Advanced graphics: practical 1

Jumping Rivers

1 Practical 1

To get you familiar with the underlying `ggplot2` concepts, we'll recreate some standard graphics. Some these of plots aren't particularly useful, we are just using them for illustration purposes.

To BEGIN WITH, load the `ggplot2` package

The `ggplot2` package is automatically installed with `jrGgplot2`.

```
library(ggplot2)
```

Next we load the beauty data set:[1]

[1] Details of the beauty data set can be found at the end of this practical.

```
library(jrGgplot2)
data(Beauty)
```

When loading in data, it's always a good idea to carry out a sanity check. I tend to use the commands

```
head(Beauty)
colnames(Beauty)
dim(Beauty)
```

2 Scatter plots

Scatter plots are created using the point geom. Let's start with a basic scatter plot

```
ggplot(data=Beauty) + geom_point(aes(x=age, y=beauty))
```

To save typing, we can also store the plot as a variable:

In this practical, we are creating the plots in a slightly verbose way.

```
g = ggplot(data=Beauty)
g1 = g + geom_point(aes(x=age, y=beauty))
```

To view this plot, type `g1`.

The arguments `x` and `y` are called aesthetics. For `geom_point`, these parameters are required. This particular geom has other aesthetics: `shape`, `colour`, `size` and `alpha`.[2] Here are some things to try out.

[2] These aesthetics are usually available for most geoms.

- Experiment with other aesthetics. For example,

  ```
  g + geom_point(aes(x=age, y=beauty, colour=gender))
  ```

  or

```
g + geom_point(aes(x=age, y=beauty,
                   alpha=evaluation, colour=gender))
```

Some aesthetics, like `shape` must be discrete. So use `shape = factor(tenured)`.

- Are there any differences between numeric values like `tenured` and characters like `gender` for some aesthetics? What happens if you convert `tenured` to a factor in the `colour` aesthetic. For example, `colour = factor(tenured)`.
- What happens if you set `colour` (or some other aesthetic) outside of the `aes` function? For example, compare

  ```
  g + geom_point(aes(x=age, y=beauty, colour="blue"))
  ```

  to

  ```
  g + geom_point(aes(x=age, y=beauty), colour="blue")
  ```

- What happens when you set an aesthetic to a constant value. For example, `colour=2`. What happens if you put this argument outside of the `aes` function?

3 *Box plots*

The box plot geom has the following aesthetics: `x`, `y`, `colour`, `fill`, `linetype`, `weight`, `size` and `alpha`. We can create a basic boxplot using the following commands:

```
g + geom_boxplot(aes(x=gender, y=beauty))
```

Similar to the point geom, we can add in aesthetics:

```
g + geom_boxplot(aes(x=gender, y=beauty,
                             colour=factor(tenured)))
```

Why do you think we have to convert `tenured` to a discrete factor?

As before, experiment with the different aesthetics. For some of the aesthetics, you will need to convert the continuous variables to discrete variables. For example, this will give an error:

```
g + geom_boxplot(aes(x=gender, y=beauty, colour=tenured))
```

while this is OK

```
g + geom_boxplot(aes(x=gender, y=beauty, colour=factor(tenured)))
```

Make sure you play about with the different aesthetics.

4 *Combining plots*

The key idea with `ggplot2` is to think in terms of layers not in terms of plot "types". For example,

In the lectures we will discuss what this means.

```
g + geom_boxplot(aes(x=gender, y=beauty,
                         colour=factor(tenured))) +
  geom_point(aes(x=gender, y=beauty))
```

- What happens to the plot if you swap the order of the `geom_boxplot` and `geom_point` function calls?
- In this case, `geom_point` isn't that great. Try using `geom_jitter`:

  We have a bit too much data for `geom_jitter`, but you get the point.

  ```
  g + geom_boxplot(aes(x=gender, y=beauty,
                           colour=factor(tenured))) +
    geom_jitter(aes(x=gender, y=beauty))
  ```

