Recap

Jumping Rivers

We've already covered a fair bit of tidyverse stuff in the Intro to R course. Namely tibbles, **dplyr** and **ggplot2**. This chapter just serves as a general recap into what we touched upon.

tibbles

A tibble (or data frame) is how we store a sheet of data. A standard tibble looks like this

```
data(example, package = "jrTidyverse")
example
```

```
## # A tibble: 4 x 3
##     age gender respond
##   <dbl> <chr>  <lgl>
## 1    24 Male   TRUE
## 2    26 Female FALSE
## 3    25 Male   FALSE
## 4    21 Female FALSE
```

dplyr

dplyr is a package for manipulating tibbles. We covered several functions, such as `filter()` and `summarise()`

```
library("dplyr")
## Give me all the rows where gender is "Male"
filter(example, gender == "Male")
```

```
## # A tibble: 2 x 3
##     age gender respond
##   <dbl> <chr>  <lgl>
## 1    24 Male   TRUE
## 2    25 Male   FALSE
```

```
## What is the average of the age variable
summarise(example, av_age = mean(age))
```

```
## # A tibble: 1 x 1
##   av_age
##    <dbl>
## 1     24
```

1. Filter the example data set so that the rows left have `age` $>$ `24`.
2. Filter the example data set so that the rows left have `respond` $=$ `TRUE`.

We can pass outputs to the first argument of the next function using the piping operator, `%>%`

```
# Give me the average age of males
example %>%
  filter(gender == "Male") %>%
  summarise(av_age = mean(age))
```

```
## # A tibble: 1 x 1
##   av_age
##    <dbl>
## 1   24.5
```

3. Grab the average age of the people who didn't respond

The piping operator can be used in any functions, not just **dplyr**

```
# Pass 1:5 on the left as the first argument to mean
1:5 %>%
  mean(na.rm = TRUE)
```

```
## [1] 3
```

```
# Explicitly pass 1:5 into the function
mean(1:5, na.rm = TRUE)
```

```
## [1] 3
```

We can apply functions to groups within variables using `group_by()`

```
# Give me the average age of each group within gender
example %>%
  group_by(gender) %>%
  summarise(av_age = mean(age))
```

```
## # A tibble: 2 x 2
##   gender av_age
##   <chr>   <dbl>
## 1 Female   23.5
## 2 Male     24.5
```

5. Grab the average age of each grouping within respond

ggplot2

ggplot2 is a fantastic package for graphics. The `ggplot()` function creates a **ggplot2** object.

```
library("ggplot2")
data(movies, package = "jrTidyverse")
ggplot(movies)
```

To add axes to this we add aesthetics

```
# visible in figure 2.1
g = ggplot(movies, aes(x = duration, y = rating))
g
```

Notice we can save plots as variables.

Then to add information onto the graph we use geoms

```
# figure 2.2
g +
  geom_point()
ggplot(movies, aes(x = rating)) +
  geom_histogram()
ggplot(movies, aes(x = classification)) +
  geom_bar()
```

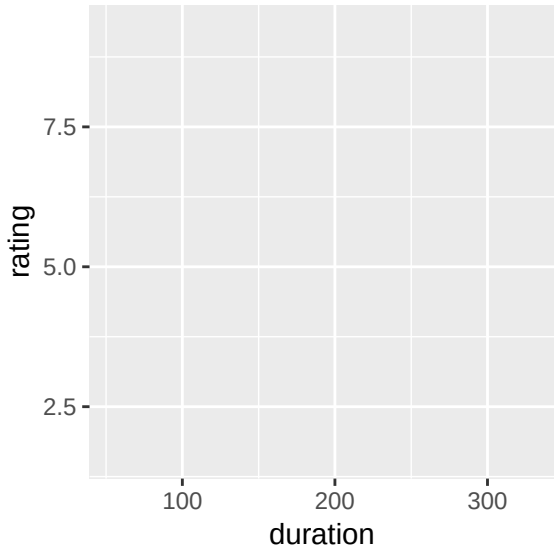

Figure 1: Initial ggplot object

```
ggplot(movies, aes(x = classification, y = rating)) +
  geom_boxplot())
```

