Team A : Rainfall Data Processing

Barry Rowlingson

November 2, 2021

1 Work-through Solution

This document is a "workthrough" - its like a walk-through but instead tries to take you through the working out of the solution and the thought processes involved. Some of the stages are shown step-by-step as the solution develops.

2 The One Function

This is what the task boils down to - read the files from a path and write the new ones to an output path. So the first thing I do is write this function. I don't expect it to work now, its just a skeleton, but it gives me a focus of what I want to end up with. I add some comments on the stages I might have to implement and think about this a bit.

```
process_weather <- function(input_path, output_path){
    ## find all input files
    ...
    ## read and process each input file into data
    ...
    ## select years from 1851 to 1900 (inclusive)
    ...
    ## merge input data
    ...
    ## output readings.csv and stations.csv
    ...
}
```

3 Getting The List of Input Files

The data files are some levels deep in a folder structure. I can use `list.files` to scan that. I'll put it in a function so the name has some meaning and if I need to adjust it later I've got a single function I can change and anything that uses that function won't break. Here's the first draft, and a test:

```
get_rainfall_files <- function(path){
### return all the station files under the given path
### as a vector of full paths to the files.
    return(
        list.files(
            path, recursive=TRUE, full.names=TRUE,
            pattern="*.csv", ignore.case=TRUE)
    )
}

### Test

all_files = get_rainfall_files("./DATA")
all_files[1:2]

[1] "./DATA/LANCASTER-BLEA-TARN-RES/LANCASTER-BLEA-TARN-RES.csv"
[2] "./DATA/LANCASTER-CATON/LANCASTER-CATON-2.csv"
```

Okay that looks good. I could test that further by putting some non-CSV files in there and making sure it doesn't include them in the output, but the input folders don't seem to have anything we don't want to read.

4 Get Station Information

Now to get the station information. First draft is a skeleton.

```
read_station_info <- function(stationfile){
### read station info for a station file
### return a data frame of name, id, elevation, location etc

}
```

Next to experiment with reading in one of the files. We know the information is in the first few lines so I'll only read that.

```
read.csv(all_files[3], nrows=3)
```

```
  CATON.GREEN...observer.Storey        X  X.1       X.2 X.3      X.4       X.5
1                     Grid ref SD549652 Long -2.690842 Lat 54.08073 Elevation
2                   Station no   RR1669             NA           NA
3                                                   NA           NA
  X.6 X.7 X.8 X.9 X.10 X.11 X.12 X.13 X.14 X.15 X.16 X.17 X.18 X.19
1 250  ft  NA  NA   NA   NA   NA   NA   NA   NA   NA   NA   NA   NA
2  NA      NA  NA   NA   NA   NA   NA   NA   NA   NA   NA   NA   NA
3  NA      NA  NA   NA   NA   NA   NA   NA   NA   NA   NA   NA   NA
```

That's read the first row into a header which seems to have modified it with dots, so let's try again and tell `read.csv` that we don't want the first row to be a header:

```
read.csv(all_files[3], nrows=3, header=FALSE)
```

```
                           V1       V2   V3        V4  V5       V6        V7
1 CATON GREEN - observer Storey                    NA           NA
2                     Grid ref SD549652 Long -2.690842 Lat 54.08073 Elevation
3                   Station no   RR1669             NA           NA
   V8 V9 V10 V11 V12 V13 V14 V15 V16 V17 V18 V19 V20 V21
1  NA     NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA
2 250 ft  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA
3  NA     NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA
```

Now we can see which row and column have the station information. We can build this into our function and test it on a couple of examples:

```
read_station_info <- function(stationfile){
### read station info for a station file
### return a data frame of name, id, elevation, location etc
    lines = read.csv(stationfile, nrows=3, header=FALSE)
    info = data.frame(name=lines[1,1],
                      stationID = lines[3,2],
                      long=lines[2,4], lat=lines[2,6],
                      elev=lines[2,8])
    return(info)
}

read_station_info(all_files[1])
```

```
                        name stationID      long      lat elev
1 LANCASTER BLEA TARN RESERVOIR      3048 -2.772221 54.01822  330
```

```
read_station_info(all_files[2])
```

```
               name stationID    long    lat elev
1 CATON SCHOOL HOUSE    RR1669 -2.7004 54.073  170
```

Next lets write a function that takes a vector of paths to multiple station data files and returns all the information records.

```
read_all_station_info <- function(paths){
### given a vector of paths, read the station info and combine
### into a data frame

    ## transform paths to list of station info
    all_info_list = Map(read_station_info, paths)

    ## join station info data frames by row-binding
    all_info = Reduce(rbind, all_info_list)
    all_info
}
```

