Painlessly Merge Data into Actuarial Loss Development Triangles with R

Sam Castillo

December 31, 2017

Contents

Objective:

Create a method which easily combines loss runs, or listings of insurance claims, into triangles. Using only Excel, the common method is to create links between the excel files which must be updated manually at each new evaluation. This is prone to human error and is time consuming. Using a script to merge the files first and then create a triangle saves time and is more consistent.

Follow along with the data and Rmd file from github:https://github.com/sdcastillo/Loss-Development-Triangles

For a definition of a loss development triangle and why they are important, see Wikipedia: https://en.wikipedia.org/wiki/Chain-ladder_method

Methods:

I use the packages `plyr`, `dplyr`, `tidyr`, and `lubridate` to manipulate the data once it is loaded into R. The package `readxl` reads the excel files from the windows file directory. It is worth noting that all five of these packages are included in Hadley Wickham's `tidyverse` package. The package `ChainLadder` has a variety of tools for actuarial reserve analysis.

```
library(dplyr)
library(readxl)
library(plyr)
library(tidyr)
library(ChainLadder)
library(lubridate)
```

Step 1: Organize the Excel Files

Copy all of the excel files into the same windows folder, inside the working directory of the R project. It is important to have *only* the files that are going to be loaded into R in this folder.

On my computer, my file structure is as follows:

C:/Users/Sam/Desktop/[R project working directory]/[folder with excel claims listing files to aggregate]/[file 1, file 2, file 3, etc]

Using the command `list.files()`, R automatically looks in the working directory and returns a vector of all the the file names which it sees. For example, R sees the data files in the current working directory:

```
#setwd("projects/Excel Aggregate Loop/excel files")
wd_path = "C:/Users/Sam/Desktop/projects/Excel Aggregate Loop/excel files"
list.files(wd_path)
```

```
## [1] "claims listing 2011.xlsx" "claims listing 2012.xlsx"
## [3] "claims listing 2013.xlsx" "claims listing 2014.xlsx"
## [5] "claims listing 2015.xlsx" "claims listing 2016.xlsx"
```

R can then loop through each of these files and perform any action. If there were 100 files, or 1000 files in this directory, this would still work.

The loss files have dummy data for a claims listing. Each row is an individual claim which would have a member name, memberID, loss date, report date, paid loss amount, incurred loss amount, case reserve amount, etc. To make this as realistic as possible, the have artibtrary columns in addition to the ones which we need, `file year`, `loss date`, and `paid`.

```
file_names = list.files(wd_path)
file_paths = paste(wd_path, "/", file_names, sep = "")
head(read_excel(file_paths[4]) %>% select(-3,-4,-5))
```

```
## # A tibble: 4 x 6
##    name `license number`   age `file year` `loss date`  paid
##   <chr>            <dbl> <dbl>       <dbl>      <dttm> <dbl>
## 1  jeff                3    23        2014  2011-01-01   400
## 2   sue                3    43        2014  2012-01-01   400
## 3  mark                2    55        2014  2013-01-01   200
## 4 sarah                1   100        2014  2014-01-01   500
```

In order to evaluate the age of the losses, we need to take into account when each loss was evaluated. This is accomplished by going into Excel and adding in a column for "file year", which specifies the year of evaluation of the file. For instance, for the "claim listing 2013" file, all of the claims have a "2013" in the "file year" column.

Step 2: Load the Data into R

Initialize a data frame which will store the aggregated loss run data from each of the excel files. **The names of this data frame need to be the names of excel file columns which need to be aggregated.** For instance, these could be "reported", "Paid Loss", "Case Reserves", or "Incurred Loss". If the excel files have different names for the same quantities (ie, "Paid Loss" vs. "paid loss"), then they should be renamed within excel first.

```
merged_data = as_data_frame(matrix(nrow = 0, ncol = 3))
names(merged_data) = c("file year", "loss date", "paid")
```

Someone once said "if you need to use a 'for' loop in R, then you are doing something wrong". Vectorized functions are faster and easier to implement. The function `my_extract` below takes in the file name of the excel file and returns a data frame with only the columns which are selected.

```
excel_file_extractor = function(cur_file_name){
  read_excel(cur_file_name[1], sheet = "Sheet1", col_names = T) %>%
    select(`file year`, `file year`, `loss date`, paid) %>%
    rbind(merged_data)
}
```

Apply the function to all of the files in the folder that you created. Obviously, if you had 100 excel files this would still work just as effectively.