5 *Bar plots*

The bar geom has the following aesthetics: `x`, `colour`, `fill`, `size`, `linetype`, `weight` and `alpha`. Here is a command to get started:

```
g + geom_bar(aes(x=factor(tenured)))
```

- As before, try different aesthetic combinations. Convert parameters to discrete versions as needed using `factor(...)`.
- Let's get a bit more fancy. First, we round ages to the nearest decade:

  ```
  Beauty$dec = factor(signif(Beauty$age, 1))
  ```

  then plot:

  ```
  g = ggplot(data=Beauty)
  g + geom_bar(aes(x=gender, fill=dec))
  ```

  We can adjust the layout of this bar plot using ggplot's position adjustments. The five possible adjustments are listed in table 1. The **default** adjustment is `stack`

  ```
  g + geom_bar(aes(x=gender, fill=dec),
               position="stack")
  ```

- Try the other adjustments.

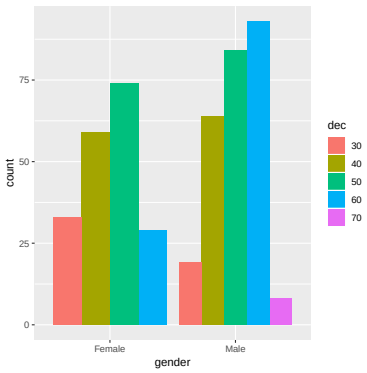

Figure 1: Barplot of ages using `position="dodge"`.

Adjustment	Description
`dodge`	Adjust position by overlapping to the side
`fill`	Stack overlapping elements; standardise stack height
`identity`	Do nothing
`jitter`	Jitter points
`stack`	Stack overlapping elements

Table 1: Position adjustments - table 4.5 in the ggplot2 book.

tenured	minority	age	evaluation	gender	students	beauty
0	1	36	4.3	Female	43	0.202
1	0	59	4.5	Male	20	-0.826
1	0	51	3.7	Male	55	-0.660
1	0	40	4.3	Female	46	-0.766
0	0	31	4.4	Female	48	1.421

Table 2: The first five rows of the beauty data set. There are a total of 463 course evaluations.

6 *The beauty data set*

This data set is from a study where researchers were interested in whether a lecturers' attractiveness affected their course evaluation.[3] This is a cleaned version of the data set and contains the following variables:

[3]

- `evaluation` - the questionnaire result.
- `tenured` - does the lecturer have tenure; 1 == Yes. In R, this value is continuous.
- `minority` - does the lecturer come from an ethnic minority (in the USA).
- `age`.
- `gender`.
- `students` - number of students in the class.
- `beauty` - each of the lecturers' pictures was rated by six undergraduate students: three women and three men. The raters were told to use a 10 (highest) to 1 rating scale, to concentrate on the physiognomy of the professor in the picture, to make their ratings independent of age, and to keep 5 in mind as an average. The scores were then normalised.

Table 2 gives the first few rows of the data set.

Advanced graphics: practical 2

Jumping Rivers

This practical aims to guide you through some of the key ideas in `ggplot2`. As with the first practical, feel free to experiment. Some of the functions introduced in this practical haven't been explicitly covered in the notes. Use the built-in R help or the `ggplot2` help pages at `http://had.co.nz/ggplot2/` as needed.

1 *Over plotting*

Scatter plots are very useful. However, when we have a large data set, points will be plotted on top of each other obscuring the relationship. We call this problem over plotting. There are a few techniques we can use to help, although the best solution is often problem specific.

