Exploring data 2

Using functions

Applying a function repeatedly

You will often want to apply a function multiple times with different input. A powerful way to do this in R is using `lists`.

We can use functions from the `purrr` library to `map` functions to each element of a list.

The `map` family of functions includes: `map`, `map_2`, and `pmap`.

Applying a function repeatedly

The `map` function works well if you have a list and you want to apply a function to each element in the list.

For example, say you have the following list:

```
a_list <- list(first_element = 1:3,
               second_element = c(11, 15, 20))
```

Applying a function repeatedly

You want to get the range of numbers in each element of the list. From looking at the list, you should have an idea of the values we'll get if we do this:

```
a_list
```

```
## $first_element
## [1] 1 2 3
##
## $second_element
## [1] 11 15 20
```

Applying a function repeatedly

You can use the `map` function from the `purrr` package to apply this function to each element of the list:

```
library(purrr)
map(.x = a_list, .f = range)
```

```
## $first_element
## [1] 1 3
##
## $second_element
## [1] 11 20
```

Applying a function repeatedly

This works well with data that you have in a "list-column" of a nested dataframe.

Say that you originally had the data in a dataframe, with one column saying whether it's the first or second element and another giving the measure value:

```
a_df <- tibble(element = c("first", "first", "first",
                           "second", "second", "second"),
               measure = c(1, 2, 3, 11, 15, 20))
```

Applying a function repeatedly

Here's what the data looks like:

```
a_df
```

```
## # A tibble: 6 x 2
##   element measure
##   <chr>     <dbl>
## 1 first         1
## 2 first         2
## 3 first         3
## 4 second       11
## 5 second       15
## 6 second       20
```

Applying a function repeatedly

You can `group_by` and `nest` the data to get a column that gives a list of values in each element, like the list we were working with before:

```
a_df %>%
  group_by(element) %>%
  nest()
```

```
## # A tibble: 2 x 2
## # Groups:   element [2]
##   element data
##   <chr>   <list>
## 1 first   <tibble [3 x 1]>
## 2 second  <tibble [3 x 1]>
```

Applying a function repeatedly

We can use `map` inside `mutate` to map the `range` function across each value:

```
a_df %>%
  group_by(element) %>%
  nest() %>%
  mutate(range = map(.x = data, .f = range))
```

```
## # A tibble: 2 x 3
## # Groups:   element [2]
##   element data             range
##   <chr>   <list>           <list>
## 1 first   <tibble [3 x 1]> <dbl [2]>
## 2 second  <tibble [3 x 1]> <dbl [2]>
```

Applying a function repeatedly

Then unnest to get the values in a "normal" column:

```
a_df %>%
  group_by(element) %>%
  nest() %>%
  mutate(range = map(.x = data, .f = range)) %>%
  unnest(range)
```

```
## # A tibble: 4 x 3
## # Groups:   element [2]
##   element data              range
##   <chr>   <list>            <dbl>
## 1 first   <tibble [3 x 1]>     1
## 2 first   <tibble [3 x 1]>     3
## 3 second  <tibble [3 x 1]>    11
## 4 second  <tibble [3 x 1]>    20
```

Applying a function repeatedly

Other members of the `map` family of functions will let you map across the elements of two (`map2`) or more (`pmap`) R objects.

As a simple example, say you have two vectors of words, `first_word` and `second_word`:

```
first_word <- c("open", "ride", "moot")
second_word <- c("source", "share", "point")
```

You want to paste these together, using `first_word` as the first element and `second_word` as the second element.

Applying a function repeatedly

You can use the `map2` function to put `first_word` in as the first argument (`.x`) for the function (`.f`) and `second_word` in as the second argument (`.y`):

```
map2(.x = first_word,
     .y = second_word,
     .f = paste)
```

```
## [[1]]
## [1] "open source"
##
## [[2]]
## [1] "ride share"
##
## [[3]]
## [1] "moot point"
```

Applying a function repeatedly

You can also use this function within a dataframe:

```
df <- tibble(first_word = c("open", "ride", "moot"),
             second_word = c("source", "share", "point"))
df
```

```
## # A tibble: 3 x 2
##   first_word second_word
##   <chr>      <chr>
## 1 open       source
## 2 ride       share
## 3 moot       point
```

Applying a function repeatedly

The resulst is a list-column:

```
df %>%
  mutate(phrase = map2(.x = first_word,
                       .y = second_word,
                       .f = paste))
```

```
## # A tibble: 3 x 3
##   first_word second_word phrase
##   <chr>      <chr>       <list>
## 1 open       source      <chr [1]>
## 2 ride       share       <chr [1]>
## 3 moot       point       <chr [1]>
```

Applying a function repeatedly

You can unnest this list-column to get it back to a "normal" column:

```
df %>%
  mutate(phrase = map2(.x = first_word,
                       .y = second_word,
                       .f = paste)) %>%
  unnest(phrase)
```

```
## # A tibble: 3 x 3
##   first_word second_word phrase
##   <chr>      <chr>       <chr>
## 1 open       source      open source
## 2 ride       share       ride share
## 3 moot       point       moot point
```