I can then test this on one or two files.

```
read_all_station_info(all_files[1])

                          name stationID      long      lat elev
1 LANCASTER BLEA TARN RESERVOIR      3048 -2.772221 54.01822  330

read_all_station_info(all_files[1:2])

                          name stationID      long      lat elev
1 LANCASTER BLEA TARN RESERVOIR      3048 -2.772221 54.01822  330
2            CATON SCHOOL HOUSE    RR1669 -2.700400 54.07300  170
```

And then read the whole lot:

```
stations = read_all_station_info(all_files)
head(stations)

                          name stationID      long      lat elev
1 LANCASTER BLEA TARN RESERVOIR      3048 -2.772221 54.01822  330
2            CATON SCHOOL HOUSE    RR1669 -2.700400 54.07300  170
3 CATON GREEN - observer Storey    RR1669 -2.690842 54.08073  250
4 CATON GREEN - observer Holden    RR1669 -2.690000 54.07980  250
5          CATON NEAR LANCASTER    RR1669 -2.709075 54.07433  160
6        LANCASTER ELLEL GRANGE     RR395 -2.792400 53.97760  125

summary(stations)

     name            stationID             long             lat
 Length:12          Length:12          Min.   :-2.807   Min.   :53.98
 Class :character   Class :character   1st Qu.:-2.794   1st Qu.:54.04
 Mode  :character   Mode  :character   Median :-2.750   Median :54.07
                                       Mean   :-2.740   Mean   :54.06
                                       3rd Qu.:-2.698   3rd Qu.:54.08
                                       Max.   :-2.610   Max.   :54.10
      elev
 Min.   : 30.0
 1st Qu.:122.2
 Median :162.5
 Mean   :176.2
 3rd Qu.:250.0
 Max.   :330.0
```

That looks ready to write the the required `stations.csv` file which we'll come back to in a bit.

5 Reading the Rainfall Data

First we'll break this down to read one file, then combine as we did before. First write a prototype function:

```
read_rainfall <- function(stationfile){
### take one station file and return its rainfall data in ID, month, year, rainfall columns
```

```
    ## 1. read the rainfall rows from the data

    ## 2. rearrange the data from wide to long

    ## 3. filter the year and any missing data

}
```

Let's look at the first sub-task there. We can use `read.csv`, but we have to skip four lines to get to the monthly data. Then we only want to read 12 lines. Let's try:

```
d = read.csv(all_files[5], skip=4, nrows=12)
## see the first ten columns...
d[,1:10]
```

```
           X X1850 X1851 X1852 X1853 X1854 X1855 X1856 X1857 X1858
1    January    NA    NA  5.43  3.95  3.84  0.63  2.72  2.93  2.14
2   February    NA    NA  3.00  1.32  2.45  0.86  4.03  3.27  0.08
3      March    NA    NA  0.13  0.75  2.32  2.82  0.36  2.81  2.43
4      April    NA    NA  1.01  3.45  0.94  1.18  2.73  1.56  3.02
5        May    NA    NA  2.83  0.60  2.52  1.63  1.75  2.33  3.16
6       June    NA    NA  2.72  3.50  4.35  4.97  4.56  3.74  1.48
7       July    NA    NA  3.51  6.92  3.04  2.82  3.48  1.88  3.54
8     August    NA    NA  6.95  2.83  4.62  2.49  4.85  3.14  3.06
9  September    NA    NA  2.83  3.12  2.84  2.02  3.47  2.83  5.61
10   October    NA    NA  4.35  4.72  3.96  6.36  1.93  2.57  5.69
11  November    NA    NA  6.64  2.18  3.07  2.13  1.44  2.42  1.91
12  December    NA    NA  9.20  0.35  5.68  2.12  3.87  2.94  3.70
```

Okay, that seems to have everything. The first column has the months and the years are in the column headers. Can we simply melt that to something close to our final form with `melt` from the `reshape2` package? Let's try:

```
d = read.csv(all_files[5], skip=4, nrows=12)
dshape = reshape2::melt(d, id.vars="X")
head(dshape)
```

```
         X variable value
1  January    X1850    NA
2 February    X1850    NA
3    March    X1850    NA
4    April    X1850    NA
5      May    X1850    NA
6     June    X1850    NA
```

```
summary(dshape)
```

```
      X                variable       value
 Length:840         X1850  : 12   Min.   :-999.000
 Class :character   X1851  : 12   1st Qu.:   2.135
 Mode  :character   X1852  : 12   Median :   3.120
                    X1853  : 12   Mean   :  -5.231
                    X1854  : 12   3rd Qu.:   4.500
                    X1855  : 12   Max.   :  11.810
                    (Other):768   NA's   :493
```

