dplyr Practical Solutions

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
```

```
## # A tibble: 1 x 1
##   ave_income
##        <dbl>
## 1    104395.
```

1. Alter the above command to calculate the median income (as well as the mean).

```
okcupid %>%
  summarise(ave_income = mean(income),
            med_income = median(income))
```

```
## # A tibble: 1 x 2
##   ave_income med_income
##        <dbl>      <dbl>
## 1    104395.     50000.
```

2. Use the `group_by()` to calculate the average incomes conditional on the answer to the `pets` question.

```
okcupid %>%
  group_by(pets) %>%
  summarise(ave_income = mean(income))
```

```
## # A tibble: 16 x 2
##    pets                              ave_income
##    <chr>                                  <dbl>
##  1 dislikes cats                        159500.
##  2 dislikes dogs                         66154.
##  3 dislikes dogs and dislikes cats      176154.
##  4 dislikes dogs and has cats            93953.
##  5 dislikes dogs and likes cats         103956.
##  6 has cats                              84498.
##  7 has dogs                             112540.
##  8 has dogs and dislikes cats           104895.
##  9 has dogs and has cats                 97995.
## 10 has dogs and likes cats               87432.
## 11 likes cats                            69234.
```

```
## 12 likes dogs                          119483.
## 13 likes dogs and dislikes cats         99667.
## 14 likes dogs and has cats              90905.
## 15 likes dogs and likes cats           106814.
## 16 <NA>                                106222.
```

4. The `arrange()` function is used to sort a tibble, .e.g

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

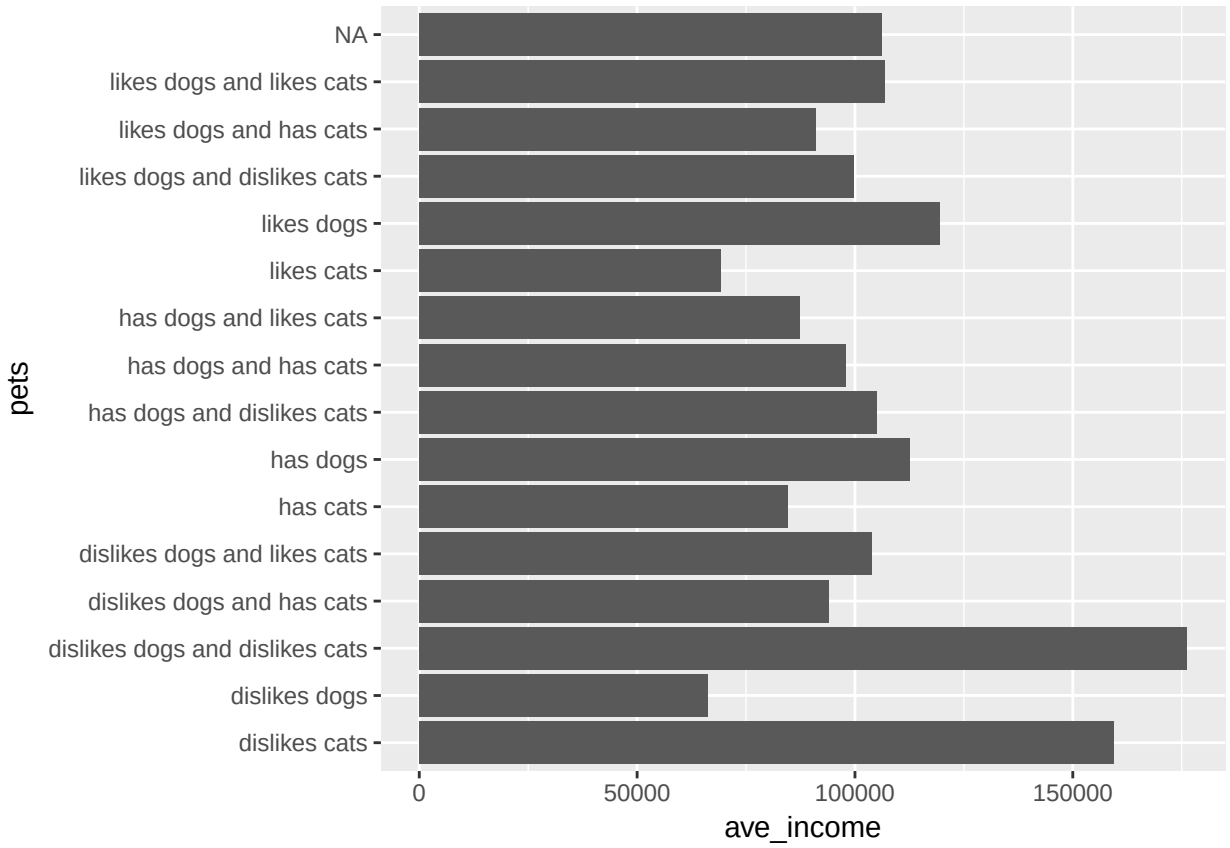
will arrange the tibble from smallest to largest. Arrange the tibble from **largest** to smallest in terms of average income.

