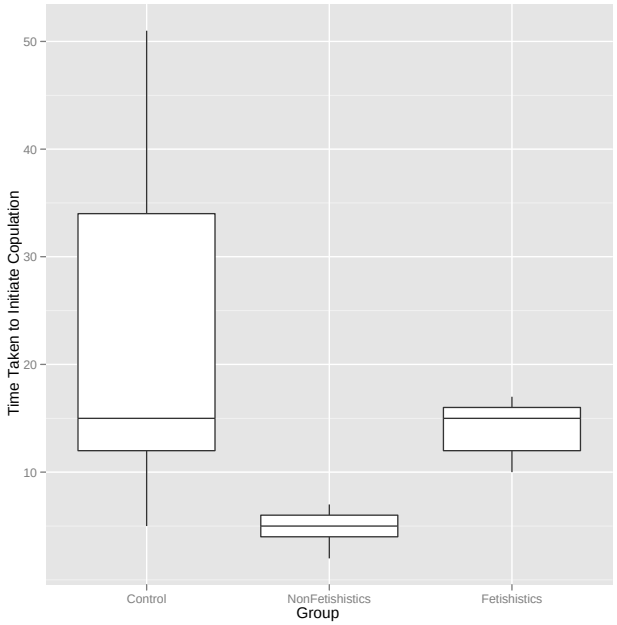
This command uses the *factor()* function to reorder the levels of the **Groups** variable. It re-creates the variable **Groups** in the *quailData* dataframe (*quailData$Soya*) based on itself, but then uses the *levels()* function to reorder the groups.

Let's create a boxplot for each of the outcome variables by executing:

```
ggplot(quailData, aes(Groups, Egg_Percent)) + geom_boxplot() +
    labs(y = "Percentage of Eggs Fertilised", x = "Group")

ggplot(quailData, aes(Groups, Latency)) + geom_boxplot() +
  labs(y = "Time Taken to Initiate Copulation", x = "Group")
```





We can next do the Kruskal–Wallis test for the dependent variable **Egg_Percent** using the *kruskal.test()* function and by executing:

```
kruskal.test(Egg_Percent ~ Groups, data = quailData)
```

Note that we have executed the command directly, without creating a model, which is fine because we don't really need to use the output of the function for any reason other than interpretation.

To interpret the Kruskal–Wallis test, it is useful to obtain the mean rank for each group. We can do this by adding a variable called **Ranks** to the dataframe with the *rank()* function:

```
quailData$Ranks<-rank(quailData$Egg_Percent)
```

This command creates a variable **Ranks** in *quailData* dataframe that is the ranks for the variable **Egg_Percent**. We can then obtain the mean rank for each group using the *by()* function in conjunction with the *mean()* function:

```
by(quailData$Ranks, quailData$Groups, mean)
```

We can then do exactly the same for the dependent variable **Latency** by executing:

```
kruskal.test(Latency ~ Groups, data = quailData)
quailData$Ranks<-rank(quailData$Latency)
by(quailData$Ranks, quailData$Groups, mean)
```

```
Kruskal-Wallis rank sum test

data:  Egg_Percent by Groups
Kruskal-Wallis chi-squared = 11.9552, df = 2, p-value = 0.002535

> quailData$Ranks<-rank(quailData$Egg_Percent)
> by(quailData$Ranks, quailData$Groups, mean)
quailData$Groups: Control
[1] 24.24074
------------------------------------------------------------------------------------------
quailData$Groups: NonFetishistics
[1] 26.96667
------------------------------------------------------------------------------------------
quailData$Groups: Fetishistics
[1] 41.82353


> kruskal.test(Latency ~ Groups, data = quailData)

        Kruskal-Wallis rank sum test

data:  Latency by Groups
```

```
Kruskal-Wallis chi-squared = 32.2441, df = 2, p-value = 9.96e-08

> quailData$Ranks<-rank(quailData$Latency)
> by(quailData$Ranks, quailData$Groups, mean)
quailData$Groups: Control
[1] 39.59259
------------------------------------------------------------------------------------
quailData$Groups: NonFetishistics
[1] 8.8
------------------------------------------------------------------------------------
quailData$Groups: Fetishistics
[1] 33.47059
```