We can start building up our function now.

```
read_rainfall <- function(stationfile){
### take one station file and return its rainfall data in ID, month, year, rainfall columns

    ## 1. read the rainfall rows from the data
    d = read.csv(stationfile, skip=4, nrows=12)
    ## 2. rearrange the data from wide to long
```

```
    dshape = reshape2::melt(d, id.vars="X")
    ## 3. filter the year and any missing data

    ## 4. return the data in long format
    return(dshape)
}
```

Stage 3 requires testing and filtering out some rows. We can test if the rainfall value is NA and remove those:

```
read_rainfall <- function(stationfile){
### take one station file and return its rainfall data in ID, month, year, rainfall columns

    ## 1. read the rainfall rows from the data
    d = read.csv(stationfile, skip=4, nrows=12)
    ## 2. rearrange the data from wide to long
    dshape = reshape2::melt(d, id.vars="X")
    ## 3. filter the year and any missing data
    dshape = dshape[!is.na(dshape$value),]
    ## 4. return the data in long format
    return(dshape)
}

d1 = read_rainfall(all_files[5])
summary(d1)
```

```
      X                variable       value
 Length:347         X1852  : 12   Min.   :-999.000
 Class :character   X1853  : 12   1st Qu.:   2.135
 Mode  :character   X1854  : 12   Median :   3.120
                    X1855  : 12   Mean   :  -5.231
                    X1856  : 12   3rd Qu.:   4.500
                    X1857  : 12   Max.   :  11.810
                    (Other):275
```

We also want to remove that negative rainfall value, so add another line to do that:

```
read_rainfall <- function(stationfile){
### take one station file and return its rainfall data in ID, month, year, rainfall columns

    ## 1. read the rainfall rows from the data
    d = read.csv(stationfile, skip=4, nrows=12)
    ## 2. rearrange the data from wide to long
    dshape = reshape2::melt(d, id.vars="X")
    ## 3. filter the year and any missing data
    dshape = dshape[!is.na(dshape$value),]
    dshape = dshape[dshape$value >=0, ]

    ## 4. return the data in long format
    return(dshape)
}

d1 = read_rainfall(all_files[5])
head(d1)
```

```
          X variable value
25  January    X1852  5.43
26 February    X1852  3.00
27    March    X1852  0.13
28    April    X1852  1.01
29      May    X1852  2.83
30     June    X1852  2.72
```

```
summary(d1)
```

```
      X                  variable       value        
 Length:344         X1852  : 12   Min.   : 0.080  
 Class :character   X1853  : 12   1st Qu.: 2.178  
 Mode  :character   X1854  : 12   Median : 3.130  
                    X1855  : 12   Mean   : 3.435  
                    X1856  : 12   3rd Qu.: 4.518  
                    X1857  : 12   Max.   :11.810  
                    (Other):272                   
```

That is looking almost there. But the year column isn't right, since the years have that "X" in front of them. To get parts of a string we can use the `substr()` function - let's test a bit. I think we want to start at character 2 and end at character 5:

```
substr("X1999", 2, 5)
```

```
[1] "1999"
```

That's still a character string, and we need a number so let's convert that:

```
as.numeric(substr("X1900", 2, 5))
```

```
[1] 1900
```

Now put that into our function. We'll also change the names of our columns to more meaningful things.

```
read_rainfall <- function(stationfile){
### take one station file and return its rainfall data in ID, month, year, rainfall columns

    ## 1. read the rainfall rows from the data
    d = read.csv(stationfile, skip=4, nrows=12)
    ## 2. rearrange the data from wide to long
    dshape = reshape2::melt(d, id.vars="X")
    ## 3. filter the year and any missing data
    dshape = dshape[!is.na(dshape$value),]
    dshape = dshape[dshape$value >=0, ]

    ## 3.1 set the name
    names(dshape) = c("month","year","rainfall")
    ## 3.2 convert the year to numeric
    dshape$year = as.numeric(substr(dshape$year,2,5))
    ## 4. return the data in long format
    return(dshape)
}

d1 = read_rainfall(all_files[5])
head(d1)
```

```
      month year rainfall
25  January 1852     5.43
26 February 1852     3.00
27    March 1852     0.13
28    April 1852     1.01
29      May 1852     2.83
30     June 1852     2.72
```

```
summary(d1)
```

```
    month                year         rainfall      
 Length:344         Min.   :1852   Min.   : 0.080  
 Class :character   1st Qu.:1859   1st Qu.: 2.178  
 Mode  :character   Median :1866   Median : 3.130  
                    Mean   :1877   Mean   : 3.435  
                    3rd Qu.:1903   3rd Qu.: 4.518  
                    Max.   :1910   Max.   :11.810  
```