```
(df = okcupid %>%
  group_by(pets) %>%
  summarise(ave_income = mean(income)) %>%
  arrange(desc(ave_income))
)
```

```
## # A tibble: 16 x 2
##    pets                             ave_income
##    <chr>                                 <dbl>
##  1 dislikes dogs and dislikes cats    176154.
##  2 dislikes cats                      159500.
##  3 likes dogs                         119483.
##  4 has dogs                           112540.
##  5 likes dogs and likes cats          106814.
##  6 <NA>                               106222.
##  7 has dogs and dislikes cats         104895.
##  8 dislikes dogs and likes cats       103956.
##  9 likes dogs and dislikes cats        99667.
## 10 has dogs and has cats               97995.
## 11 dislikes dogs and has cats          93953.
## 12 likes dogs and has cats             90905.
## 13 has dogs and likes cats             87432.
## 14 has cats                            84498.
## 15 likes cats                          69234.
## 16 dislikes dogs                       66154.
```

5. Using **ggplot2** and `geom_col()` plot your results. Hint use `+ coord_flip()` to rotate your plot.

```
ggplot(df) +
  geom_col(aes(x = pets, y = ave_income)) +
  coord_flip()
```



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
```

```
## [1] 60
```

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

```
## [1] 110
```

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)
```

```
## # A tibble: 11,504 x 22
##      age body_type diet   drinks  drugs education  ethnicity height income
##    <int> <chr>     <chr>  <chr>   <chr> <chr>      <chr>      <int>  <int>
##  1    35 average   mostl~ often   some~ working o~ white         70 8.00e4
##  2    23 thin      veget~ social~ <NA>  working o~ white         71 2.00e4
##  3    28 average   mostl~ social~ never graduated~ white         72 4.00e4
##  4    30 skinny    mostl~ social~ never graduated~ white         66 3.00e4
##  5    29 thin      mostl~ social~ never working o~ hispanic~     62 5.00e4
##  6    40 fit       <NA>   social~ <NA>  graduated~ white         71 6.00e4
```

```
##  7    31 thin      stric~ social~ some~ dropped o~ <NA>          67 1.00e6
##  8    22 athletic  mostl~ rarely  never working o~ asian         65 2.00e4
##  9    35 athletic  mostl~ social~ some~ graduated~ native a~     73 1.50e5
## 10    31 curvy     mostl~ social~ never graduated~ indian        61 5.00e4
## # ... with 11,494 more rows, and 13 more variables: job <chr>,
## #   last_online <dttm>, location <chr>, offspring <chr>,
## #   orientation <chr>, pets <chr>, religion <chr>, sex <chr>, sign <chr>,
## #   smokes <chr>, speaks <chr>, status <chr>, decade <dbl>
```

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

```
okcupid %>%
  mutate(decade = floor(age/10)*10) %>%
  filter(income < quantile(income, probs = 0.95))
```