1. Using **dplyr**, work out the average rating for each year of movies in the data set.
2. Use **ggplot2** to plot a line graph of the year against average rating. **Hint:** Use `geom_line()`
3. Change the axis labels and titles to something more sensible. Remember, to change labels and titles, we use `labs()`

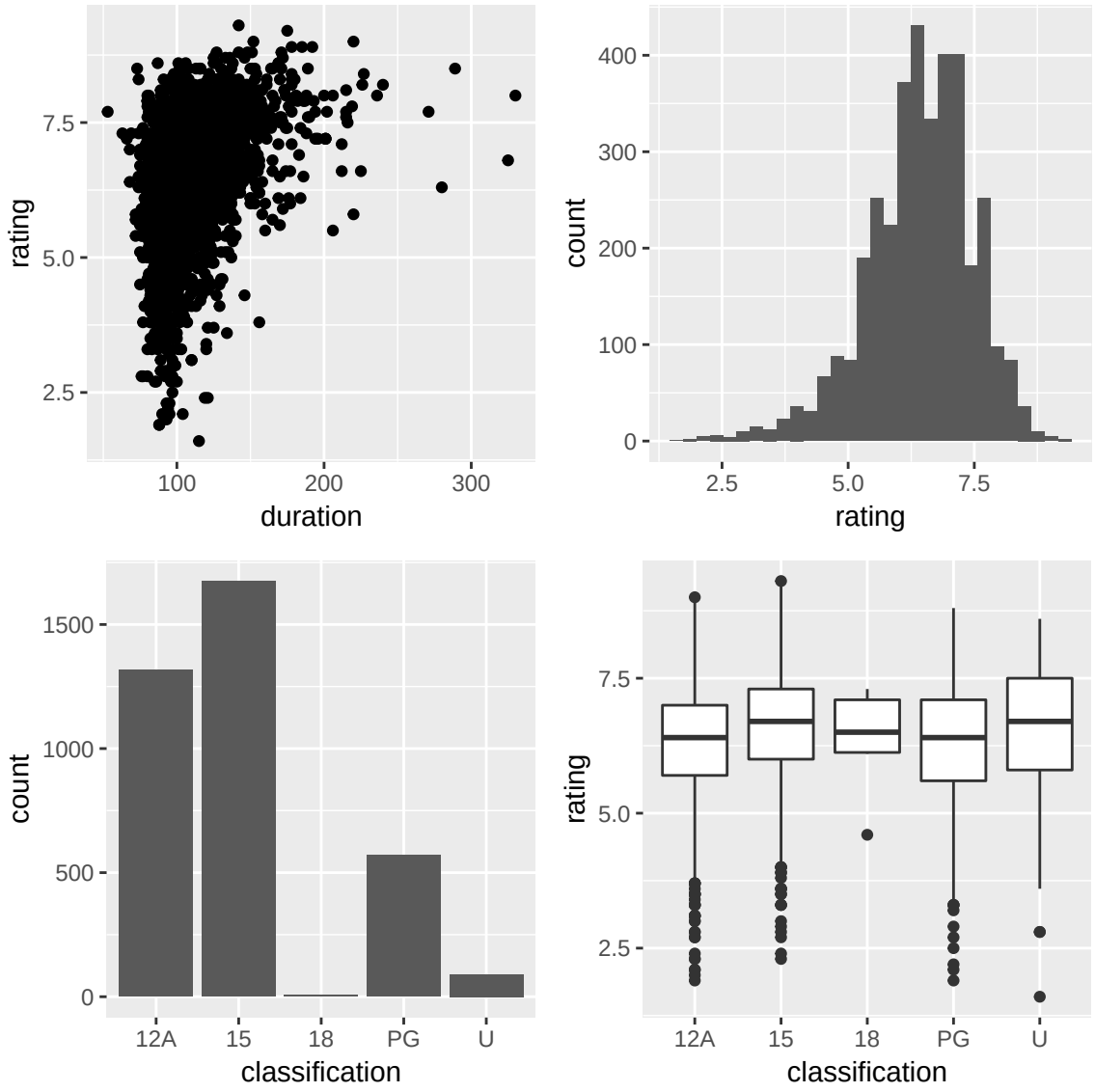

Figure 2: Examples of different geoms