Part1: Survival Analysis in Plotly

Assignment was to plot something using plotly related to survival analysis.

I used the cancer dataset from Survival package. It contains the information about 228 patients with advanced lung cancer from the North Cancer Treatment Group. This dataset is interesting to me as it has many daily life practical attributes against which lung cancer can be analyzed upon.

I plotted two graphs in plotly. First one is Survival vs time when all the attributes are considered. Second one is gender wise survival rate.

Links to plotly are:

1. First graph: Survival vs Time

<https://plot.ly/~sarthak12/2/survival-vs-time/>

1. Second Graph: Gender wise - Survival vs time

<https://plot.ly/~sarthak12/4/survival-vs-time/>

I have submitted the following files:

* **SurvivalAnalysis\_plotly.R** - contains all the code.
* **SurvivalAnalysis\_plotly.html** - Detailed HTML report about the analysis.
* **SurvivalAnalysis\_plotly.Rmd** – R Markdown file for the above HTML file.

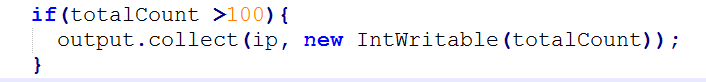
Part2: R Exercises

I have submitted **R\_Exercises.docx** file which contains the code and screenshots of the output.

Part3: Lab3

Use MapReduce to filter the IP addresses which occurred more than 100 times and sort them in decreasing order

For the first part of the problem where IP address having count > 100 needs to be filtered, I modified the IpReducer.java and included the following condition.

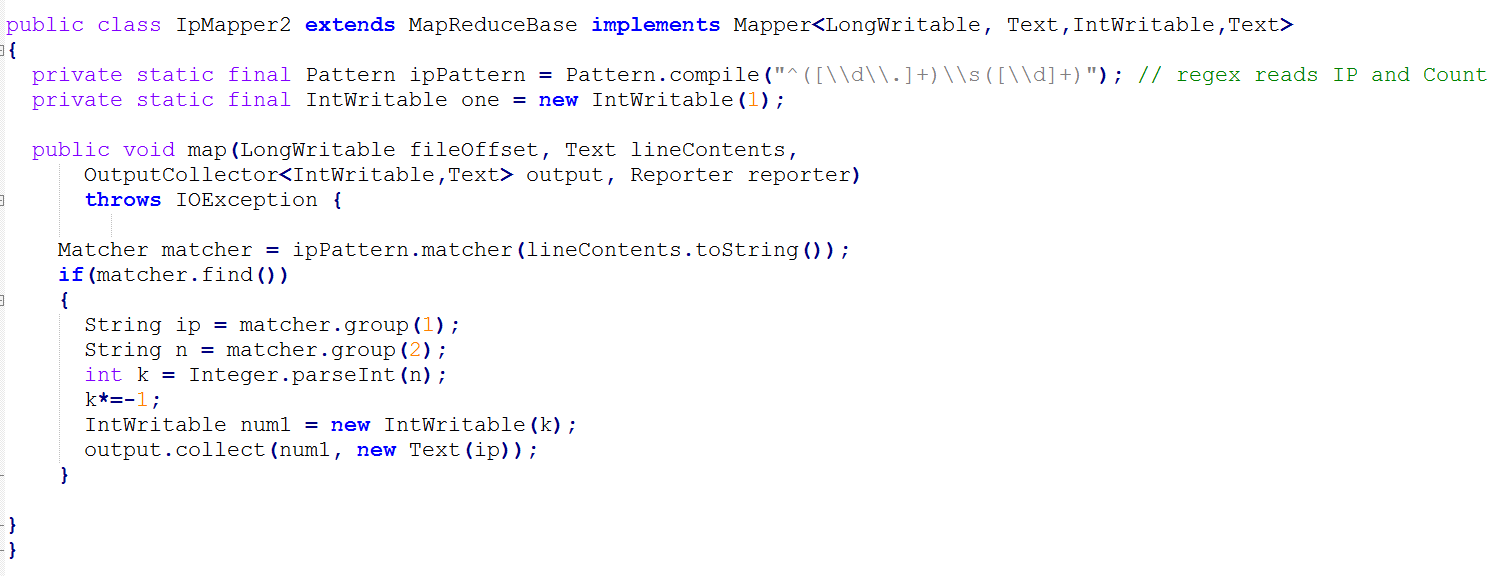


For the second part of the problem - the output needs to be sorted in decreasing order.

