# **Submitted files and papers:**

1. findingCom.Rmd: contains R code
2. findingCom.HTML: the document demonstrating a solution to the problem
3. sqm.r: the R code for self-organized map
4. CSV files:
   1. Jira.csv: Data from Jira
   2. Effort\_on\_repos: Coding effort on repository
   3. Effort\_on\_tasks: Effort on tasks
5. RData files: contains the data loaded into Rmd script

# **Train of thoughts:**

I don’t have much experience working with the graph and network though I did one or two assignments about it when I was in my Master degree. When received the problem, I was excited but a bit intimidated. As usual I divided the big task into smaller pieces so that I could work on each of them easily and effectively.

First of all, I skimmed the paper quickly to understand what technique that I need to use for the problem. The paper is not too difficult but the demonstrated methodology is not much clear. After that, I search for igraph and what it can do when it comes to the problem of finding communities in a weighted network. I could see there are a few algorithms pre-installed in the library which would be useful for me to test. Also, on the internet, I found an algorithm using self-organised map for finding communities in weighted graphs and it was proved to be even faster than fast greedy. I decided to use these first. Unless the performance is high, I will think about installing the kuramoto algorithm.

Understanding the data is big, I only used 5% of data at the beginning for a test with different algorithms. The graph was firstly built as an adjacency matrix. The computer was freezing and taking about 0.5 hour for fast greedy algorithm. Knowing that this is not working, I started going back to Big Data techniques which I learned from a course in UC San Diego specialisation such as NeoJ and PySpark. NeoJ is not very much useful, so that I started writing the code in PySpark and trying to use GiraphX.

In the meantime, I realise that I could represent the graph in an edge-list and remove any edge with weight as 0 (i.e. no connection). By doing such, the graph reduces its size by 90%. I stopped investing my time on Big Data, but coming back to R with igraph and started increasing the size of samples for each experiment. Fast greedy works very well even with the full dataset though it takes maximum memory of the computer to build the graph.

The second and the last task were done quite quickly based on the result from the first one. I was quite surprised at first to know that these datasets actually do not have much in common. Therefore, I came to a conclusion that we could swap the dataset to re-build the community (using the same technique) and evaluate the models based on customer’s further requirements.