**ECS640U**
**Big Data Processing**
**Coursework Two**
**Big Data Analytics Group Project**

Mitema Paulker Emmanuel (120694961)
William Seaford (120170465)
Stefan Tod (120272714)
Mohamed Sharif (120232578)

## Introduction

The aim of this project is to analyse the success factors of the popularity of games and streamers in the video streaming service Twitch.tv with two main questions asked. “How will you detect the top streamers/games, vs the non- popular ones (in MapReduce terms)?” and “What factors you think you can extract from the data/additional sources that might have an influence in popularity? Keep in mind that over the 10 month dataset, we collect all the active channels every 30 mins.” Therefore we have created MapReduce jobs that will enable us to collate specific results to analyse and conclude factors connecting game popularity and views.

## Description of MapReduce Jobs

### Job: Most_Viewed_game

**Rationale:** We thought it was important to select a few games to use as our sample in analysing the factors affecting the popularity of games and streamers. The output of this job is a list of games in order of their views.

**Description:** We have filtered the mapper by removing all the spaces in the game names to ensure consistency and then split the data by tabs. In the .csv file we have noticed that some of the fields are empty hence we only processed lines that have 7 fields containing data by filtering using “If(line.length ==7)”. We then used a try and catch statement that contains code for converting a string into an integer. This is important to catch some abnormal values that can be in the .csv file. Finally, we wrote out the gameName as the key and the viewers as the value in the mapper for all the games that do not have an empty gameName field and zero as the number of viewers. In the reducer we basically add up all the sums of the viewers for particular keys, then wrote out the gameName as the key and the final sum of viewers as the value.

### Job: Streamer_Viewers

**Rationale:** To get the results for streamer_viewers we selected a sample of games consisting of games with the most views, middle ranged views and least value in viewership. This selection will enable us to collect the streamers with the most views from a fair representation of the sample pool. This is important as an accurate assessor of the popularity of a particular game.

**Description:** Here we have created an array to hold the games we have picked. Then we filtered the twitch data repeating removal of spaces and selection of dataset with 7 fields as done in the previous MapReduce job. We created a try and catch to handle abnormal values while converting from String to an Integer to get the values for the number of viewers. Using a “For Loop” and an “If Statement” we check if the gameName is equal to any of the games we have selected. In cases where this is true the Stringpair was set to hold the

streamer and the game name as a dual value.Then we wrote out the the pair as our key and the value as the viewers.

**Job:Game_Delay**

**Rationale:** We believe the delay will have little or no effect in the popularity of a game/streamer as the delay is the value of delay observed from api, indicating the delay value to viewer. However we created a MapReduce job for games with varying viewership as the key and their average delays as the value to help us decide if the viewership of a game is independent of the delay. Also in reality the delay value does not affect the viewers choice in viewing any game/channel as this information is not necessarily available to the viewers.

**Description:** We have filtered the data as we did in previous MapReduce jobs, creating a StringPair holding game name and delay.The mapper writes out an Inwritable of 1 as the key and the StringPair as the value. This is important so that all the data from the mapper go to a single reducer that computes the average values of delays for each game. In the reducer we create an arrays of “Sum” and “Count” to hold the sum values of delays and count values respectively.Then using a for-loop we divide sums by their corresponding count values and then we write out names of games as key with their corresponding average delay as values.

**Job:Game_Timestamp**

**Rationale:** We wished to see if we could establish a relationship between games and timestamps of viewing to see if we could analyse and discover any trends of a game over a period of time. We believed that certain games would spike in viewership from certain events such as tournaments. We created a MapReduce job which outputs the game and what hour of the day they were viewed on a 24 hour clock. We believed the viewership would be higher at night between 10pm and 4am where over in America it would be between 4pm and 11pm.

**Description:** We decided to create two different MapReduce jobs to show the relationship between the game and the timestamp. First, we created a MapReduce job which looked at the dates of when games were viewed. In the mapper we filtered out all games which did not match the specific game name we chose or contain a value in the timestamp field. This filtered out a significant number of results which is an issue covered in the constraining factors section. We chose to use month as our time period as by using day of the month collected too many results which were difficult to analyse. We collected a substring of the timestamp containing the month and then created a string containing the game and date and mapped this as the key alongside a value of 1 in the outgoing key/value. We repeated this method in the hour dataset collection changing the substring to contain the hour instead of the date.

