Reflective essay on tier 1: the parser & database connection tier

**Approach to the project**

Explain how you set about the project detailing:

Interaction with the team

* how you interacted with the other members of the group

Starting with the project approach and initial thoughts, it was simple to divide up the tasks as we had a relatively clear view of what we would each be good at or have the skills to do. However, 6 weeks or so into the project, Elizabeth left the course, and so we were left with just myself and Michael. Prior to Elizabeth leaving, we had set up a group workspace on Slack.com, which myself and Michael continued to use throughout the project. It was very useful in that it allowed us to have web chats and video conferences as well as share code snippets. We used this to document our correspondences, pass on thoughts and arrange meetings. It proved to be a very effective productivity tool.

Overall project requirements

* how you (as a group) came up with the requirements for the overall project

For our initial meeting, we assigned roles and discussed the structure of the project as a group. We made sure everyone understood the concept, and the function of the final product, as well as what exactly each person had to do. We also discussed some intricacies of implementation that we felt uncertain about, and what we thought may be a suitable aesthetic for the webpage. The first meeting was preliminary, and we waited until we had all started coding in order to iron out more specific aspects of our respective components. At this point, we hadn’t decided on any version control systems to utilise. Our respective parts were decidedly separate. However, a few weeks in, we had developed enough of our individual components and had enough direction to enable us to pool our work. Following the GitHub lecture, we created a repository and from then on used that to manage the project.

Requirements for my contribution

* how you (as an individual) came up with the requirements for your contribution

My task was to design and create the SQL database. It was also my job to parse the data file and port the data into this database. In order to determine the requirements for my tier, I had to examine the text file for chromosome 15 along with the project guidelines to work out what exactly needed to be extracted. I created a list of all the components I needed to extract, along with their corresponding format in the file. Key format information included: surrounding characters, whether it was a section or subsection (i.e. beginning of line, or not), length of item, data cleanliness (i.e. are there stripping steps I need to perform to ‘clean’ the data). After this, I began the process of searching for anomalies and things that would trip up any code that was very general or sweeping. Clearly, it was impossible at this stage to identify all of these cases, so the anomaly list was far from comprehensive. The rest of these were dealt with in a trial and error manner throughout the development process. As well as the dealing with the constraints provided by the raw data file, I also had to consider constraints coming from the database layer. In order to ensure I was working towards the correct endpoint, I created the database schema during this initial preparation phase. I did not parse the data and then try to design a database to fit it, instead I considered the multi-directional constraints put on the parsing process by the tiers above and below it. This informed my subsequent code development phase.

**Performance of the development cycle**

The development process

How well did the development cycle work within the group?

For the development cycle, we used Slack.com and were in constant communication. We highlighted anomalies that the other may have missed in the raw chromosome 15 file and informed each other about data formats to ensure eventual compatibility of the tiers. We had no issues with this. However, it was sometimes useful to have face-to-face meetings to hash out larger problems and misunderstandings (of which there were very few).

You should document the development process - what stages did you go through and what steps did you take? Did you go through several iterations or did you spend a long time in design and then a relatively short development stage?

I spent some time on planning (as discussed above), but perhaps I could have saved more development time by having a more comprehensive plan. Although, an entirely comprehensive plan would have been challenging, as the process highlighted problems as we went along, requiring us to make alterations and fixes that we could not have predicted.

My initial code was clunky and long. The final code is a result of a process of iterative refinement and compartmentalisation. Initially my code was very degenerate, having the file iteration process written out separately for extracting each list of items. This did not work very well – it lead to non-uniform collection of the data and allowed room for bugs. Condensing all this code into a two functions – the findall\_matcher function and the match\_finder function made the remainder of the development phase much easier, and reduced the opportunity for bugs to arise through inconsistencies. I could then alter the core parsing functions once as opposed to multiple times.

**Code testing**

What strategies (if any) did you take for testing your code - particularly if other people's code wasn't ready?

Of key importance was for me to test my code for the format of the output. The extraction format was key to the entire project, as this would determine the data manipulation that could be performed in higher tiers (namely the middle tier). I did this as I went along, at each major change testing the list formats – examples of formats at different stages can be observed in the notes throughout the paser.py file. At the end of the parser.py file, there is a length test script at the bottom. This was to inform me of any changes to the length of my lists. The lengths all had to be 241 items long, or else the data would not have aligned when I combined the lists into dataframes. To test the functions, I used small subsets of the data for which I knew what the expected outcome should be. This dataset included extreme examples that I picked up on during my study of the chromosome file, enabling me to test that my functions were able to cope with these.

**Known issues**

Are there any known issues or bugs, or anything that doesn't work as specified?

In terms of known issues, the main thing that I was uncertain about was how to identify the exons that span multiple genes. In the code comments I defined these exons as those with the following format:

['U59692.1:2089..2187', 'U59693.1:710..809', 'U59693.1:1858..2093', 'U59693.1:2465..4329', '344..1028']. This example shows a list of exon ranges for a particular gene ID. As there are multiple IDs associated with the exons from this gene, I based this section of the parser on the assumption that any gene with its exons in this format had exons spanning multiple genes. My strategy was to design a regular expression that would target genes with this exon format and, instead of storing exon range data in the table, the phrase ‘exons span multiple genes’ was inserted into the exon start and end columns of the database for these entries. If this assumption was incorrect, then this could be a source of error in the parsing tier.

**What worked and what didn't - problems and solutions**

You should discuss what worked well and what could have worked better - both with your code and the group interaction.

Were there any particular problems?

Conversely were there any solutions or ideas that you were particularly proud of?

The solution that I am proudest of is a small script in the parser.py file that utilises a list comprehension to solve the problem of the exon\_start and exon\_end lists containing lists of exon start and end points respectively – so that each gene was associated with a single row containing lists for the start and end points of its exons. Michael required the Coding\_region table to be of following form for the ease of his middle-tier data processing:

GENE EXON START EXON END

Gene1 12 182

Gene1 73 200

Gene2 28 100

Gene2 12 19

Gene2 200 250

… … …

So the solution I used was the following:

zipped\_id\_start\_end = [(id, v1, v2) **for** id, val1, val2 **in** zip(gene\_ids, exon\_start, exon\_end) **for** v1, v2 **in** zip(val1, val2)]

This is a neat piece of code, and allows – in a very succinct way – for the layout of the Coding\_region table, and the association of a single Gene\_ID with multiple exon start and end points. This is a good example of where myself and Michael’s communication and collaboration efforts were very successful and led to better implementation of the task.

Conversely to this, what did not work so well was the time management aspect in terms of our attempt to produce a finished project by way of taking on Elizabeth’s tier as well as our own work. Michael was to write the HTML for the website – which he did to some extent – and mine was to write the CSS script for the website, which I failed to find time to begin. Perhaps with a better timetable for the project work, we could have completed these additional tasks.

**Alternative strategies**

Were there alternative strategies you could have used? (Either for the project design or implementation)

**Personal insights**

What do you think you gained out of the project?

How has this experience helped you?

What experience or insights have you gained?