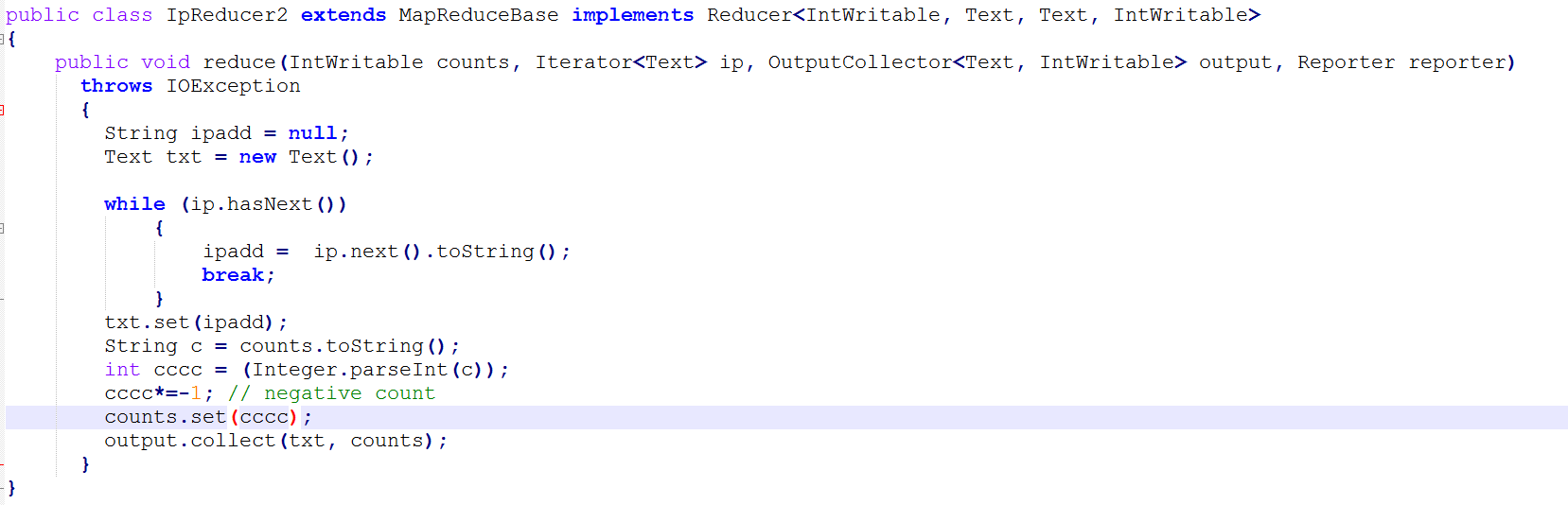
**Logic:**

I created a second mapper and reducer. The output of the first reducer is input to the second mapper. Using regex, the IP and count were passed as key-value pair in the second mapper. Then reversing the order and tricking the MapReduce, I passed the count as key and IP as value to the Reducer. This is necessary because MapReduce sorts the output based on key and not on value. Since by default it is sorted in ascending order, to sort it in descending order I made the count as negative before passing it to the second reducer. Finally, the count is made positive again before outputting.

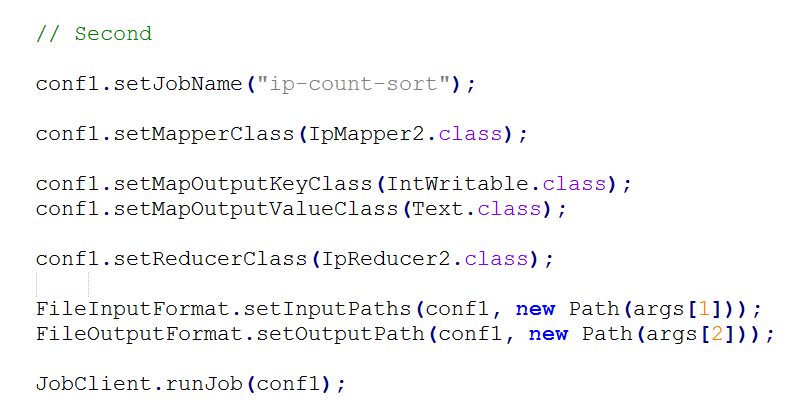
Second Mapper:



Second Reducer:



Reducer: For second Mapper and Reducer



**Output:**

