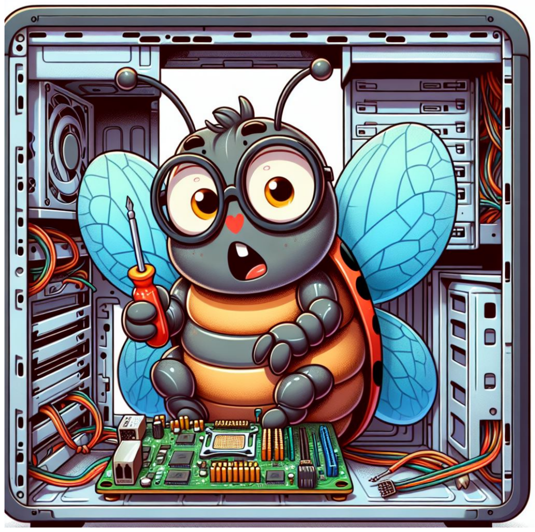
MATHS 7107 Data Taming
Assignment 2

Trimester 3, 2024


Source: Bing Copilot

1 Background

A large software company is hoping to optimise their software debugging process, and they asked you for some help to analyse the data (in exchange for a big pile of money).

The company has four divisions around the world, called "Division 0", "Division 1", "Division 2" and "Division 3". They have collected debugging information on a large number of software programs produced by each division. The programs have required vastly different amounts of work, and are of varying levels of complexity. Of course, larger and more complex programs are expected to have more bugs slip through, so it's hard to compare the data directly. Therefore, to make the data comparable, the company tries to account for this by measuring things in units of 1000 lines of code.

They'd like to get an estimated function for the amount of debugging time needed to achieve an acceptable number of bugs in a program. Using that function, they'd like an estimate of the average number of bugs left in their programs when doing the median amount of debugging work. Finally, they have a large software project coming up, expected to consist of 80,000 lines of code. They've set aside 4 weeks of debugging staff time (28 days of 24 hours), and they want you to predict how many bugs will be in the program once they've finished. In particular, they're hoping that the total number of bugs left is no more than 3,000. You'll need to provide intervals around your estimates, and (for some reason) the company wants the intervals at the 98% level.

Conveniently for you, they have just started using `R` and `R Markdown`, so they want your report as a PDF generated using `R Markdown`. The company's CIO studied Data Taming last trimester, so she wants you to only use commands from the course, so that she can easily see what analysis you've done. In your `R Markdown` code chunks: make sure that you **do not** set `echo = FALSE` so that she can see what `R` code you used to generate your output. But of course, she doesn't want to see irrelevant warnings or messages.

But remember that your report is for the CEO, who is not really a technical person, and who certainly doesn't know `R`. So make sure you include descriptions allowing the average person to understand what you are doing and what the output means.

1.1 Number of digits

When writing your own text, or **USING** the output from `R`:

- For integer results, report the whole integer.
- For non-integers with absolute value > 1: use 2 decimal places
- For non-integers with absolute value < 1: use 3 significant figures.

  For example:
  - $135.5681 \approx 135.57$
  - $-0.0004586 \approx -0.000459$

Exceptions:

- If you're just **PRINTING** the output from `R`, then just keep the output as it is.
  - But if you have `R` do the rounding for you then you need to conform to these two conventions listed above.
- If your data has fewer digits of precision than specified above (eg. because of the way it was stored in the original data, or because of the way it was calculated) then only report that level of precision.
- In some cases, the question may specify a different level of precision — in which case, do what the question says.

2 The data

The company has four datasets labelled `division_0.csv`, `division_1.csv`, `division_2.csv` and `division_3.csv` (one from each of the divisions). Each dataset contains 4 columns:

- `PID`: The data has been de-identified, so the program name has been replaced with this identifier. These numbers were randomly allocated to prevent re-identification of the program names.
- `LOC`: The number of lines of code in the program.
- `Debug Time`: The total time spent by the company debugging the program before it was released to customers. The units of hours and minutes have been rounded off to the nearest integer.
- `Bugs REM`: The total number of bugs reported by customers using the program. (So these are bugs that were still in the program after the company had spent `Debug Time` debugging it.)

