purrr solutions

Jumping Rivers

First we must load the **tidyverse**

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

Question 1

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

a) `l` is a list with 3 vector elements; `A`, `B`, `C`. Each vector contains the heights of a different class at a university.

- How many students are in each class?
- Which class has the highest average height?
- Which class has the smallest student?
- Which class has the tallest student?
  Hint: use `map()`, `map_dbl()`, or `map_df()`

```
map_dbl(l, length)
##  A  B  C
## 28 30 24
map_dbl(l, mean)
##        A        B        C
## 176.4357 168.9733 181.8833
map_dbl(l, min)
##     A     B     C
## 154.1 159.1 154.7
map_dbl(l, max)
##     A     B     C
## 199.8 176.8 215.6

### OR

map_df(l, ~tibble(length = length(.x),
               mean = mean(.x),
               min = min(.x),
               max = max(.x)))
## # A tibble: 3 x 4
##   length  mean   min   max
##    <int> <dbl> <dbl> <dbl>
## 1     28  176.  154.  200.
## 2     30  169.  159.  177.
## 3     24  182.  155.  216.
```

b) For each class, work out the lower and upper confidence interval bounds for the mean given that they are $\mu - 1.96\sigma$ and $\mu + 1.96\sigma$ respectively. Where μ is the sample mean and σ is the sample standard deviation.
Hint: use the formula notation and either `map()`, `map_dbl()` or `map_df()`.

```
map_df(l, ~tibble(lower = mean(.x) - 1.96*sd(.x),
                  upper = mean(.x) + 1.96*sd(.x)))
## # A tibble: 3 x 2
##   lower upper
##   <dbl> <dbl>
## 1  155.  198.
## 2  160.  178.
## 3  144.  220.
```

Question 2

Now we're going to look at a list containing happiness rankings for countries around the globe.

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

a) How long is the list? Is this a recursive list? How many countries does the list contain information on? For each country how many pieces of information is there?
Hint: use `str()`

```
str(happiness, max.level = 0)
## List of 146
# 146 element in the list
str(happiness, max.level = 1, list.len = 3)
## List of 146
##  $ :List of 12
##  $ :List of 12
##  $ :List of 12
##   [list output truncated]
# Yes, recursive list
str(happiness, max.level =2, list.len = 3)
## List of 146
##  $ :List of 12
##   ..$ Country                      : chr "Switzerland"
##   ..$ Region                       : chr "Western Europe"
##   ..$ Year                         : num [1:3] 2015 2016 2017
##   .. [list output truncated]
##  $ :List of 12
##   ..$ Country                      : chr "Iceland"
##   ..$ Region                       : chr "Western Europe"
##   ..$ Year                         : num [1:3] 2015 2016 2017
##   .. [list output truncated]
##  $ :List of 12
##   ..$ Country                      : chr "Denmark"
##   ..$ Region                       : chr "Western Europe"
##   ..$ Year                         : num [1:3] 2015 2016 2017
##   .. [list output truncated]
##   [list output truncated]
# Each element of the list is another list representing a country.
# Each list contains elements representative of happiness information
# on that country for three successive years. Therefore there is
# 146 countries and 12 vectors of information on each.
```

b) Return the name of each country contained in the list. To make it a bit easier to read return the output as a character vector.

```
country_names = map_chr(happiness, "Country")
```

c) Try `names(happiness)`, what happens? Use the answer to `b)` to rename each element of the list after it's representative country.

```
names(happiness) = country_names
```

d) What has the UKs average happiness rank been over the last 3 years?

```
UK_rank = happiness[["United Kingdom"]]$`Happiness Rank`
mean(UK_rank)
## [1] 21
```

e) Over the last 3 years, what is the average happiness score for every country? Store this in a vector of doubles.

```
mean_hap = happiness %>%
  map_dbl(~ mean(.x[["Happiness Score"]]))
head(mean_hap)
## Switzerland     Iceland     Denmark      Norway      Canada     Finland
##    7.530000    7.522000    7.525000    7.519000    7.382333    7.429333
```

f) Which region of the world has the high average happiness score?
Hint: store the region for each country in a vector, combine it into a data frame with the average happiness then use **dplyr**.

```
region = happiness %>%
  map_chr("Region")

region_hap = data.frame(region = region, mean_hap = mean_hap)

region_hap %>%
  group_by(region) %>%
  summarise(av_region_hap = mean(mean_hap)) %>%
  arrange(av_region_hap)
## # A tibble: 10 x 2
##    region                          av_region_hap
##    <fct>                                   <dbl>
##  1 Sub-Saharan Africa                       4.11
##  2 Southern Asia                            4.59
##  3 Middle East and Northern Africa          5.36
##  4 Central and Eastern Europe               5.37
##  5 Southeastern Asia                        5.40
##  6 Eastern Asia                             5.49
##  7 Latin America and Caribbean              6.05
##  8 Western Europe                           6.69
##  9 North America                            7.23
## 10 Australia and New Zealand                7.30
```

g) Using **ggplot2** and `geom_col()`, plot the answer to f) as a bar chart

```
region_hap %>%
  group_by(region) %>%
  summarise(av_region_hap = mean(mean_hap)) %>%
  arrange(av_region_hap) %>%
  ggplot() +
  geom_col(aes(x = region, y = av_region_hap)) +
  coord_flip()
```

Question 3

Load the data and the required packages

```
library("broom")
data(beer_tidy, package = "jrTidyverse2")
```

Here we have a data set of around 500 beers along with their alcohol percentage, colour and type.

```
head(beer_tidy)
## # A tibble: 6 x 4
##   URL                                    ABV Color Type
##   <chr>                                <dbl> <dbl> <chr>
## 1 Deschutes Mirror Pond Pale Ale Clone  5.08  6.58 Ale
## 2 Kiwanda Cream Ale                     5.21  3.08 Ale
## 3 Christmas Ale                         6.82 24.0  Ale
## 4 Strong Scotch Ale Smuttynose Clone    8.96 21.1  Ale
## 5 Creamy Cream Ale                      4.96  3.07 Ale
## 6 Lemon Ale                             5.31  9.14 Ale
```

We're going to run a linear regression on the data testing how colour affects alcohol percentage. However, using **purrr**, we are going to do it for each different type of beer.

1) Nest the beers within each type. Save this data in a variable called `beer_nest`.

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

2) For each type of beer, fit a linear regression model with alcohol perecentage as the response variable and colour as the predictor. Store the models in a list column next to your nested data. To help you out, the below code is how you would fit the model to the entire data set, before nesting.

```
fit = lm(ABV ~ Color, data = beer_tidy)
```

Hint: use `mutate()` and `map()`

```
beer_nest = beer_nest %>%
  mutate(fit = map(data, ~lm(ABV ~ Color, data = .x)))
```

3) For each model, mutate a new column with the tidied models in. We want to plot the points eventually, so we want to tidy the model such that it returns the fitted points.
Hint: use `mutate()`, `map()` and `augment()`

```
beer_nest = beer_nest %>%
  mutate(tidyfit = map(fit, ~augment(.x)))
```

4) Select the columns containing the beer type and the tidied models and unnest the data.

```
beer_nest = beer_nest %>%
  select(Type, tidyfit) %>%
  unnest()
```

5) Given you have stored unnested models in the variable `beer_nest()`, the following code WILL plot the lines made by the linear regression models

```
ggplot(beer_nest) +
  geom_line(aes(x = Color, y = .fitted, colour = Type), size = 2)
```