To begin with we will create an example data frame:

```
## If your computer is slow when plotting reduce the value of n
library("jrGgplot2")
library("ggplot2")
df = overplot_data(n=20000)
```

We can create a simple scatter plot of this data using the following command

```
h = ggplot(df) + geom_point(aes(x, y))
```

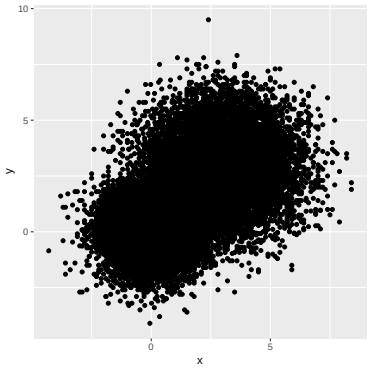

Figure 1: A scatter plot that suffers from over plotting.

This plot isn't particularly good. Try to improve it by using a combination of:

- changing the transparency level: `alpha`;
- change the shape: `shape=1` and `shape='.'`
- use some jittering - `geom_jitter`.
- adding a contour to the plot using `stat_density2d`.
- What does

  ```
  h +  stat_density2d(aes(x,y, fill=..density..),
                  contour=FALSE, geom="tile")
  ```

  do?
- What does `stat_bin2d()` and `stat_binhex()` do - add it to the plot to find out! Try varying the parameters `bins` and `binwidth`.

`alpha` takes a value between 0 and 1.

2 *Displaying distributions*

The diamonds data set contains the prices and other attributes of almost 54,000 diamonds. It is a data frame with 53,940 rows and 10 variables. First, load the diamonds data set:

```
data(diamonds, package="ggplot2")
```

and look at the help file:

```
?diamonds
```

We can construct a histogram of diamond depth using the following commands:

```
i1 = ggplot(data=diamonds) +
  geom_histogram(aes(x=depth))
```

to get figure 2. Let's experiment a bit.

1. Change the `binwidth` in the `geom_histogram`. What value do you think is best?
2. What happens when you set `colour=cut` in the `geom_histogram` aesthetic? What other options can you change?[1]
3. Try `geom_density`. Set `fill=cut` and change the `alpha` value.
4. Try `geom_boxplot`.

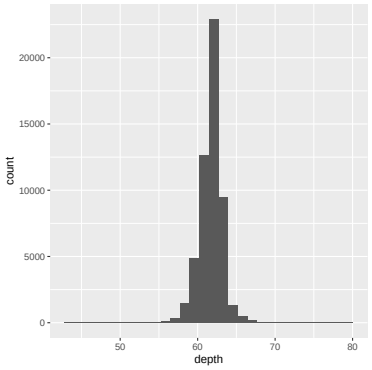

Figure 2: Histogram of the diamond data set.

[1] Look at the `geom_histogram` help page: `http://had.co.nz/ggplot2/geom_histogram.html`

3 *Copy cat*

The aim of this section is to recreate the graphics in figure 3. Feel free to experiment. To begin, load the package

```
library("ggplot2")
```

and the `mpg` data set

```
data(mpg, package="ggplot2")
dim(mpg)
```

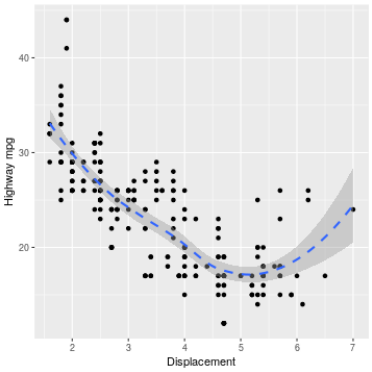

Figure 3: Graphics for section 1.

1. Figure 3: Create a scatter plot of engine displacement, `displ`, against highway mpg, `hwy`. To get started:

   ```
   ggplot(data=mpg, aes(x=displ, y=hwy)) +
     geom_point() + xlab("Displacement")
   ```

   Now add a dashed loess line and change the y-axis label. Hint: try `stat_smooth` and `ylab('New label')`.