```
## # A tibble: 10,786 x 22
##      age body_type diet   drinks  drugs education  ethnicity height income
##    <int> <chr>     <chr>  <chr>   <chr> <chr>      <chr>      <int>  <int>
##  1    35 average   mostl~ often   some~ working o~ white         70  80000
##  2    23 thin      veget~ social~ <NA>  working o~ white         71  20000
##  3    28 average   mostl~ social~ never graduated~ white         72  40000
##  4    30 skinny    mostl~ social~ never graduated~ white         66  30000
##  5    29 thin      mostl~ social~ never working o~ hispanic~     62  50000
##  6    40 fit       <NA>   social~ <NA>  graduated~ white         71  60000
##  7    22 athletic  mostl~ rarely  never working o~ asian         65  20000
##  8    35 athletic  mostl~ social~ some~ graduated~ native a~     73 150000
##  9    31 curvy     mostl~ social~ never graduated~ indian        61  50000
## 10    21 fit       mostl~ rarely  never working o~ white         71  20000
## # ... with 10,776 more rows, and 13 more variables: job <chr>,
## #   last_online <dttm>, location <chr>, offspring <chr>,
## #   orientation <chr>, pets <chr>, religion <chr>, sex <chr>, sign <chr>,
## #   smokes <chr>, speaks <chr>, status <chr>, decade <dbl>
```

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))`

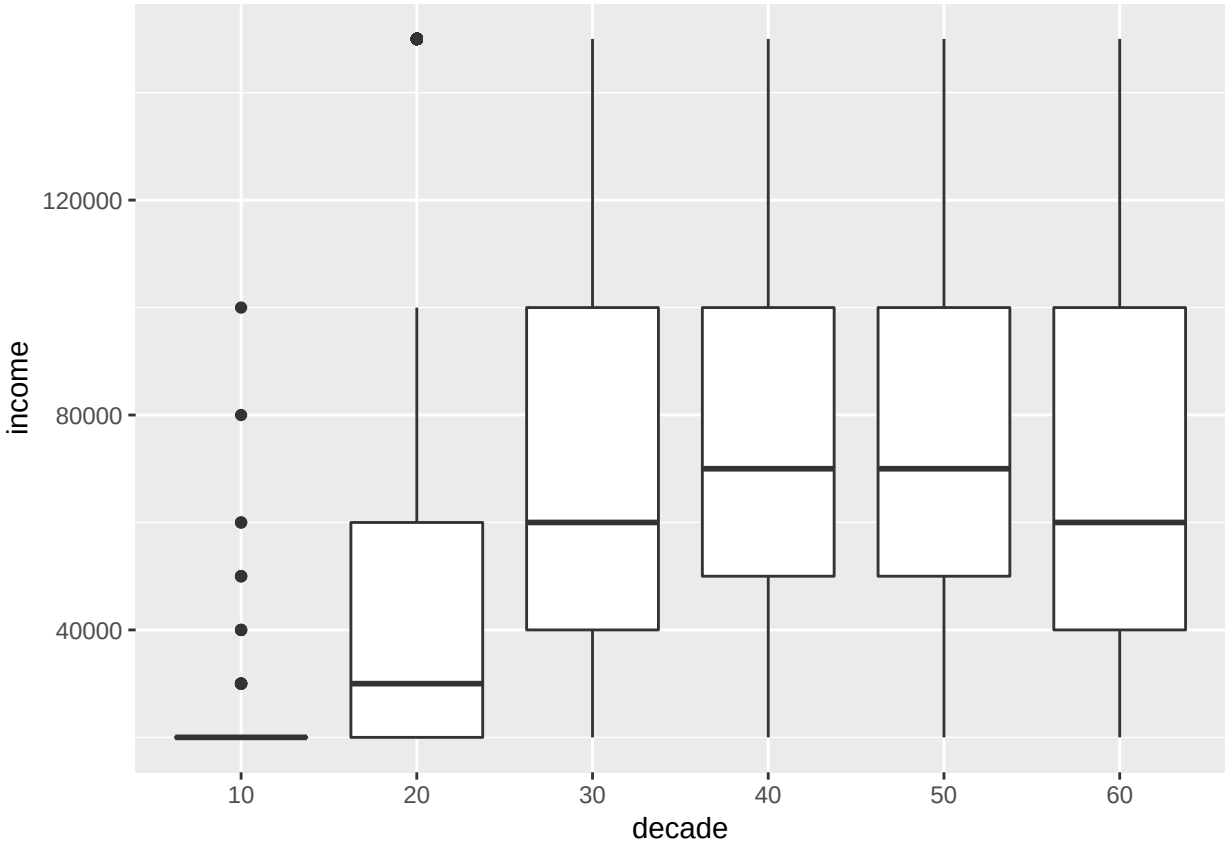
```
(df = okcupid %>%
  mutate(decade = floor(age/10)*10) %>%
  filter(income < quantile(income, probs = 0.95)) %>%
  mutate(decade = as.character(decade))
)
```

```
## # A tibble: 10,786 x 22
##      age body_type diet   drinks  drugs education  ethnicity height income
##    <int> <chr>     <chr>  <chr>   <chr> <chr>      <chr>      <int>  <int>
##  1    35 average   mostl~ often   some~ working o~ white         70  80000
##  2    23 thin      veget~ social~ <NA>  working o~ white         71  20000
##  3    28 average   mostl~ social~ never graduated~ white         72  40000
##  4    30 skinny    mostl~ social~ never graduated~ white         66  30000
##  5    29 thin      mostl~ social~ never working o~ hispanic~     62  50000
##  6    40 fit       <NA>   social~ <NA>  graduated~ white         71  60000
##  7    22 athletic  mostl~ rarely  never working o~ asian         65  20000
##  8    35 athletic  mostl~ social~ some~ graduated~ native a~     73 150000
##  9    31 curvy     mostl~ social~ never graduated~ indian        61  50000
## 10    21 fit       mostl~ rarely  never working o~ white         71  20000
```

