Data Science for Operational Researchers Using R Online

8. Tibble Manipulation with purrr and tidyr

Prof. Jim Duggan,
School of Computer Science
University of Galway.

https://github.com/JimDuggan/explore_or

> R has numerous ways to iterate over elements of a list (or vector), and Hadley Wickham aimed to improve on and standardise that experience with the `purrr` package.
>
> — Jared P. Lander (Lander, 2017)

Overview

- Overview
- dplyr::group_split()
- tidyr::nest()
- Linear regression example
- Additional purrr functions

1. Overview

- A benefit of using the tidyverse is having the facility to combine tools from different packages, and switching between the use of lists and tibbles where appropriate.
- A common task is to divide a tibble into sub-groups, and perform operations on these.
- We have already seen how this can work using the package dplyr, which allows you to use the functions group_by() and summarize() to aggregate data
- The two functions we use with purrr are dplyr:: group_split() tidyr::nest()

2. group_split()

- This function, contained in the package dplyr, can be used to split a tibble into a list of tibbles, based on groupings specified by group_by().
- This list can be processed using the map() family of functions.

Create a subset of mpg

```
set.seed(100)
test <- mpg %>%
        dplyr::select(manufacturer:displ,cty,class) %>%
        dplyr::filter(class %in% c("compact","midsize")) %>%
        dplyr::sample_n(5)
test
#> # A tibble: 5 x 5
#>   manufacturer model  displ   cty class
#>   <chr>        <chr>  <dbl> <int> <chr>
#> 1 volkswagen   jetta    2      21 compact
#> 2 volkswagen   jetta    2.5    21 compact
#> 3 chevrolet    malibu   3.6    17 midsize
#> 4 volkswagen   gti      2      21 compact
#> 5 audi         a4       2      21 compact
```

Use group_by() and group_split()

```r
test_s <- test %>%
          dplyr::group_by(class) %>%
          dplyr::group_split()
```

```
#> [[1]]
#> # A tibble: 4 x 5
#>   manufacturer model displ   cty class
#>   <chr>        <chr> <dbl> <int> <chr>
#> 1 volkswagen   jetta   2      21 compact
#> 2 volkswagen   jetta   2.5    21 compact
#> 3 volkswagen   gti     2      21 compact
#> 4 audi         a4      2      21 compact
```

```
#> [[2]]
#> # A tibble: 1 x 5
#>   manufacturer model  displ   cty class
#>   <chr>        <chr>  <dbl> <int> <chr>
#> 1 chevrolet    malibu   3.6    17 midsize
```

Processing the list with purrr::map_int()

```
test_s %>% purrr::map_int(~nrow(.x))
#> [1] 4 1
```

```
#> [[1]]
#> # A tibble: 4 x 5
#>   manufacturer model displ   cty class
#>   <chr>        <chr> <dbl> <int> <chr>
#> 1 volkswagen   jetta   2      21 compact
#> 2 volkswagen   jetta   2.5    21 compact
#> 3 volkswagen   gti     2      21 compact
#> 4 audi         a4      2      21 compact
```

```
#> [[2]]
#> # A tibble: 1 x 5
#>   manufacturer model  displ   cty class
#>   <chr>        <chr>  <dbl> <int> <chr>
#> 1 chevrolet    malibu   3.6    17 midsize
```

More detailed example…

- Our goal is to calculate the correlation coefficient between two variables: mean sea level pressure and average wind speed.
- We simplify the dataset to daily values, where we take (1) the maximum wind speed (wdsp) recorded and (2) the average mean sea level pressure (msl).
- Our first task is to use dplyr to generate a summary tibble, and we also exclude any cases that have missing values, by combining complete.cases() within filter().
- Note that the function complete.cases() returns a logical vector indicating which rows are complete.
- The new tibble is stored in the variable d_data.

```r
d_data <- observations %>%
          dplyr::filter(complete.cases(observations)) %>%
          dplyr::group_by(station,month,day) %>%
          dplyr::summarize(MaxWdsp=max(wdsp,na.rm=TRUE),
                           DailyAverageMSL=mean(msl,na.rm=TRUE)) %>%
          dplyr::ungroup()
d_data
#> # A tibble: 8,394 x 5
#>    station month   day MaxWdsp DailyAverageMSL
#>    <chr>   <dbl> <int>   <dbl>           <dbl>
#>  1 ATHENRY     1     1      12           1027.
#>  2 ATHENRY     1     2       8           1035.
#>  3 ATHENRY     1     3       6           1032.
#>  4 ATHENRY     1     4       4           1030.
#>  5 ATHENRY     1     5       9           1029.
#>  6 ATHENRY     1     6       9           1028.
#>  7 ATHENRY     1     7       6           1032.
#>  8 ATHENRY     1     8       9           1029.
#>  9 ATHENRY     1     9      16           1015.
#> 10 ATHENRY     1    10      13           1013.
#> # ... with 8,384 more rows
```

```r
cor7 <- d_data %>%
         dplyr::group_by(station) %>%
         dplyr::group_split() %>%
         purrr::map_df(~{
                  corr <- cor(.x$MaxWdsp,.x$DailyAverageMSL)
                  tibble(Station=first(.x$station),
                         CorrCoeff=corr)
         }) %>%
         dplyr::arrange(CorrCoeff) %>%
         dplyr::slice(1:7)
```

```r
cor7
#> # A tibble: 7 x 2
#>   Station              CorrCoeff
#>   <chr>                    <dbl>
#> 1 SherkinIsland           -0.589
#> 2 VALENTIA OBSERVATORY    -0.579
#> 3 ROCHES POINT            -0.540
#> 4 MACE HEAD               -0.539
#> 5 MOORE PARK              -0.528
```

