Practical 4

Jumping Rivers

In this question, we are going to use a `for` statement to loop} over a large data set and construct some scatter plots. As a note, we could do a lot of this task using facets in **ggplot2**, but it is good practise for your for loop skills and as you'll see in some places using a for loop is benificial. To generate the data, run the following piece of R code

```
data(experiment, package = "jrProgramming")
head(exper)
```

The data frame `exper` represents an experiment, where we have ten treatments: $A, B, \ldots, J$ and measurements at some time points. We want to create a scatter plot of measurement against time, for each treatment type.

1. First we create a scatter plot of one treatment:

   ```
   library("dplyr")
   library("ggplot2")
   treat_a = filter(exper, treat == "A")
   ggplot(treat_a, aes(x = time, y = values)) +
     geom_point()
   ```

2. To generate a scatter-plot for each treatment, we need to iterate over the different treatment types:

   ```
   for(treatment in unique(exper$treat)) {
     group = filter(exper, treat == treatment)
     g = ggplot(group, aes(x = time, y = values)) +
       geom_point()
     print(g)
     readline("Hit return for next plot")
   }
   ```

   - What does `unique(exper$treat)` give?
   - In the `for` loop, what variable is changing? What are it's possible values?
   - What does the `readline()` function do?

Questions

1. We can change the x-axis label using the `xlab()` function:

```
ggplot(group, aes(x = time, y = values)) +
  geom_point() +
  xlab("Time")
```

Use the `ylab()` function to alter the y-axis label.

2. To add a title to a plot we use the `ggtitle()` function, viz:

```
ggplot(treat_a, aes(x = time, y = values)) +
  geom_point() +
  xlab("Time") +
  ylab("Measurement") +
  ggtitle("Treatment")
```

We can combine strings/characters using the `paste()` function,

Rather than have a static title, make the title of each plot display the treatment type.

3. The y-axis range should really be the same in all graphics. Use the `ylim()` function to fix the range, like below, but using better y-limits. **Hint**: Work out the range before the `for` loop.

```
ggplot(treat_a, aes(x = time, y = values)) +
  geom_point() +
  ylim(-100,100)
```

4. At each iteration, use the `message()` function to print the average measurement level across all time points. Look at the message help page for examples of it's use!

5. On each graph, highlight any observations with a blue point if they are larger than the mean + standard deviations or less than the mean - standard deviations. You should be using another `geom_point()`. **Hint**: You don't need `if` statements here. Just subset your data frame and pass this new data frame to the `geom_point()` function, i.e.

```
geom_point(data = outliers, aes(x = time, y = values))
```

6. Suppose we wanted to save individual graphs in a pdf file. We can use the ggsave function like so

```
ggsave(filename = "treatment.pdf", plot = g)
```

To get unique file names, use the `paste0()` function.

7. Put your code, i.e. the `for` loop and plotting commands, in a function which takes the data frame as an argument.

Solutions

Solutions are contained within this package:

```
vignette("solutions4", package = "jrProgramming")
```