Looking at the two output tables above, we can see that for both variables there is a significant effect. So there are differences between the groups, but we don't know where these differences lie. We can investigate this by doing some *post hoc* tests using the *kruskalmc()* function, executing:

```
kruskalmc(Egg_Percent ~ Groups, data = quailData)
 kruskalmc(Latency ~ Groups, data = quailData)
```

```
> kruskalmc(Egg_Percent ~ Groups, data = quailData)
Multiple comparison test after Kruskal-Wallis
p.value: 0.05
Comparisons
                              obs.dif critical.dif difference
Control-NonFetishistics      2.725926     13.24124      FALSE
Control-Fetishistics        17.582789     12.73068       TRUE
NonFetishistics-Fetishistics 14.856863     14.56587       TRUE

> kruskalmc(Latency ~ Groups, data = quailData)
Multiple comparison test after Kruskal-Wallis
p.value: 0.05
Comparisons
                              obs.dif critical.dif difference
Control-NonFetishistics      30.792593     13.24124       TRUE
Control-Fetishistics          6.122004     12.73068      FALSE
NonFetishistics-Fetishistics 24.670588     14.56587       TRUE
```

For the Kruskal–Wallis test, we need only report the test statistic (denoted by H), its degrees of freedom and its significance. So, we could report something like: 'Kruskal–Wallis analysis of variance (ANOVA) confirmed that female quail partnered with the different types of male quail produced different percentages of fertilized eggs, $H(2) = 11.96$, $p < .05$. Subsequent pairwise comparisons of the mean ranks between groups indicated that fetishistic male quail yielded higher rates of fertilization than both the nonfetishistic male quail (*difference* = 14.86) and the control male quail (*difference* = 17.59). However, the nonfetishistic group was not significantly different from the control group (*difference* = 2.73)' (p. 429). For the latency data: 'A Kruskal–Wallis analysis indicated significant group differences, $H(2) = 32.24$, $p < .05$. Pairwise comparisons of the mean ranks between groups indicated that that the nonfetishistic males had significantly shorter copulatory latencies than both the fetishistic male quail (*difference* = 24.67) and the control male quail (*difference* = 30.79). However, the fetishistic group was not significantly different from the control group (*difference* = 6.12)' (p. 430).

These results support the authors' theory that fetishist behaviour may have evolved because it offers some adaptive function (such as preparing for the real thing).

Smart Alex's solutions

Task 1

- A psychologist was interested in the cross-species differences between men and dogs. She observed a group of dogs and a group of men in a naturalistic setting (20 of each). She classified several behaviours as being dog-like (urinating against trees and lamp posts, attempts to copulate with anything that moved, and attempts to lick their own genitals). For each man and dog she counted the number of dog-like behaviours displayed in a 24-hour period. It was hypothesized that dogs would display more dog-like behaviours than men. The data are in the file **MenLikeDogs.dat**. Analyse them with a Wilcoxon rank-sum test.

First of all, load in the data:

```
MenDogs<-read.delim("MenLikeDogs.dat", header = TRUE)
```

We can then view the data by executing the name of the dataframe:

```
 MenDogs
```

```
    species  behaviou
1       Dog        21
2       Dog        21
3       Dog        42
4       Dog        18
5       Dog        15
6       Dog        24
7       Dog        36
8       Dog        36
9       Dog        18
10      Dog        24
11      Dog        30
12      Dog        39
13      Dog        21
14      Dog        15
15      Dog        51
16      Dog        24
17      Dog        48
18      Dog        18
19      Dog        36
20      Dog        24
21      Man        30
22      Man        27
23      Man        30
24      Man        33
25      Man        24
26      Man        15
27      Man        12
28      Man        30
29      Man        21
30      Man        54
31      Man        30
32      Man        18
33      Man        39
34      Man        30
35      Man        27
36      Man        24
37      Man        39
38      Man        18
39      Man        15
40      Man        21
```

As you can see, the variable **species** which contains text, has been be imported as a factor. The order of the groups will be alphabetic, for these data this is fine because it makes sense to have **Dog** as the baseline category.

We can have a look at some descriptive statistics by executing:

```
 by(MenDogs[,2], MenDogs$species, stat.desc, basic=FALSE, norm=TRUE)
```

```
MenDogs$species: Dog
median  mean    SE.mean CI.mean.0.95  var     std.dev  coef.var skewness  skew.2SE
24.00   28.05   2.46    5.14          120.58  10.98    0.39     0.63      0.61
------------------------------------------------------------------------------
MenDogs$species: Man
median   mean   SE.mean CI.mean.0.95  var     std.dev  coef.var  skewness skew.2SE
27.00    26.85   2.21    4.63         98.03    9.90    0.37      0.80     0.78
```

I have edited the output above to make it clearer and take up less space.