Each dataset has data on 1,200 programs. There is likely to be some errors in the data, so make sure you clean it before you do any analysis.

3 Data cleaning

IMPORTANT!

Make sure you only remove data that you must remove. Do not just delete data because it is inconvenient. You must have specific instructions from the client, or it must be an impossible value, before you remove any data from your analysis. Even then, you need to describe why it was removed.

Only perform cleaning operations if you know there is a problem. (Performing unnecessary operations on data is a good way to accidentally introduce errors.) So make sure you have clearly identified the unclean piece of data before you clean it.

Instructions:

- There may be some duplicated rows, in which case, remove one of them.
- Some test data may have been left in. Remove it.
- Any negative numbers indicate that the data is corrupted. Remove the entire row.
- If there are any values that are impossible (in absolute value) then remove the entire row.
- There may be some other typos, so fix them if possible. If they're not possible to fix, then delete the entire row.

4 Your job

> **Note**
>
> Make sure you write text to explain what you are doing at each point and why you are doing it. You need to justify all the things you do or claim. Also describe the results. This report is for the CEO, so aim your explanations at the average person and avoid jargon wherever possible.

1. Load the correct dataset directly as a tibble (don't load it as a general dataset and then convert it, as that can introduce errors). Output the first 10 lines of the dataset and the dimensions of the data set.
2. Set the correct seed, then randomly permute all rows in your data set using `sample_n()`. Output the first 10 lines of the dataset and the dimensions of the data set.

   > **Note**
   >
   > Use this random permutation of the data from Q2 for the remainder of the assignment (ie. don't go back to the original data you imported in Q1.)

3. We want to clean up our data, but first we'll put in an extra column of row numbers, so we can track some changes we've made to the data.
   - Add a column at the far left of the dataset called `ROWS` that contains the row numbers.

   Output the first 10 rows of the dataset.
4. Now clean the data. Make sure you justify every step of cleaning that you do. Then display the first 10 rows of the dataset, and the dimensions of the dataset.
   - If you discover any problems with the data in the following questions then you should come back and redo this question before you submit. Your data should be clean and shiny from this point.
5. Let's tidy up the data a bit. Replace the `Debug Time` column with 2 new columns at the right of the dataset:
   - extract the number of hours from `Debug Time` and store it in `DB_HRS`
   - extract the number of minutes from `Debug Time` and store it in `DB_MINS`

   (Don't combine the numbers to give a total time, just pull out the numbers from the text. We are just tidying at the moment, and we'll do some calculations later.)

   When you've done these steps output the first 10 lines of the dataset and the dimensions of the data set.
6. Using dot points, identify what types of variables we now have in our data set, i.e., "Quantitative Discrete", "Quantitative Continuous", "Categorical Nominal", "Categorical Ordinal". (Don't just describe what data type they are in the tibble — you need to think about the type of variable in the context of the meaning of the data.) Make sure you provide some justification for your choice of variable types.
   - Don't just provide vague statements, but be very concrete about describing this particular set of data.

7. Now it's time to tame our data.
   - Make your data set correspond to the Tame Data conventions on page 3 of Module 2. You'll need to use your answers to Q6.
   - Also make sure the `R` data types in your tibble match the variable types that you identified in Q6.
   - *(Reminder: Your data should already be clean by this point. You may want to check here if there is any more cleaning required. If so, go back to Q4 and try again.)*

   Output the first 10 rows, and the dimensions, of your clean, tidy and tame data set.

8. Making sure you set the seed correctly choose a random sample of 700 programs from the dataset, and order them by the row numbers that we introduced in Q3. Then output the first 10 lines of the dataset and the dimensions of the data set.

   Note

   Use this random subset from Q8 for the remainder of the assignment.

9. (a) Now let's get on with some analysis. Add four new columns to the data set:
   - `db_totalh`: the total time spent debugging the program, in hours.
   - `db_totalm`: the total time spent debugging the program, in minutes.
   - `time_per_loc`: the total time spent debugging the program, in minutes, per 1000 lines of code.
   - `bugs_per_loc`: the number of remaining bugs, per 1000 lines of code.

   and remove the columns containing the hours and minutes from the old `Debug Time` variable. Output the first 10 rows, and the dimensions, of the data set.