2. Figure 4: Using `stat_smooth`, add a loess line conditional on the drive.

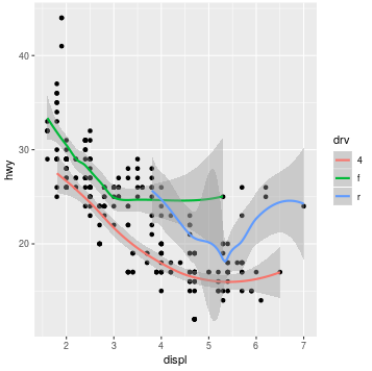

Figure 4: Graphics for section 1.

Solutions

Solutions are contained within this package:

```
library(jrGgplot2)
vignette("solutions2", package="jrGgplot2")
```

Advanced graphics: practical 3

Jumping Rivers

This practical aims to guide you through some of the key ideas in data manipulation. I've tried to construct this practical in such a way that you get to experiment with the various tools. Feel free to experiment!

1 Factors

When using `ggplot2`, the easiest way of rearranging the graph or to alter labels is to manipulate the data set. Consider the `mpg` data set:

```
data(mpg, package = "ggplot2")
```

Suppose we generate a scatter plot of engine displacement against highway mpg.

```
g = ggplot(data=mpg, aes(x=displ, y=hwy)) +
  geom_point()
```

Next, we add a loess line, conditional on the drive type:

```
g + stat_smooth(aes(colour=drv))
```

While this graph is suitable for exploring the data; for publication, we would like to rename the axis and legend labels. To change the axis labels, we can rename the data frame columns or use `xlab` and `ylab`. To change the order of the legend, we need to manipulate the data. Since `drv` is a character, we could use:

```
mpg[mpg$drv == "4",]$drv = "4wd"
mpg[mpg$drv == "f",]$drv = "Front"
mpg[mpg$drv == "r",]$drv = "Rear"
```

However the legend will still be ordered alphabetically. Instead, we can use a factor:

```
##Reload the data just to make sure
data(mpg, package="ggplot2")
mpg$drv = factor(mpg$drv, labels = c("4wd", "Front", "Rear"))
```

To change the order of the, we need to use the `factor` function:

```
mpg$drv = factor(mpg$drv,
                 levels = c("Front", "Rear", "4wd"))
```

The legend now displays the labels in the order: `Front`, `Rear` and `4wd`.

2 Aphids

This data set consists of seven observations on cotton aphid counts on twenty randomly chosen leaves in each plot, for twenty-seven

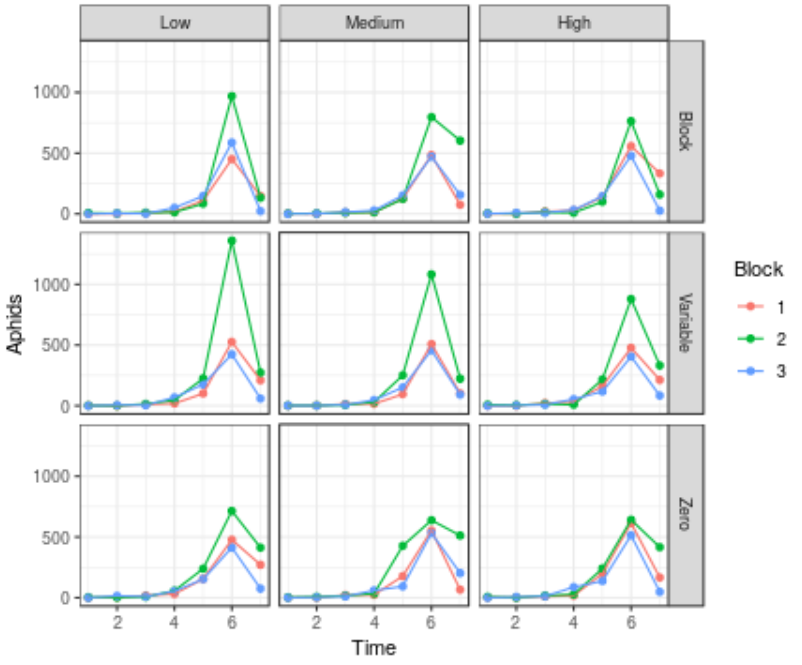

Figure 1: Final figure from section 2.

