dplyr Practical

Jumping Rivers

We'll start by loading the necessary packages and data sets

```
library("dplyr")
library("ggplot2")
data(okcupid, package = "jrTidyverse")
```

Summarising the data

In this section, we will gradually chain the commands together. We'll start things off, by calculating the average income

```
new_data = okcupid %>%
  summarise(ave_income = mean(income))
new_data
```

1. Alter the above command to calculate the median income (as well as the mean).
2. Use the `group_by()` to calculate the average incomes conditional on the answer to the `pets` question.
3. The `arrange()` function is used to sort a tibble, .e.g

   ```
   ... %>%
     arrange(ave_income)
   ```

   will arrange the tibble from smallest to largest. Arrange the tibble from **largest** to smallest in terms of average income.

Creating columns with `mutate()`

1. The `floor()` function rounds down to the nearest integer. To round to the nearest 10, we use the trick

   ```
   floor(61/10)*10
   floor(119/10)*10
   ```

   We can use the `mutate()` function to create a new column that contains the persons age (to the decade), i.e. 50, 60, 70, etc. The `mutate()` function isn't directly in the notes, but it is relatively easy to understand. It creates a new column with the given name, based on manipulation of existing columns. So we could create this new column `decade`.

   ```
   okcupid %>%
     mutate(decade = floor(age/10) * 10)
   ```

2. Since this data set has high earners, use `filter()` to remove the top 5% of earners. Hint: `quantile(income, probs = 0.95)` will give you the 95%-tile of income
3. To help with plotting, convert the `decade` column into a character using the `as.character()` function. This can be achieved via `mutate(decade = as.character(decade))`