Now we want to run the Wilcoxon test using the *wilcox.test()* function. Because we have the data for different groups stored in a single column, the wilcox*.test()* function is used like the *lm()* function (in other words, like a regression):

```
MenDogsModel<-wilcox.test(behaviou ~ species, data = MenDogs)
MenDogsModel
```

```
        Wilcoxon rank sum test with continuity correction

data:  behaviou by species
W = 205.5, p-value = 0.8918
alternative hypothesis: true location shift is not equal to 0
```

We can also calculate the effect size (make sure that you have executed the function from the book chapter first) by executing:

```
rFromWilcox(MenDogsModel, 40)
```

```
behaviou by species Effect Size, r =  -0.02151555
```

This represents a tiny effect (it is close to zero), which tells us that there truly isn't much difference between dogs and men.

Writing and interpreting the result

We could report something like:

- ✓ Men (*Mdn* = 27) and dogs (*Mdn* = 24) did not significantly differ in the extent to which they displayed dog-like behaviours, *W* = 205.5, *ns*, *r* = −.02.

Task 2

- ✓ There's been much speculation over the years about the influence of subliminal messages on records. To name a few cases, both Ozzy Osbourne and Judas Priest have been accused of putting backward masked messages on their albums that subliminally influence poor unsuspecting teenagers into doing things like blowing their heads off with shotguns. A psychologist was interested in whether backward masked messages really did have an effect. He took the master tapes of Britney Spears' 'Baby One More Time' and created a second version that had the masked message 'deliver your soul to the dark lord' repeated in the chorus. He took this version, and the original, and played one version (randomly) to a group of 32 people. He took the same group six months later and played them whatever version they hadn't heard the time before. So each person heard both the original, and the version with the masked message, but at different points in time. The psychologist measured the number of goats that were sacrificed in the week after listening to each version. It was hypothesized that the backward message would lead to more goats being sacrificed. The data are in the file **DarkLord.dat**. Analyse them with a Wilcoxon signed-rank test.

First of all, as always, we need to load the data:

```
darkLord<-read.delim("DarkLord.dat", header = TRUE)
```

We can have a look at the data, by executing:

```
darkLord
```

```
   message   nomessag
1       11         9
2        8         8
3        7         6
4        4         5
5       10        19
6       11        20
7       16        11
8       10         8
9        8        21
10       2         4
11      11        11
12       9        14
13      11        10
14      11        12
15      17        17
16       9         8
17      14        11
18       6        12
19       9        10
20       7         9
21       6        12
22      11         9
23      15        12
24       8        11
25      11        21
26       5        17
27       8         9
28      11        10
```

```
29       6       12
30      11       11
31       8        7
32       2       12
```

We can see that our data are stored in different columns (**message** and **nomessag**).

We can then have a look at some descriptive statistics using the *stat.desc()* function and executing:

```
stat.desc(darkLord, basic=FALSE, norm=TRUE)
```

```
                message    nomessag
nbr.val        32.0000000  32.0000000
nbr.null        0.0000000   0.0000000
nbr.na          0.0000000   0.0000000
min             2.0000000   4.0000000
max            17.0000000  21.0000000
range          15.0000000  17.0000000
sum           293.0000000 368.0000000
median          9.0000000  11.0000000
mean            9.1562500  11.5000000
SE.mean         0.6271887   0.7751171
CI.mean.0.95    1.2791598   1.5808617
var            12.5877016  19.2258065
std.dev         3.5479151   4.3847242
coef.var        0.3874856   0.3812804
```

```
> stat.desc(darkLord, basic=FALSE, norm=TRUE)
                   message    nomessag
median         9.00000000 11.00000000
mean           9.15625000 11.50000000
SE.mean        0.62718871  0.77511706
CI.mean.0.95   1.27915980  1.58086166
var           12.58770161 19.22580645
std.dev        3.54791511  4.38472422
coef.var       0.38748561  0.38128037
skewness       0.07694431  0.72064177
skew.2SE       0.09282537  0.86937990
kurtosis      -0.15977594 -0.17286102
kurt.2SE      -0.09870373 -0.10678722
normtest.W     0.96119718  0.91376919
normtest.p     0.29609977  0.01413934
```