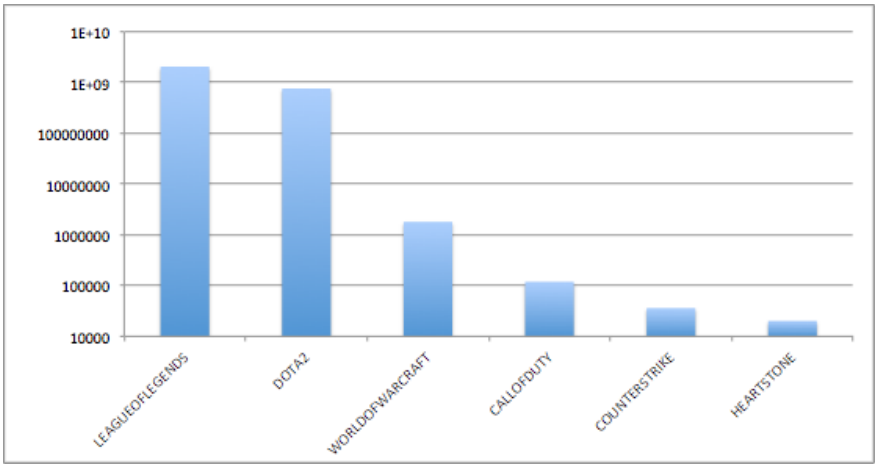
**Job:Streamer_TimeStamp**

**Rationale:** We wanted to see if we could see a relationship between the streamer of a particular game and the timestamp of the viewing and analyse any trend or significant feature of the data. We decided to replicate the previous Game_Timestamp MapReduce job in order to do so.

**Description:** We replicated the method from the previous MapReduce job to include an additional string array containing the names of two popular streamers for each of the games. After the conditional statement checking if the game matches the game we are searching for, the program will check if the game's streamer name we are looking for is contained in the display name field. We used String.contains() instead of String.equals() as the the full dataset was not being included and omitted some relevant results once again covered in the constraining factors section.

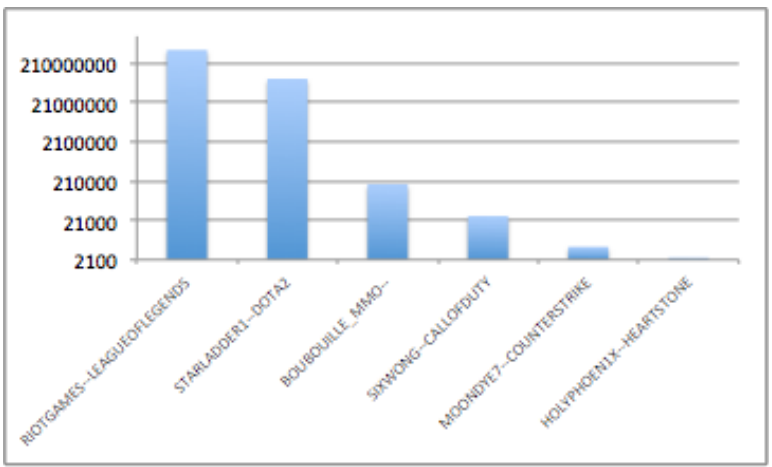
## Results Analysis

### Most_Viewed_Game




The first graph shows the number of views for each game we have selected. The graph also shows that the games were not randomly selected completely. There are sample selection of games which are used and show to have high, middle and low number of viewership. These games were selected so we could infer results from games with different popularities. This could lead us to make more holistic conclusions as well as specific analysis of particular patterns and trends amongst the viewers of twitch.tv.