treatment-block combinations. The data were recorded in July 2004 in Lamesa, Texas. The treatments consisted of three nitrogen levels (blanket, variable and none), three irrigation levels (low, medium and high) and three blocks, each being a distinct area. Irrigation treatments were randomly assigned within each block as whole plots. Nitrogen treatments were randomly assigned within each whole block as split plots.

```
data(aphids, package="jrGgplot2")
```

The sampling times are once per week.

REPRODUCE figure 1. Here are some hints to get you started. The key idea is to think of the plot in terms of layers. So

- Leave the ordering of factors to the end
- The plot contains a combination of `geom_line` and `geom_point`.
- You can change the x-axis label using

```
+ xlab("Time")
```

- Change the theme using `theme_bw()`

3 *The Beauty data set*

First load the beauty data set

```
data(Beauty, package="jrGgplot2")
```

In practical 1, we split data up by both gender and age:

```
Beauty$dec = signif(Beauty$age, 1)
g = ggplot(data=Beauty)
g1 = g + geom_bar(aes(x=gender, fill=factor(dec)))
```

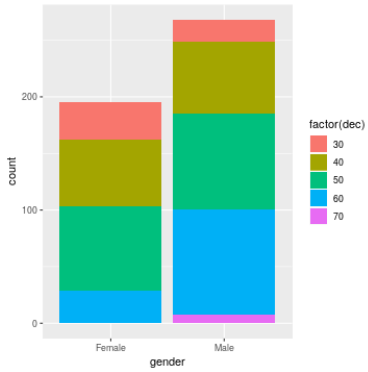

Figure 2: Stacked bar chart of the Beauty data set.

to get figure 2. Rather than using the `fill` aesthetic, redo the plot but use `facet_grid` and `facet_wrap`. For example,

```
g2 = g + geom_bar(aes(x=gender)) +
  facet_grid(. ~ dec)
```

Experiment with:

- the `margins` argument
- the `scales='free_y'` argument
- the layout, i.e. column or row.

How would you change the panel labels?

4 *The Google data set*

Google recently released a data set of the top 1000 websites. The data set contains the following categories: `Rank`, `Site`, `Category`, `Users`, `Views` and `Advertising`. First we load the data

```
data(google, package="jrGgplot2")
```

1. Create a scatter of plot of `Rank` and `Views`.
2. Using `scale_y_log10()` transform the `y` scale.
3. Uses facets to split the plot by its advertising status.
4. Use another `geom_point()` layer to highlight the *Social Networks* sites to figure 3.

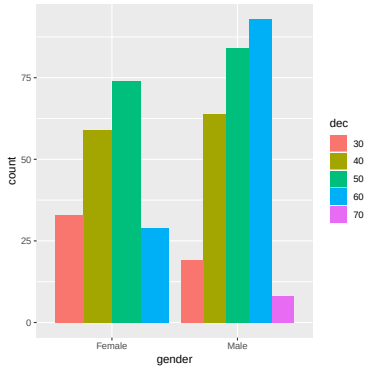

Figure 3: The Google data set.

5 *An example data set*

An experiment was conducted: two treatments, `A` and `B`, were tested. The data can be downloaded into R using the following commands:

```
data(cell_data, package="jrGgplot2")
```

Within each treatment group, there are two patient types: `Case` and `Control`. What plot do you think would be most suitable for this data?

First we'll create a base object

```
##This doesn't plot anything
g = ggplot(cell_data, aes(treatment, values)) +
  facet_grid(.~type)
```

Experiment plotting the data using boxplots, jittered points, histograms, errors, etc. Which methods is optimal (for this data set).

Solutions

Solutions are contained within this package:

```
library(jrGgplot2)
vignette("solutions3", package="jrGgplot2")
```