Next add a line to select the years between 1851 and 1900 (inclusive). I'll make the function flexible in this regard by adding a start and end year as parameters, and having these as defaults.

```
read_rainfall <- function(stationfile, first_year=1851, last_year=1900){
### take one station file and return its rainfall data in ID, month, year, rainfall columns

    ## 1. read the rainfall rows from the data
    d = read.csv(stationfile, skip=4, nrows=12)
    ## 2. rearrange the data from wide to long
    dshape = reshape2::melt(d, id.vars="X")
    ## 3. filter the year and any missing data
    dshape = dshape[!is.na(dshape$value),]
    dshape = dshape[dshape$value >=0, ]

    ## 3.1 set the name
    names(dshape) = c("month","year","rainfall")
    ## 3.2 convert the year to numeric
    dshape$year = as.numeric(substr(dshape$year,2,5))
    ## 4. return the data in long format
    dshape = dshape[dshape$year >= first_year & dshape$year <= last_year,]
    return(dshape)
}

## test over default year range
d1 = read_rainfall(all_files[5])
head(d1)
```

```
      month year rainfall
25  January 1852     5.43
26 February 1852     3.00
27    March 1852     0.13
28    April 1852     1.01
29      May 1852     2.83
30     June 1852     2.72
```

```
summary(d1)
```

```
    month                year         rainfall
 Length:228         Min.   :1852   Min.   : 0.080
 Class :character   1st Qu.:1856   1st Qu.: 2.237
 Mode  :character   Median :1861   Median : 3.130
                    Mean   :1863   Mean   : 3.452
                    3rd Qu.:1866   3rd Qu.: 4.545
                    Max.   :1900   Max.   :11.810
```

```
## test over a short year range, make sure years returned are within:
d2 = read_rainfall(all_files[5], first_year=1900, last_year=1902)
summary(d2)
```

```
    month                year         rainfall
 Length:32          Min.   :1900   Min.   : 0.450
 Class :character   1st Qu.:1900   1st Qu.: 1.762
 Mode  :character   Median :1901   Median : 2.630
                    Mean   :1901   Mean   : 3.200
                    3rd Qu.:1902   3rd Qu.: 4.147
                    Max.   :1902   Max.   :11.810
```

There's still one thing missing from the data frame - the station ID. We can get that by calling the function we wrote earlier and getting its `stationID` column.

```
read_rainfall <- function(stationfile, first_year=1851, last_year=1900){
### take one station file and return its rainfall data in ID, month, year, rainfall columns

    ## 1. read the rainfall rows from the data
```

```
    d = read.csv(stationfile, skip=4, nrows=12)
    ## 2. rearrange the data from wide to long
    dshape = reshape2::melt(d, id.vars="X")

    ## 2.5 add the station ID column
    info = read_station_info(stationfile)
    ID = info$stationID
    dshape$stationID = ID # will repeat over all rows

    ## 3. filter the year and any missing data
    dshape = dshape[!is.na(dshape$value),]
    dshape = dshape[dshape$value >=0, ]

    ## 3.1 set the name
    names(dshape) = c("month","year","rainfall","stationID")
    ## 3.2 convert the year to numeric
    dshape$year = as.numeric(substr(dshape$year,2,5))
    ## 4. return the data in long format
    dshape = dshape[dshape$year >= first_year & dshape$year <= last_year,]

    return(dshape)
}

d1 = read_rainfall(all_files[5])
head(d1)
```

```
      month year rainfall stationID
25  January 1852     5.43    RR1669
26 February 1852     3.00    RR1669
27    March 1852     0.13    RR1669
28    April 1852     1.01    RR1669
29      May 1852     2.83    RR1669
30     June 1852     2.72    RR1669
```

```
summary(d1)
```

```
    month                year         rainfall       stationID
 Length:228         Min.   :1852   Min.   : 0.080   Length:228
 Class :character   1st Qu.:1856   1st Qu.: 2.237   Class :character
 Mode  :character   Median :1861   Median : 3.130   Mode  :character
                    Mean   :1863   Mean   : 3.452
                    3rd Qu.:1866   3rd Qu.: 4.545
                    Max.   :1900   Max.   :11.810
```