Using summarize()

```
cor7_b <- d_data %>%
          dplyr::group_by(station) %>%
          dplyr::summarize(CorrCoeff=cor(MaxWdsp,DailyAverageMSL)) %>%
          dplyr::arrange(CorrCoeff) %>%
          dplyr::slice(1:7)
cor7_b
#> # A tibble: 7 x 2
#>   station              CorrCoeff
#>   <chr>                    <dbl>
#> 1 SherkinIsland           -0.589
#> 2 VALENTIA OBSERVATORY    -0.579
#> 3 ROCHES POINT            -0.540
#> 4 MACE HEAD               -0.539
#> 5 MOORE PARK              -0.528
```

3. nest()

- The function nest(), which is part of the package tidyr, can be used to create a list column within a tibble that contains a tibble.
- Nesting generates one row for each defined group, which is identified using the function group_by().
- The second column is named data, and is a list, and each list element contains all of the tibble's data for a particular group.

```r
data_n <- d_data %>%
          dplyr::group_by(station) %>%
          tidyr::nest()

data_n %>% head()
#> # A tibble: 6 x 2
#> # Groups:   station [6]
#>   station      data
#>   <chr>        <list>
#> 1 ATHENRY      <tibble [365 x 4]>
#> 2 BALLYHAISE   <tibble [365 x 4]>
#> 3 BELMULLET    <tibble [365 x 4]>
#> 4 CASEMENT     <tibble [365 x 4]>
#> 5 CLAREMORRIS  <tibble [365 x 4]>
#> 6 CORK AIRPORT <tibble [365 x 4]>
```

```r
data_n %>% head()
#> # A tibble: 6 x 2
#> # Groups:   station [6]
#>   station       data
#>   <chr>         <list>
#> 1 ATHENRY       <tibble [365 x 4]>
#> 2 BALLYHAISE    <tibble [365 x 4]>
#> 3 BELMULLET     <tibble [365 x 4]>
#> 4 CASEMENT      <tibble [365 x 4]>
#> 5 CLAREMORRIS   <tibble [365 x 4]>
#> 6 CORK AIRPORT  <tibble [365 x 4]>
```

```r
data_n %>%
  dplyr::pull(data) %>%
  dplyr::first()
```

```r
#> # A tibble: 365 x 4
#>     month   day MaxWdsp DailyAverageMSL
#>     <dbl> <int>   <dbl>           <dbl>
#>  1      1     1      12           1027.
#>  2      1     2       8           1035.
#>  3      1     3       6           1032.
#>  4      1     4       4           1030.
#>  5      1     5       9           1029.
#>  6      1     6       9           1028.
#>  7      1     7       6           1032.
#>  8      1     8       9           1029.
#>  9      1     9      16           1015.
#> 10      1    10      13           1013.
#> # ... with 355 more rows
```