Next, we can run the Wilcoxon signed-rank test. We can again use the *wilcox.test()* function, but this time because our data are stored in different columns (**message** and **nomessag**) we need to enter the names of the two variables we want to compare rather than a formula, and we need to include the option *paired=TRUE* to tell **R** that the data are paired. (If we don't include this option **R** will do a Wilcoxon rank-sum test.) For these examples, we're also going to include the option *correct = FALSE*, because we do not want a continuity correction. Therefore, to run the analysis execute:

```
darkModel<-wilcox.test(darkLord$message, darkLord$nomessag,  paired = TRUE,
correct= FALSE)
```

We can see the output by executing the model name:

```
darkModel
```

```
Wilcoxon signed rank test

data:  darkLord$message and darkLord$nomessag
V = 111.5, p-value = 0.03627
alternative hypothesis: true location shift is not equal to 0
```

Finally, we can calculate the effect size. If you have executed the function from the book chapter, you can get the effect sizes by inputting the model name and the number of observations into the function:

```
rFromWilcox(darkModel, 64)
```

```
darkLord$message and darkLord$nomessag Effect Size, r =  -0.2617321
```

This represents a medium effect (it is close to Cohen's benchmark of .3), which tells us that the effect of whether or a subliminal message was present was a substantive effect.

Writing and interpreting the result

- ✓ The number of goats sacrificed after hearing the message (*Mdn* = 9) was significantly less than after hearing the normal version of the song (*Mdn* = 11), $p < .05$, $r = -.26$.

Task 3

- ✓ A psychologist was interested in the effects of television programmes on domestic life. She hypothesized that through 'learning by watching', certain programmes might actually encourage people to behave like the characters within them. This in turn could affect the viewer's own relationships (depending on whether the programme depicted harmonious or dysfunctional relationships). She took episodes of three popular TV shows and showed them to 54 couples, after which the couple were left alone in the room for an hour. The experimenter measured the number of times the couple argued. Each couple viewed all three of the TV programmes at different points in time (a week apart) and the order in which the programmes were viewed was counterbalanced over couples. The TV programmes selected were *EastEnders* (which typically portrays the lives of extremely miserable, argumentative, London folk who like nothing more than to beat each other up, lie to each other, sleep with each other's wives and generally show no evidence of any consideration to their fellow humans!), *Friends* (which portrays a group of unrealistically considerate and nice people who love each other oh so very much – but for some reason I love it anyway!), and a National Geographic programme about whales (this was supposed to act as a control). The data are in the file **Eastenders.dat**. Access the file and conduct Friedman's ANOVA on the data.

Load in the data:

```
eastendersData<-read.delim("Eastenders.dat", header = TRUE)

stat.desc(eastendersData, basic = FALSE, norm = TRUE)
```

```
                eastend     friends      whales
median       8.0000000000  6.00000000  7.00000000
mean         8.0370370370  5.64814815  6.01851852
SE.mean      0.3506637043  0.42226421  0.51224744
CI.mean.0.95 0.7033423206  0.84695475  1.02743826
var          6.6401118099  9.62858141 14.16946191
std.dev      2.5768414406  3.10299555  3.76423457
coef.var     0.3206208244  0.54938282  0.62544205
skewness     0.7780513090 -0.26031167 -0.13704621
skew.2SE     1.1986385649 -0.40102702 -0.21112858
kurtosis     0.0452954476 -1.14673646 -0.98347696
kurt.2SE     0.0354483792 -0.89744005 -0.76967259
normtest.W   0.9139935672  0.94256841  0.94299775
normtest.p   0.0008919632  0.01194442  0.01245366
```

To run the Friedman test we simply input the name of our dataframe, but within the *as.matrix()* function, which converts it to a matrix. In this example, we would execute:

```
friedman.test(as.matrix(eastendersData))
```

```
Friedman rank sum test

data:  as.matrix(eastendersData)
Friedman chi-squared = 7.5859, df = 2, p-value = 0.02253
```

The output shows the chi-square test statistic and its associated degrees of freedom (in this case we had three groups so the degrees of freedom are 3 – 1 = 2), and the significance. Therefore, we could conclude that the type of programme watched significantly affected the subsequent number of arguments (because the significance value is less than .05). However, this result doesn't tell us exactly where the differences lie.

