Practical 2: Solutions

Jumping Rivers

First we must load the **tidyverse**

```
library("tidyverse")
```

Question 1 - Cocktails!

We've put together a small list containing the ingredients of some classic cocktails

```
data(cocktails, package = "jrTidyverse2")
```

a) How many cocktails are in the list?

```
str(cocktails)
## List of 6
##  $ pina_colada         : chr [1:3] "rum" "coconut cream" "pineapple"
##  $ bloody_mary         : chr [1:7] "vodka" "tomato juice" "lemon" "tabasco sauce" ...
##  $ long_island_iced_tea: chr [1:7] "vodka" "gin" "tequila" "rum" ...
##  $ espresso_martini    : chr [1:3] "vodka" "espresso" "coffee liqueur"
##  $ zombie              : chr [1:6] "rum" "orange liqueur" "apricot brandy" "orange juice" ...
##  $ margarita           : chr [1:3] "tequila" "orange liqueur" "lime juice"
```

b) Create a tibble called `drinks`, where one column contains the name of the cocktail, and the other column contains the vector of ingredients

```
drinks = tibble(cocktail = names(cocktails),
                ingredients = cocktails)
```

c) Create a new column that contains the number of ingredients in each column using `mutate()` and **purrr**

```
drinks = drinks %>%
  mutate(total_ingredients = map(ingredients, length))
```

d) We're off out! Tonight we're particularly thirsty for a cocktail with rum in it. Filter `drinks` such that it only has cocktails containing rum

```
drinks %>%
  mutate(contains_rum = map_lgl(ingredients,
                                ~ any(.x == "rum"))) %>%
  filter(contains_rum)
## # A tibble: 3 x 4
##   cocktail             ingredients total_ingredients contains_rum
##   <chr>                <list>      <list>            <lgl>
## 1 pina_colada          <chr [3]>   <int [1]>         TRUE
## 2 long_island_iced_tea <chr [7]>   <int [1]>         TRUE
## 3 zombie               <chr [6]>   <int [1]>         TRUE
```

Question 2 - Beer !

So, we're at the pub with 8 mates and it's your round. In total you've been tasked with ordering 4 ales, 3 ipas, 1 stout plus an ale for yourself! We can load a data set of all of the ale, ipa and stouts that the pub sells from the course package

```
(data(beer_tidy, package = "jrTidyverse2"))
## [1] "beer_tidy"
```

We're going to randomly select each persons drink using **purrr**. If people had asked for an even number of ales, ipas and stouts we could have done this without **purrr** like so

```
beer_tidy %>%
  group_by(Type) %>%
  sample_n(3)
## # A tibble: 9 x 4
## # Groups:   Type [3]
##   URL                           ABV Color Type
##   <chr>                       <dbl> <dbl> <chr>
## 1 Peanut Butter Brown Ale      5.7  29.8  Ale
## 2 Dry City American Amber Ale  5.97 10.6  Ale
## 3 Brown Ale                    6.83 20.8  Ale
## 4 3 Floods Thai Ipa            6.08  9.15 Ipa
## 5 Ford Ipa                     5.47  8.4  Ipa
## 6 Transplanted Oregonian Ipa   7.74 12.8  Ipa
## 7 Cm Foreign Extra Stout       6.15 35.6  Stout
## 8 Imperial Stout               9.8  50    Stout
## 9 Chocolate Milk Stout         5.77 50    Stout
```

a) Nest the data according to the drink `Type` and save it as `pub`.

```
pub = beer_tidy %>%
  nest(-Type)
```

b) Create a column called `n` that contains the total number of each drink `Type` you need to order

```
pub = pub %>%
  mutate(n = c(5, 3, 1))
```

c) Create a new column called order that contains the randomly sampled drinks you are going to order. You should be using `map2()` to parallel map over the columns `data` and `n`. You should also be using `sample_n()` to perform the sampling.

```
pub = pub %>%
  mutate(order = map2(data, n, sample_n))
# or
# mutate(order = map2(.x = data, .y = n, ~sample_n(.x, .y)))
```

d) To see the drinks, select only the `Type` and `order` column, then `unnest()`

```
pub %>%
  select(Type, order) %>%
  unnest(cols = order)
## # A tibble: 9 x 4
##   Type  URL                          ABV Color
##   <chr> <chr>                      <dbl> <dbl>
## 1 Ale   Lucky Jack Pale Ale         5.51  6.38
## 2 Ale   Bas Ale                     5.36  9.45
```

```
## 3 Ale   Holy Hunk Pale Ale            5.78  9.39
## 4 Ale   Amber Ale                     4.81 13.5
## 5 Ale   Blonde Ale Centennial Blonde  4.5   4.28
## 6 Ipa   Double Ipa                    7.11  7.94
## 7 Ipa   Rye Ipa                       7.91 10.5
## 8 Ipa   Uk Rye Ipa                    6.95  7.73
## 9 Stout Dry Irish Stout               6.06 39.8
```

Question 3 - Happiness

You may remember the happiness data we used for practical 1 was recorded over 3 years; 2015, 2016 and 2017. For this question I've turned the happiness list in 3 tibbles, with each one representing the year. Running the following code will copy each file into your current working directory as a `.csv` file

```
library("jrTidyverse2")
get_happiness()
## Files have been copied successfully!
##           Check your current working directory.
```

a) Using a combination of **purrr** and the `unnest()` function from **tidyr**, read in and combine the 3 data sets. Don't delete the column containing the file name!

```
fnames = list.files("happiness", recursive = TRUE, full.names = TRUE)
happiness = tibble(fname = fnames) %>%
  mutate(data = map(fnames, read_csv)) %>%
  unnest()
```

b) The data within the csv files doesn't contain the year. Fortunately the file name does! Use the column containing the filename to create a column called `Year`. Have a look at the `str_remove()` or `parse_number()` functions from **stringr** and **readr** respectively

```
happiness = happiness %>%
  mutate(Year = parse_number(fname)) %>%
  select(-fname)
# or
# happiness = happiness %>%
#   mutate(Year = str_remove(fname, ".csv"),
#          Year = as.numeric(Year)) %>%
#   select(-fname)
```

c) Pick 3 countries and plot their happiness rank over time.

```
happiness %>%
  filter(Country %in% c("Denmark", "United Kingdom", "United States")) %>%
  ggplot(aes(x = Year, y = `Happiness Rank`, colour = Country)) +
  geom_line() +
  geom_point()
```

d) Every country in the data set has requested that they have their data sent to them individually. Make use of **purrr**'s parallel mapping functions and `nest()` to write the data to `.csv` files

```
happiness_nest = happiness %>%
  nest(cols = -Country)
map2(happiness_nest$Country, happiness_nest$data,
     ~write_csv(.y, path = paste0(.x, ".csv"))
```