4. Run linear regression model.

```
data_n <- data_n    %>%
            dplyr::mutate(LM=map(data,
                                      ~lm(MaxWdsp~DailyAverageMSL,
                                          data=.)))

data_n %>%
  head()
#> # A tibble: 6 x 3
#> # Groups:   station [6]
#>   station      data                 LM
#>   <chr>        <list>               <list>
#> 1 ATHENRY      <tibble [365 x 4]> <lm>
#> 2 BALLYHAISE   <tibble [365 x 4]> <lm>
#> 3 BELMULLET    <tibble [365 x 4]> <lm>
#> 4 CASEMENT     <tibble [365 x 4]> <lm>
#> 5 CLAREMORRIS  <tibble [365 x 4]> <lm>
#> 6 CORK AIRPORT <tibble [365 x 4]> <lm>
```

```r
data_n %>%
  head()
#> # A tibble: 6 x 3
#> # Groups:   station [6]
#>   station     data               LM
#>   <chr>       <list>             <list>
#> 1 ATHENRY     <tibble [365 x 4]> <lm>
#> 2 BALLYHAISE  <tibble [365 x 4]> <lm>
#> 3 BELMULLET   <tibble [365 x 4]> <lm>
#> 4 CASEMENT    <tibble [365 x 4]> <lm>
#> 5 CLAREMORRIS <tibble [365 x 4]> <lm>
#> 6 CORK AIRPORT <tibble [365 x 4]> <lm>
```

```r
data_n %>%
  dplyr::filter(station=="BELMULLET") %>%
  dplyr::pull(LM) %>%
  dplyr::first() %>%
  summary()
#>
#> Call:
#> lm(formula = MaxWdsp ~ DailyAverageMSL, data = .)
#>
#> Residuals:
#>     Min      1Q  Median      3Q     Max
#> -14.021  -4.069  -0.516   3.958  17.962
#>
#> Coefficients:
#>                 Estimate Std. Error t value Pr(>|t|)
#> (Intercept)      242.786     26.365    9.21   <2e-16 ***
#> DailyAverageMSL   -0.222      0.026   -8.53    4e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 5.8 on 363 degrees of freedom
#> Multiple R-squared:  0.167,  Adjusted R-squared:  0.165
#> F-statistic: 72.8 on 1 and 363 DF,  p-value: 4.03e-16
```

```r
data_n <- data_n %>%
            dplyr::mutate(RSq=map_dbl(LM,~summary(.x)$r.squared)) %>%
            dplyr::arrange(desc(RSq))
data_n  <- data_n %>% head(n=3)
data_n
#> # A tibble: 3 x 4
#> # Groups:   station [3]
#>   station              data                 LM        RSq
#>   <chr>                <list>               <list>  <dbl>
#> 1 SherkinIsland        <tibble [365 x 4]> <lm>    0.347
#> 2 VALENTIA OBSERVATORY <tibble [365 x 4]> <lm>    0.335
#> 3 ROCHES POINT         <tibble [365 x 4]> <lm>    0.291
```

5. Additional purr functions

pluck()

- The function pluck() provides a generalized form of the [[operator and provides the means to index data structures in a flexible way.
- The arguments include .x, which is a vector, and a list of accessors for indexing into the object, which can include an integer position or a string name.
- Here are some examples.

```r
library(ggplot2)
library(repurrrsive)

# Use pluck() to access the second element of an atomic vector
mpg %>% dplyr::pull(class) %>% unique() %>% purrr::pluck(2)
#> [1] "midsize"
# Use pluck() to access the director in the first list location
sw_films %>% purrr::pluck(1,"director")
#> [1] "George Lucas"
```

walk()

- The function walk(.x,.f) is similar to map, except that it returns the input .x and calls the function .f to generate a side effect.
- The side effect, for example, could be displaying information onto the screen, and no output value needs to be returned.

```r
l <- list(el1=20,el2=30,el3=40)
o <- purrr::walk(l,~cat("Creating a side effect...\n"))
#> Creating a side effect...
#> Creating a side effect...
#> Creating a side effect...
str(o)
#> List of 3
#>  $ el1: num 20
#>  $ el2: num 30
#>  $ el3: num 40
```

keep()

- The function keep(.x,.f) takes in a list .x and, based on the evaluation of a predicate function, will either keep or discard list element.
- In effect, it provides a way to filter a list. Here, we can filter those movies that have George Lucas as a director, and then confirm the result using walk().

```
o <- sw_films %>% keep(~.x$director=="George Lucas")
purrr::walk(o,~cat(.x$director," ==> Title =",.x$title,"\n"))
#> George Lucas  ==> Title = A New Hope
#> George Lucas  ==> Title = Attack of the Clones
#> George Lucas  ==> Title = The Phantom Menace
#> George Lucas  ==> Title = Revenge of the Sith
```

3. Generate the following daily summaries of rainfall and mean sea level pressure, for all the weather stations in `aimsir17::observations`, and only consider observations with no missing values.

```
#> `summarize()` has grouped output by 'station', 'month'. You can
#> override using the `.groups` argument.
```

```
d_sum
#> # A tibble: 8,394 x 5
#>    station month   day TotalRain AvrMSL
#>    <chr>   <dbl> <int>     <dbl>  <dbl>
#>  1 ATHENRY     1     1       0.2  1027.
#>  2 ATHENRY     1     2       0    1035.
#>  3 ATHENRY     1     3       0    1032.
#>  4 ATHENRY     1     4       0    1030.
#>  5 ATHENRY     1     5       0.1  1029.
#>  6 ATHENRY     1     6      18    1028.
#>  7 ATHENRY     1     7       1.4  1032.
#>  8 ATHENRY     1     8       1.2  1029.
#>  9 ATHENRY     1     9       5.4  1015.
#> 10 ATHENRY     1    10       0.7  1013.
#> # ... with 8,384 more rows
```

Next, using the tibble d_sum as input, generate the top 6 correlations between TotalRain and AvrMSLusing group_split() and map_df(). Here are the results you should find.

```
cors
#> # A tibble: 6 x 2
#>   Station                  Corr
#>   <chr>                   <dbl>
#> 1 MOORE PARK             -0.496
#> 2 MULLINGAR              -0.496
#> 3 CLAREMORRIS            -0.484
#> 4 VALENTIA OBSERVATORY   -0.464
#> 5 KNOCK AIRPORT          -0.459
#> 6 OAK PARK               -0.456
```