As with the Kruskal–Wallis test, there is a function that enables us to compare all groups, or to compare groups to a baseline. This function, *friedmanmc()* requires the data to be in exactly the same format as the *friedman.test()* function and we use it in exactly the same way. Therefore, for the current data we would execute:

```
friedmanmc(as.matrix(eastendersData))
```

The results are pasted below. As with the Kruskal–Wallis test, you need to look at the column labelled *differences*: if this says *TRUE* then the groups differ significantly, but if it says *FALSE* they don't. In this case, we have a clean sweep of significant results.

```
Multiple comparisons between groups after Friedman test
p.value: 0.05
Comparisons
    obs.dif critical.dif difference
1-2    26.0     24.87897       TRUE
1-3    20.5     24.87897      FALSE
2-3     5.5     24.87897      FALSE
```

Writing and interpreting the result

For Friedman's ANOVA we need only report the test statistic (which we saw earlier is denoted by χ^2), its degrees of freedom and its significance. So, we could report something like:

- ✓ The number of arguments that couples had was significantly affected by the programme they had just watched, $\chi^2(2) = 7.59$, $p < .05$.

We need to report the follow-up tests as well:

- ✓ The number of arguments that couples had was significantly affected by the programme they had just watched, $\chi^2(2) = 7.59$, $p < .05$. *Post hoc* tests were used with Bonferroni correction applied. It appeared that watching *EastEnders* significantly affected the number of arguments compared to *Friends (difference* = 26.0). However, the number of arguments was not significantly different after *EastEnders* compared to after the programme about whales *(difference* = 20.5), or after *Friends* compared to the programme about whales (*difference* = 5.5). In all cases, the critical difference (α = .05 corrected for the number of tests) was 24.88. We can conclude that watching *EastEnders* did produce significantly more arguments than watching *Friends*. However, the whale programme didn't produce any substantial reduction in the number of arguments relative to the control programme.

Task 4

- ✓ A researcher was interested in trying to prevent coulrophobia (fear of clowns) in children. She decided to do an experiment in which different groups of children (15 in each) were exposed to different forms of positive information about clowns. The first group watched some adverts for McDonald's in which their mascot Ronald McDonald is seen cavorting about with children going on about how they should love their mums. A second group was told a story about a clown who helped some children when they got lost in a forest (although what on earth a clown was doing in a forest remains a mystery). A third group was entertained by a real clown, who came into the classroom and made balloon animals for the children. A final group acted as a control condition and they had nothing done to them at all. The researcher took self-report ratings of how much the children liked clowns, resulting in a score for each child that could range from 0 (not scared of clowns at all) to 5 (very scared of clowns). The data are in the file **coulrophobia.dat**. Access them and conduct a Kruskal–Wallis test.

Load in the data:

```
clownData<-read.delim("coulrophobia.dat", header = TRUE)
```

The variable **infotype**, which contains text, will be imported as a factor. This is fine except that when we import the data the order of the groups will be alphabetic: Advert, Exposure, None, Story. For reasons that will become apparent, it's useful to have None as our control category. Therefore, we need to reorder the factor levels. We can do this by executing:

```
clownData$infotype<-factor(clownData$infotype, levels =
levels(clownData$infotype)[c(3, 1, 2, 4)])
```

This command uses the *factor()* function to reorder the levels of the **infotype** variable. It re-creates the variable **infotype** in the *clownData* dataframe (*clownData$infotype*) based on itself, but then uses

the *levels()* function to reorder the groups. We simply put the order of the levels that we'd like in the *c()* function, so in this case we have asked for the levels to be ordered 3, 1, 2, 4, which means that the current third group (None) will become the first group, the current first group will become the second group and so on. Having executed this command, our groups will be ordered None, Advert, Exposure, Story.