From the `plyr` package, `ldply` takes in a list and returns a data frame. The way to remember this is by the ordering of the letters ("list"-"data frame"-"ply"). For example, ff we wanted to read in a data frame and return a data frame, it would be `ddply`.

```
loss_run_data = ldply(file_paths, excel_file_extractor)
names(loss_run_data) = c("file_year", "loss_date", "paid losses")
```

The data now has only the columns what we selected, despite the fact that the loss run files had different columns in each of the files.

```
glimpse(loss_run_data)
```

```
## Observations: 21
## Variables: 3
## $ file_year   <dbl> 2011, 2012, 2012, 2013, 2013, 2013, 2014, 2014, 20...
## $ loss_date   <dttm> 2011-01-01, 2011-01-01, 2012-01-01, 2011-01-01, 2...
## $ paid losses <dbl> 100, 200, 300, 300, 350, 100, 400, 400, 200, 500, ...
```

```
head(loss_run_data)
```

```
##   file_year  loss_date paid losses
## 1      2011 2011-01-01         100
## 2      2012 2011-01-01         200
## 3      2012 2012-01-01         300
## 4      2013 2011-01-01         300
## 5      2013 2012-01-01         350
## 6      2013 2013-01-01         100
```

Step 3: Create Development Triangles

Finally, once we have the loss run combined, we just need to create a triangle. This is made easy by the `as.triangle` function from the `ChainLadder` package.

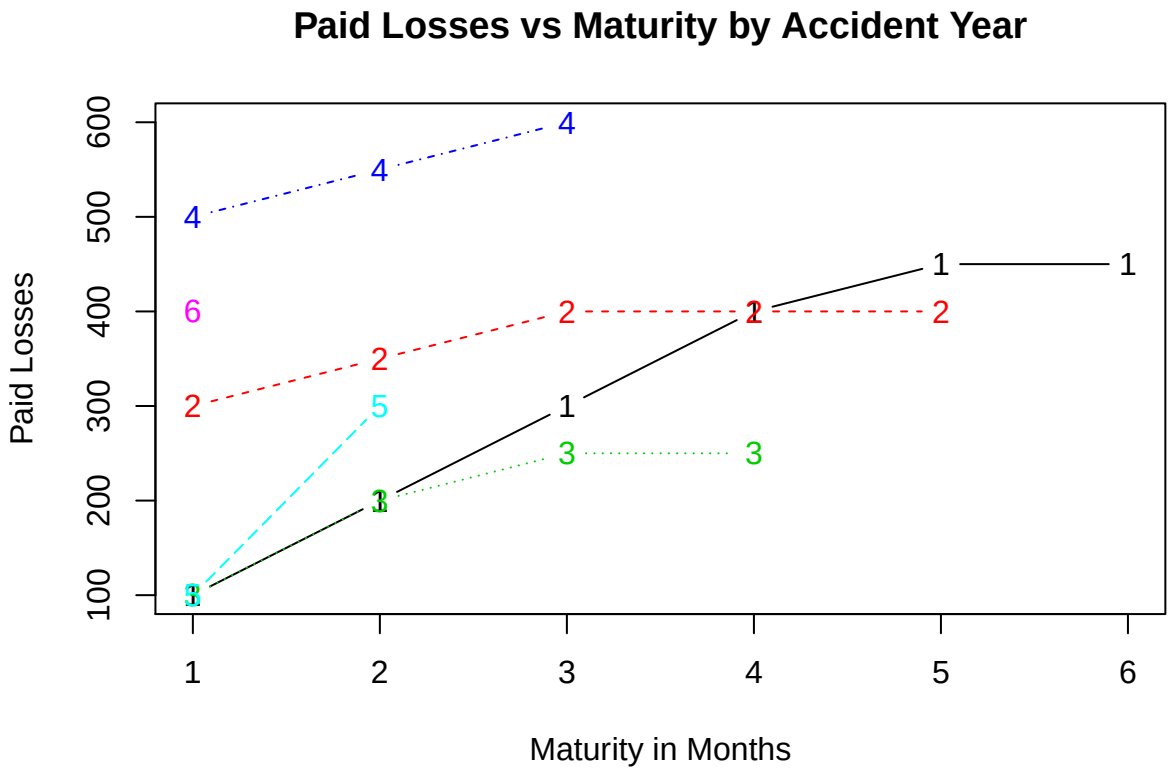
The only manual labor required in excel was to go into each file and create the `file year` column, which was just the year of evaluation of each loss run file.

```
loss_run_data  = loss_run_data %>%
  mutate(accident_year = as.numeric(year(ymd(loss_date))),
         maturity_in_months = (file_year - accident_year)*12)

merged_triangle = as.triangle(loss_run_data,
                              dev = "maturity_in_months",
                              origin = "accident_year",
                              value = "paid losses")

merged_triangle
```

```
##              maturity_in_months
## accident_year   0  12  24  36  48  60
##          2011 100 200 300 400 450 450
##          2012 300 350 400 400 400  NA
##          2013 100 200 250 250  NA  NA
##          2014 500 550 600  NA  NA  NA
##          2015 100 300  NA  NA  NA  NA
##          2016 400  NA  NA  NA  NA  NA
```

Within the package `ChainLadder` is a plot function, which shows the development of losses by accident year. Because these are arbitrary amounts, the plot is not realistic.

```
plot(merged_triangle,
     main = "Paid Losses vs Maturity by Accident Year",
     xlab = "Maturity in Months",
     ylab = "Paid Losses")
```



Conclusion:

When it comes to aggregating excel files, R can be faster and more consistent than linking together each of the excel files, and once this script is set in place, making modifications to the data can be done easily by editing the `exel_file_extractor` function.