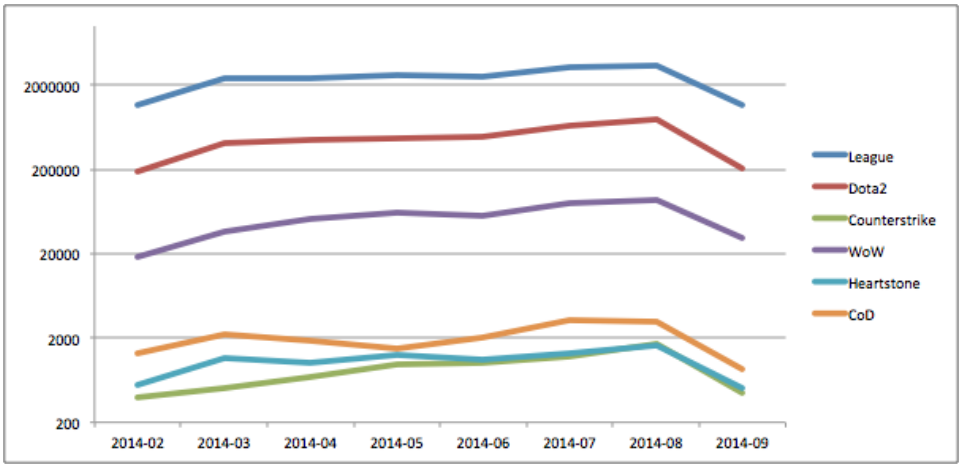
### Streamer_Viewers




We establish a connection between the previous graph and can infer that streamers with the most viewers are streaming games which have the most viewers. Hence we can conclude that the

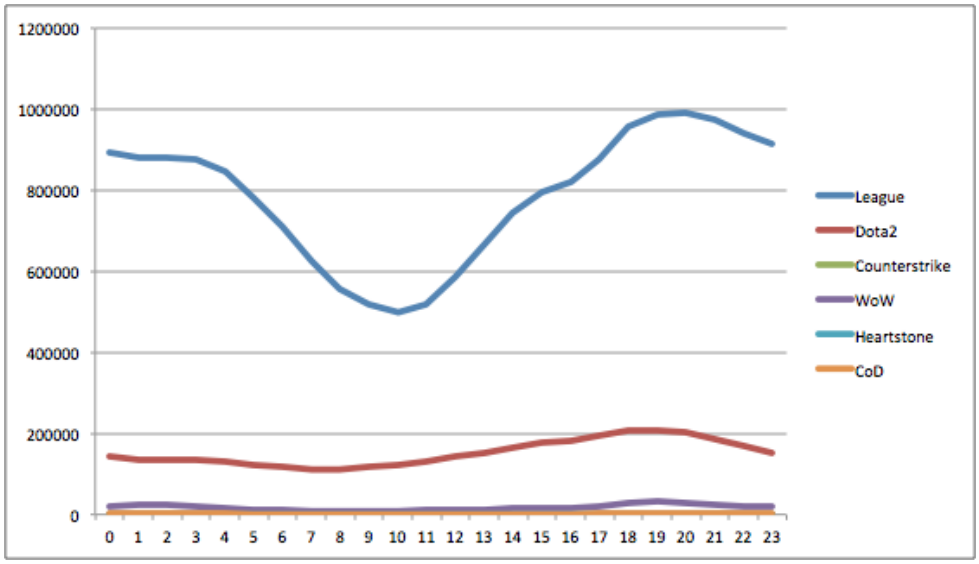
popularity of a streamer is affected by the popularity of the game they are streaming and the popularity of a game has a direct relationship on the viewership.

**Game_Timestamp**




We found in February there was a significantly lower amount of viewers amongst all the games but from March through to July there is a steady trend of viewers through the dataset. However this changes in July and August where there is a rise in viewers amongst all games included. We attribute this to the summer holidays in which children are home from school to spend more time on Twitch. This rationale is supported by the September data fields in which there is a plummet in viewership across the board as children return to school.