We can look at some descriptive statistics by executing:

```
by(clownData$beliefs, clownData$infotype, stat.desc, basic=FALSE)
```

```
clownData$infotype: None
      median         mean      SE.mean  CI.mean.0.95       var      std.dev     coef.var
   2.0000000    2.3333333    0.1259882    0.2702177    0.2380952    0.4879500    0.2091214
------------------------------------------------------------------------------------------
clownData$infotype: Advert
      median         mean      SE.mean  CI.mean.0.95    var         std.dev     coef.var
   4.0000000    3.6000000    0.3055050    0.6552432    1.4000000    1.1832160    0.3286711
------------------------------------------------------------------------------------------
clownData$infotype: Exposure
     median      mean       SE.mean      CI.mean.0.95    var         std.dev     coef.var
   1.0000000    1.8666667    0.4349786    0.9329362    2.8380952    1.6846647    0.9024990
------------------------------------------------------------------------------------------
clownData$infotype: Story
      median         mean      SE.mean   CI.mean.0.95  var         std.dev     coef.var
   1.0000000    1.6666667    0.3607752    0.7737858    1.9523810    1.3972763    0.8383658
```

The Kruskal–Wallis test is done using the *kruskal.test()* function. For the current data we would execute:

```
kruskal.test(beliefs ~ infotype, data = clownData)
```

To interpret the Kruskal–Wallis test, it is useful to obtain the mean rank for each group. We can do this by adding a variable called **Ranks** to the dataframe with the *rank()* function:

```
clownData$Ranks<-rank(clownData$beliefs)
```

This command creates a variable **Ranks** in the *clownData* dataframe that is the ranks for the variable **beliefs**. We can then obtain the mean rank for each group using the *by()* function in conjunction with the *mean()* function:

```
by(clownData$Ranks, clownData$infotype, mean)
```

Let's have a look at the output from the Kruskal–Wallis test:

```
Kruskal-Wallis rank sum test

data:  beliefs by infotype
Kruskal-Wallis chi-squared = 17.0581, df = 3, p-value = 0.0006876
```

This shows this test statistic (**R** labels it chi-squared rather than *H*) and its associated degrees of freedom (in this case we had four groups so the degrees of freedom are 3), and the significance (which is less than the critical value of .05). Therefore, we could conclude that the type of information presented to the children about clowns significantly affected their fear ratings of clowns.

```
clownData$infotype: None
[1] 31.33333
------------------------------------------------------------------------------------------
clownData$infotype: Advert
[1] 45.03333
------------------------------------------------------------------------------------------
clownData$infotype: Exposure
[1] 23.76667
------------------------------------------------------------------------------------------
clownData$infotype: Story
[1] 21.86667
```

The output table above tells us the mean rank in each condition. These mean ranks are important later for interpreting any effects.

A nice succinct set of comparisons would be to compare each group against the control:

- Test 1: Advert compared to control
- Test 2: Exposure compared to control
- Test 3: Story compared to control

Fortunately, we can implement this analysis using the *kruskalmc()* function by using the *cont* option. This option takes the form of *cont = 'one-tailed'* or *'two-tailed'* and, if included, will compare all levels against the first. Therefore, the only complication is that we need to make sure that the None group is the first level of the **infotype** factor. Fortunately we thought ahead and made the None group the first level when we loaded the data into **R**. Therefore, to compare each group to the None group (using a two-tailed test) we simply execute:

```
kruskalmc(beliefs ~ infotype, data = clownData, cont = 'two-tailed')
```

```
Multiple comparison test after Kruskal-Wallis, treatment vs control (two-tailed)
p.value: 0.05
Comparisons
               obs.dif critical.dif difference
None-Advert   13.700000     13.57063       TRUE
None-Exposure  7.566667     13.57063      FALSE
None-Story     9.466667     13.57063      FALSE
```

The effect we got seems to mainly reflect the fact that McDonald's adverts significantly increased fear beliefs about clowns relative to controls (which is no surprise given what a creepy weirdo Ronald McDonald is!).

Writing and interpreting the result

For the Kruskal–Wallis test, we need only report the test statistic (denoted by H), its degrees of freedom and its significance. So, we could report something like:

- ✓ Children's fear beliefs about clowns were significantly affected the format of information given to them, $H(3) = 17.06$, $p < .01$).

However, we need to report the follow-up tests as well (including their effect sizes):

- ✓ Children's fear beliefs about clowns were significantly affected the format of information given to them, $H(3) = 17.06$, $p < .01$. Focused comparisons of the mean ranks between groups showed that fear beliefs counts were significantly higher after the adverts compared to the control (*difference* = 13.7). However, fear beliefs were not significantly different after the stories (*difference* = 9.47) or exposure (*difference* = 7.57), relative to the control. In all cases, the critical difference (α = .05 corrected for the number of tests) was 13.57. We can conclude that Ronald McDonald was sufficient to significantly increase fear beliefs about clowns.