Hi everyone,

Thank you once again for joining our purrr resolution for 2018, which aims to help us learn (at least) one purrr function each week. As one of you mentioned, this is the purrrfect resolution for 2018 and we are now in a position where we can leverage our joint passion and knowledge to bring this resolution to life.

As you know, the challenge for Week 1 was for us to get more familiar with the function modify_depth() from purrr. The blurb at http://www.ghement.ca/modify_depth.pdf illustrated a particular use of this function.

If you followed the blurb, you know that its aim was to help produce a nested list called `yearly_data_mod` which looked like this:

```
> yearly_data_mod
$`2016`
$`2016`$Winter
  Site Temp
1   01   15
2   02   17
3   03   20

$`2016`$Summer
  Site Temp
1   01   14
2   02   16
3   03   19


$`2017`
$`2017`$Winter
  Site Temp
1   01   16
2   02   18
3   03   21

$`2017`$Summer
  Site Temp
1   01   17
2   02   19
3   03   22
```

The blurb concluded by asking how we can convert the values of the Temp variable (which stands for Temperature, measured in degrees Celsius) from degrees Celsius to degrees Fahrenheit for each year and season represented in the nested list (while preserving the structure of the list).

Following suggestions from Bruno Rodrigues and Christopher Peters, I was able to come up with two different solutions for this question, though I am sure there may be others.

Solution No. 1 [Uses modify_depth() call]:

```
require(purrr)

yearly_data_mod_transf <- modify_depth(yearly_data_mod, .depth = 2, .f =
function(x){ x$Temp <- x$Temp*(9/5) + 32; return(x) })

yearly_data_mod_transf
```

If we wanted to use the pipe operator, the code above could be re-expressed as:

```
yearly_data_mod_transf <- yearly_data_mod %>%
                          modify_depth(.depth = 2, .f = function(x){ x$Temp
<- x$Temp*(9/5) + 32; return(x)})



yearly_data_mod_transf
```

Using the modify_depth() function to apply a transformation only to the Temp column from each of the data frames embedded within a season by year combination involves two things:

1. Specifying the depth at which the data frame is located (in this case 2, since the year represents level 1 and the season within year represents level 2) - this is achieved with the option `.depth = 2` of modify_depth();

2. Specifying the function to be applied to the Temp column - this is achieved with the option `.f = function(x){ x$Temp <- x$Temp*(9/5) + 32; return(x)}` of modify_depth().

Because we want the nested list structure to be preserved after converting Temp from degrees Celsius to degrees Fahrenheit, we have to apply the conversion function `.f` to the entire data frame (denoted generically by `x`) but make sure that, inside the function, we only pluck Temp from the data frame (using the syntax `x$Temp`) and subject it to the necessary conversion. The conversion consists of multiplying the plucked temperature `x$Temp` by 9/5 and then adding 32 to the result of that multiplication.

These two steps are illustrated below in schematic fashion via the annotations listed in red colour:

```
> yearly_data_mod


$`2016`         <---  level 1 (year)
$`2016`$Winter <---  level 2 (season within year)
  Site Temp     <---  data frame x (which includes the Temp variable, x$Temp)
1   01   15
2   02   17
3   03   20
```

```
$`2016`$Summer
  Site Temp     <---  data frame x (which includes the Temp variable, x$Temp)
1   01   14
2   02   16
3   03   19


$`2017`
$`2017`$Winter
  Site Temp     <---  data frame x (which includes the Temp variable, x$Temp)
1   01   16
2   02   18
3   03   21

$`2017`$Summer
  Site Temp     <---  data frame x (which includes the Temp variable, x$Temp)
1   01   17
2   02   19
3   03   22
```

Solution No. 2 [Uses nested map() call]:

The second solution follows the same principles as explained above for the first solution, but uses different means to implement them. We still concern ourselves with the data frame `.x` (i.e., the data frame containing the Temp variable, of which we have one for each season and year combination) and we still want to apply the Celsius to Fahrenheit conversion to the Temp variable located in that data frame. But notice how we now use an inner call to the function map() from the purrr package, `~ map(.x, function(x){ x$Temp <- x$Temp*(9/5) + 32; return(x)})`, to apply the temperature conversion to the data frame. This inner call to the function map() is first preceded by the tilde operator **~** and then embedded in an outer call to the function map(), which helps us apply the conversion to **all** data frames corresponding to the years and seasons combinations represented in the nested list. The outer call to map() looks like this:

```
map(yearly_data_mod, ~ map (.x, function(x){ x$Temp <- x$Temp*(9/5) + 32;
return(x)}))
```

Here is the R code for the second solution:

```
require(purrr)

yearly_data_mod_transf <-  map(yearly_data_mod, ~ map(.x, function(x){ x$Temp
<- x$Temp*(9/5) + 32; return(x)} ))

yearly_data_mod_transf
```

Of course, if we wanted to use the pipe operator, the code above for the second solution could be re-expressed as:

```
require(purrr)

require(magrittr)
```

```
yearly_data_mod_transf <-  yearly_data_mod %>%
                             map(~ map(.x, function(x){ x$Temp <-
x$Temp*(9/5) + 32; return(x)} ))

yearly_data_mod_transf
```

Our next installment of purrr goodness for Week 2 will come from Christopher Peters (Twitter Handle: @statwonk).

Keep on purrring!

Isabella

Isabella R. Ghement, Ph.D.
Ghement Statistical Consulting Company Ltd.
301-7031 Blundell Road, Richmond, B.C., Canada, V6Y 1J5
Tel: 604-767-1250
Fax: 604-270-3922
E-mail: isabella@ghement.ca
Web: www.ghement.ca