tidyr Practical Solutions

Jumping Rivers

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

tidyr: Getting started with `separate()`

The original state of the okcupid data has numerous messy columns. Let's tidy some of them up. First, the location variable currently stores both the area and the state. We can separate it into two further variables using `separate()`

```
okcupid = okcupid %>%
  separate(location, c("area", "state"), sep = ", ")
```

Notice the warning, R is just telling us that in one of the rows there was two commas and so there were three pieces of information. In this case it was the inclusion of country information on top of the area and state. Next up, ethnicity. In some cases people have listed 3 ethnicities per person. We are only interested in the first one (obviously for actual analysis this is not recommended as it a gross under representation of todays multi ethnic society). Again we can use separate to rid of everything after first ethnicity

```
okcupid = okcupid %>%
  separate(ethnicity, "ethnicity", sep = ", ")
```

1. How many people put poor english as their first language (Hint: use `separate()` to separate the `speaks` variable into 3 different columns then use `count()`)

```
okcupid = okcupid %>%
  separate(speaks, c("first_lan", "sec_lan", "third_lan"), sep = ", ")
okcupid %>%
  count(first_lan)
```

```
## # A tibble: 5 x 2
##   first_lan              n
##   <chr>              <int>
## 1 english             4901
## 2 english (fluently)  6195
## 3 english (okay)       235
## 4 english (poorly)     168
## 5 <NA>                   5
```

```
# 168
```

2. How many people put the programming language c++ as their second language? Hint (apply another `separate()` operation on the column `sec_lan`)

```
okcupid = okcupid %>%
  separate(sec_lan, "second_lan", sep = " ")
```

```
okcupid %>%
  count(second_lan) %>%
  filter(second_lan == "c++")
```

```
## # A tibble: 1 x 2
##   second_lan     n
##   <chr>      <int>
## 1 c++          200
```

3. Use `separate()` to extract a persons religion, given that a persons religion is always stated as the first word in the column "religion".

```
okcupid = okcupid %>%
  separate(religion, c("religion"), sep = " ")
```

4. Which religion is the most commmon?

```
okcupid %>%
  count(religion)
```

```
## # A tibble: 10 x 2
##    religion         n
##    <chr>        <int>
##  1 agnosticism   1905
##  2 atheism       1658
##  3 buddhism       445
##  4 catholicism   1029
##  5 christianity  1452
##  6 hinduism        91
##  7 islam           36
##  8 judaism        459
##  9 other         1944
## 10 <NA>          2485
```

Using `spread()`

Long format data is great for us as R programmers as it is a convenient format for lots of things that we wish to do with our data. However it is not always the most useful way to share a table with others. An example of this might be as follows.

The code

```
(df = okcupid %>%
   group_by(sex) %>%
   count(orientation)
 )
```

```
## # A tibble: 6 x 3
## # Groups:   sex [2]
##   sex   orientation     n
##   <chr> <chr>       <int>
## 1 f     bisexual      395
## 2 f     gay           243
## 3 f     straight     2475
## 4 m     bisexual      226
## 5 m     gay           573
## 6 m     straight     7592
```

creates a data structure that works well for **ggplot** but is more difficult to read in a report.

1. Use `spread()` to give a column for each orientation, filled with the counts. Does it look a bit nicer?

```r
df %>%
  spread(orientation, n)
```

```
## # A tibble: 2 x 4
## # Groups:   sex [2]
##   sex   bisexual   gay straight
##   <chr>    <int> <int>    <int>
## 1 f          395   243     2475
## 2 m          226   573     7592
```