```
## # ... with 10,776 more rows, and 13 more variables: job <chr>,
## #   last_online <dttm>, location <chr>, offspring <chr>,
## #   orientation <chr>, pets <chr>, religion <chr>, sex <chr>, sign <chr>,
## #   smokes <chr>, speaks <chr>, status <chr>, decade <chr>
```

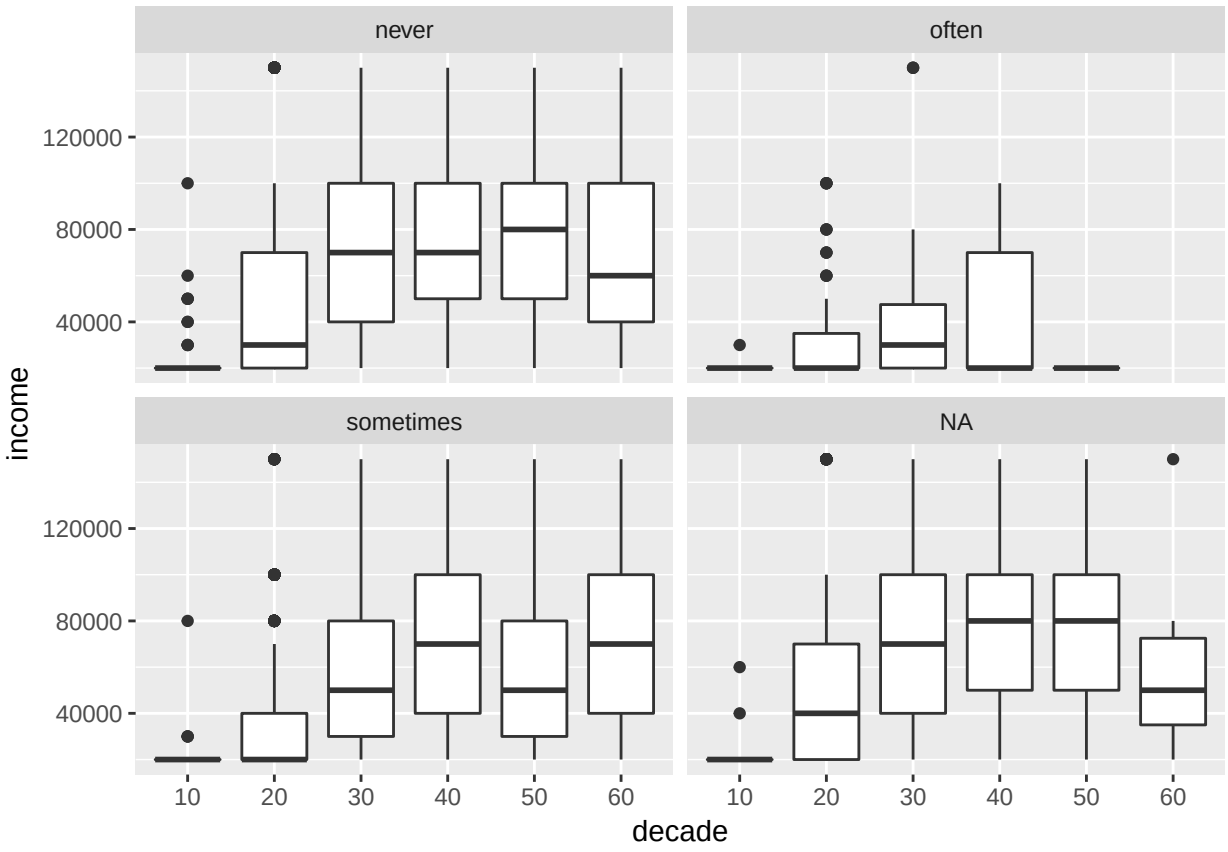
4. Use **ggplot2** to create boxplots of `x = decade` and `y = income`.

```
ggplot(df) +
  geom_boxplot(aes(x = decade, y = income))
```



5. Create facets by using `+ facet_wrap(~ drugs)`

```
ggplot(df) +
  geom_boxplot(aes(x = decade, y = income)) +
  facet_wrap(~drugs)
```


never
often
sometimes
NA
income
120000
80000
40000
10
20
30
40
50
60
decade