6 Reading All The Rainfall Files

Now I can use the same approach used to build the station information data frame to build a large rainfall information data frame - “map” my `read_rainfall` function over the filenames and “reduce” it via `rbind` again.

```
read_all_rainfall <- function(rainfallfiles){
    return(
        Reduce(rbind, Map(read_rainfall, rainfallfiles))
    )
}

all_rain = read_all_rainfall(all_files)
head(all_rain)
```

```
      month year rainfall stationID
25  January 1852     5.43    RR1669
26 February 1852     3.00    RR1669
27    March 1852     0.13    RR1669
```

```
28    April 1852     1.01    RR1669
29      May 1852     2.83    RR1669
30     June 1852     2.72    RR1669
```

```
summary(all_rain)
```

```
    month                year         rainfall       stationID
 Length:1898        Min.   :1851   Min.   : 0.020   Length:1898
 Class :character   1st Qu.:1867   1st Qu.: 2.000   Class :character
 Mode  :character   Median :1880   Median : 3.120   Mode  :character
                    Mean   :1878   Mean   : 3.357
                    3rd Qu.:1890   3rd Qu.: 4.487
                    Max.   :1900   Max.   :11.810
```

7 Job Done!

Now we can complete the function we started earlier. It needs to get the full paths to all the data files under the input path, read the station info and the rainfall readings, and then save them to the output path.

You can join folders and file names with `paste` and `paste0`, but you need to be careful to make code that runs on different operating systems. The `file.path` function understands different systems and constructs paths that will work.

```
process_weather <- function(input_path, output_path){
    ## find all input files
    all_files = get_rainfall_files(input_path)

    ## read and process each input file into data
    ##
    ## first the station info
    stations = read_all_station_info(all_files)

    ## select years from 1851 to 1900 (inclusive)
    ## merge input data
    readings = read_all_rainfall(all_files)

    ## output readings.csv and stations.csv
    ## use `file.path` to construct paths
    write.csv(readings, file.path(output_path,"readings.csv"), row.names=FALSE)
    write.csv(stations, file.path(output_path,"stations.csv"), row.names=FALSE)

}
```

Now I'll test this by reading from the data folder and writing to temporary folder that you can get with the `tempdir` function:

```
tmp = tempdir()
message("Writing to ",tmp)
```

```
Writing to /tmp/RtmpVtfPlh
```

```
process_weather("./DATA", tmp)
```

We can test this by reading back.

```
stations_in = read.csv(file.path(tmp, "stations.csv"))
readings_in = read.csv(file.path(tmp, "readings.csv"))
head(stations_in)
```

```
                         name stationID      long      lat elev
1 LANCASTER BLEA TARN RESERVOIR      3048 -2.772221 54.01822  330
2            CATON SCHOOL HOUSE    RR1669 -2.700400 54.07300  170
3 CATON GREEN - observer Storey    RR1669 -2.690842 54.08073  250
4 CATON GREEN - observer Holden    RR1669 -2.690000 54.07980  250
5          CATON NEAR LANCASTER    RR1669 -2.709075 54.07433  160
6        LANCASTER ELLEL GRANGE     RR395 -2.792400 53.97760  125
```

```
head(readings_in)
```

```
     month year rainfall stationID
1  January 1852     5.43    RR1669
2 February 1852     3.00    RR1669
3    March 1852     0.13    RR1669
4    April 1852     1.01    RR1669
5      May 1852     2.83    RR1669
6     June 1852     2.72    RR1669
```

and maybe doing some summary tables, for example make sure we have readings from several stations, and that all the IDs of the readings are in the station IDs.

```
table(readings_in$stationID)
```

```
  3049 3050/5 RR1669  RR256 RR2797   RR81
    96    408    228    281    380    505
```

```
## expect TRUE
all(readings_in$stationID %in% stations_in$stationID)
```

```
[1] TRUE
```

As a final check, let's see if we have at least one reading for every month-year combination. If this is true, then a table of month-year should be 12 by 50 with no zeroes in it.

```
T = table(readings_in$month, readings_in$year)
## expect 12x50
dim(T)
```

```
[1] 12 50
```

```
## expect > 0
min(T)
```

```
[1] 1
```