Lab 1

Instructions

Your solutions must be written in an R Markdown (Rmd) file called `lab-01.Rmd`. This file must include your code and write up for each question. Your "submission" will be whatever is in your exam repository at the deadline. In order to receive full credit, your notebook must knit to HTML without any errors.

This lab is open book, open internet, closed other people. You may use any online or book based resource you would like, but you must include citations for any code that you use. You may not consult with anyone else about this lab other than the Professor or TA for this course.

You have until 11:59pm on Sunday, October 31 to complete this lab and turn it in via your personal GitHub repo - late work will not be accepted. Technical difficulties are not an excuse for late work - do not wait until the last minute to knit / commit / push.

On questions asking you to produce a graph, be sure to include an appropriate title and labels. Graphs should include a brief (one or two sentence) narrative describing the graph and what it reveals about the data. For this lab, you can only use base R (**do not use any third-party libraries such as `ggplot`, `dplyr`, etc.**).

Getting help

You are not allowed to post any questions on the Canvas Discussion board. Any questions about the exam must be asked during office hours or via email to the Professor or the TA.

Grading and feedback

The total points for the questions add up to 90 points. The remaining 10 points are allocated to code style, overall organization, spelling, grammar, etc. There is also an extra credit question that is worth 5 points. You will receive feedback as an issue posted to your repository, and your grade will also be recorded on Canvas.

The data

The data can be found inside the `data` folder in the main directory. `ppauto_pos` contains information on private passenger auto liability/medical claims for various property-casualty insurers that write business in the US. A description of each field is below:

- `GRCODE` NAIC company code (including insurer groups and single insurers)
- `GRNAME` NAIC company name (including insurer groups and single insurers)
- `AccidentYear` Accident year(1988 to 1997)
- `DevelopmentYear` Development year (1988 to 1997)
- `DevelopmentLag` Development year (AY-1987 + DY-1987 - 1)
- `IncurLoss_B` Incurred losses and allocated expenses reported at year end
- `CumPaidLoss_B` Cumulative paid losses and allocated expenses at year end

- `BulkLoss_B` Bulk and IBNR reserves on net losses and defense and cost containment expenses reported at year end
- `PostedReserve97_B` Posted reserves in year 1997 taken from the Underwriting and Investment Exhibit – Part 2A, including net losses unpaid and unpaid loss adjustment expenses
- `EarnedPremDIR_B` Premiums earned at incurral year - direct and assumed
- `EarnedPremCeded_B` Premiums earned at incurral year - ceded
- `EarnedPremNet_B` Premiums earned at incurral year - net
- `Single` 1 indicates a single entity, 0 indicates a group insurer

Questions

Question 1 (5 points): Load the data/ppauto_pos.csv into R and assign it to a variable `ppauto`.

Question 2 (20 points): Create a bar plot showing paid losses (CumPaidLoss_B) for State Farm Mut Grp by accident year at development lag 10.

Question 3 (10 points): Create a function, `incurred_loss_ratio`, which takes two parameters, `incurred_loss` and `earned_premium`. If `earned_premium` equals `0`, the function should return `NaN`, otherwise it should return the ratio of `incurred_loss` to `earned_premium`.

Question 4 (15 points): Using the function created in Question 3, add a new column, `IncurredLossRatio`, to `ppauto` which contains the ratio of incurred losses (`IncurLoss_B`) to net earned premiums (`EarnedPremNet_B`).

Question 5 (10 points): Which insurer (`GRNAME`) has the maximum, non-`NaN`, incurred loss ratio?

Question 6 (20 points): Create a boxpolot showing the distribution of `IncurredLossRatio` by accident year at development lag 10. *TIP: when creating your plot try setting `outline = FALSE` to prevent showing outliers*

Question 7 (10 points): What's the difference between `[`, `[[`, and `$` when applied to a list?

`[` selects sub-lists: it always returns a list. `[[` selects an element within a list. `$` is a convenient shorthand for `[[`.

Extra Credit (5 points): Insurance claims are often presented in the form of a triangle structure, showing the development of claims over time for each accident year (see example below).

```
##            1       2       3       4       5       6       7       8       9      10
## 1988 2439272 4722902 5705646 6238289 6519491 6677426 6750431 6787444 6808809 6815646
## 1989 2828267 5368026 6494597 7096377 7417869 7575814 7655217 7693240 7712077      NA
## 1990 3186948 5913490 7140613 7774615 8096374 8251086 8325184 8364955      NA      NA
## 1991 3192619 5878000 7075254 7698459 7996404 8140065 8215810      NA      NA      NA
## 1992 3561950 6474291 7763969 8404713 8716926 8876813      NA      NA      NA      NA
## 1993 3895076 7024867 8343417 9003120 9337099      NA      NA      NA      NA      NA
## 1994 4323103 7590944 8924062 9640098      NA      NA      NA      NA      NA      NA
## 1995 4491070 7664190 9006113      NA      NA      NA      NA      NA      NA      NA
## 1996 4444088 7486113      NA      NA      NA      NA      NA      NA      NA      NA
## 1997 4344144      NA      NA      NA      NA      NA      NA      NA      NA      NA
```

The most established method for predicting future claim payments (i.e. the `NA`s in the triangle above) is known as the chain-ladder algorithm. The first step of the chain ladder algorithm is to calculate the development ratios, f_k, from the loss development triangle from one lag period to the next. The formula to calculate f_k is below.

$$f_k = \frac{\sum_{i=1}^{n-k} C_{i,k+1}}{\sum_{i=1}^{n-k} C_{i,k}}$$

Where C is an $n \times n$ claim development triangle (matrix). For example, using the claim development triangle above, the development ratio from development lag 1 to development lag 2, f_1, is

```
f1 <- sum(C[1:9, 2]) / sum(C[1:9, 1])
f1
```

```
## [1] 1.795999
```

Create a function `development_ratios`, which takes a $n \times n$ matrix parameter, `C`, and returns a vector of the development ratios calculated using the formula above.