   (b) Describe what type of variable these four new columns represent ("Quantitative Discrete", "Quantitative Continuous", "Categorical Nominal", "Categorical Ordinal"). Are the data types correct in the tibble? (Explain your answer.) If they are not correct, make sure you change them.

10. (a) Use `inspect_num()` to display the summary statistics for the numerical values in your dataset.

    (b) Report the following statistics:

    i. The median debugging time (per 1000 lines of code), to 2 decimal places.

    ii. The IQR of the number of remaining bugs (per 1000 lines of code), to 3 signficant figures.

    iii. The program ID and number of lines of code for the longest program in the dataset.

11. We want to understand how many bugs are remaining in the code. So for the first look at the data, plot the histogram of the `bugs_per_loc` data and find the skewness. Comment on the shape of the histogram.

12. (a) Now make a scatterplot to see if `bugs_per_loc` is related to `time_per_loc`. Put the independent/explanatory variable on the horizontal axis, and explain why this is the explanatory variable. Include a straight line of best fit on your plot.

    (b) Does it look like there is a linear relationship between the two variables? (Provide some reasons for your answer.)

13. We would like to fit a linear model of `bugs_per_loc` against `time_per_loc` for this data. But we will first apply a Box-Cox transformation.

    (a) Use the Box-Cox algorithm described on page 7 of Module 5 to obtain an estimate of λ. (Extend the range of the search for $-5 \leq \lambda \leq 5$, in steps of 0.1.) What is the estimated λ?

    (b) Apply the transformation to create a new column called `tf_bugs` on the right of your dataset.

    Output the first 10 rows, and the dimensions, of the data set.

14. Produce a scatterplot of the Box-Cox transformed data (with a line of best fit), as well as a histogram of the response variable and the corresponding skewness. Write 2–3 sentences about this output and how it compares to the untransformed data.

15. We will now try to fit a linear model to `tf_bugs` using `time_per_loc` as the predictor.

    (a) Write down the general equation for the true linear model when fit to the entire population. Make sure you define all of the notation you introduce. *(Hint: this equation should include the error terms, and contains the true parameters.)*

    (b) Now build a linear model in `R`, and use the output to find estimates for the model parameters. Use these estimates to write down your estimate for the true linear model. Make sure you use the correct notation, and also define the notation. *(Hint: you should have three estimated parameters.)*

16. Before we use our model for anything, we need to check if it satisfies the 4 assumptions for a linear model (as described on pages 10–13 of Module 6). So now check if our model satisfies these assumptions. Importantly, the client needs to understand the implications, so as part of your answer:

    - Describe in your own words what each assumption means in terms of the specific linear model you have fit.
    - If you refer to any graphs make sure you describe (in plain language) what is on the horizontal and vertical axes of those graphs.
    - Give an explanation of why each assumption is, or is not, satisfied.
    - Make sure you identify at least one possible problem with the **Independence** assumption.

    (Note that we are going to use a linear model regardless of any problems that you find in the assumptions, but it is always good to highlight any shortcomings of the model so the client knows about them.)

17. (a) Use your model to find the mean number of bugs remaining after the median debugging time (in minutes, per 1000 lines of code). You also need the correct interval.

    (b) Then use your model to predict the number of bugs remaining for the company's new software project. Again, you also need to provide the correct interval. Pay attention to the units.

    (Hint: you will need to transform your predictions and intervals back into the scale of the original variables.)

18. Write a paragraph or two describing what you have found about the company's debugging process, and providing answers to their specific queries. You can also discuss any observations or conjectures that you have. (Everybody's data will be different so there is no right or wrong answer here, as long as you justify your claim with reasonable arguments.)

That's enough for this report. We might investigate this data further in Assignment 3. (Then we can charge the client more money for another deliverable!)

5 Submission

You must submit your assignment via MyUni. Do not email it to the teaching staff. Detailed instructions are on the assignment submission page in MyUni. Make sure that all your output is relevant to the questions being asked.

6 Deliverable Specifications (DS)

Before you submit your assignment, make sure you have met all the criteria in the **Deliverable Specifications (DS)**. The client will not be happy if you do not deliver your results in the format that they've asked for.