**Game_Hourly Timestamp**

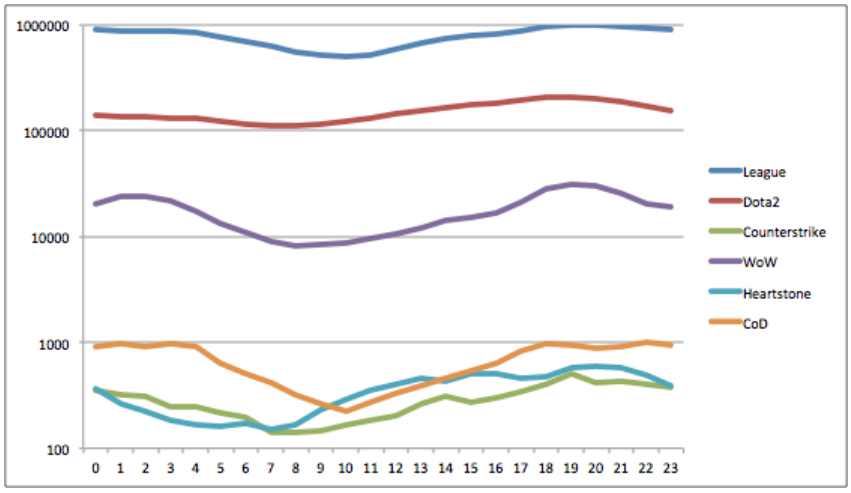



In the graph on the left we looked at the viewer timestamps specifically for League of Legends and observed there is a large rise and fall of viewership. If you shift the hour axis six hours forward. The large dip happens to be around 12am and 5am when most people are asleep in America. The peak of the viewership generally happens around 6pm and midnight in Europe, specifically the UK. This correlates to around midday to 6pm in America, a favourably active period of time where viewership can easily increase. This graph and the evidence it

suggests correlates with our hypothesis that a significant amount of viewers come from America.

Of course we can also explain the dip and rise in League of Legend's viewership by applying the UK circadian rhythm towards this graph as well. The fall in viewers from 3am until the figures reaches the wave's trough at roughly 9/10am can be explained by UK viewers signing off to go to sleep. The rise in viewership from 10am onwards can be explained by viewers in the UK who went to sleep earlier waking to come back onto the streaming site. The peak at 8pm could be explained as 8pm is a period where many people would be coming back home from work and school to consume an activity to help relax for the evening.

We think a combination of America and the UK's viewership pattern provided a multiplicative factor for the fluctuation between the highest viewership number and the lowest and can be used as an explanation for the cause of these particular numbers in viewers for League of Legends.




The logarithmic graph on the left supports the previous paragraphs statement and shows that all streams tend to have a fall in viewers between the hours of 3am and midday. The lower viewed games are show to have a higher fluctuations of viewers as their viewership is smaller and so more statistically receptive to changes in number of viewers.

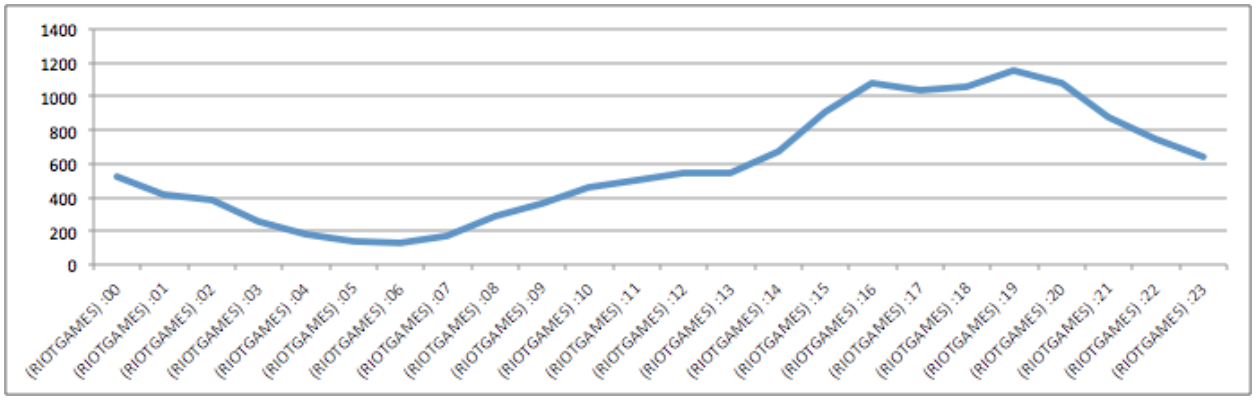
We can gather that the peak of viewers on Twitch.tv can be seen around 8pm and the lowest can be seen around 8am generally. However, some games have a slightly different pattern such as Call of Duty. Call of Duty hits the trough at 10am, 3 hours later than most other games. It also seems to plateau from 6pm until 4am. This plateau tells us that Call of Duty streamers have a loyal fanbase which consistent viewer number can be seen.

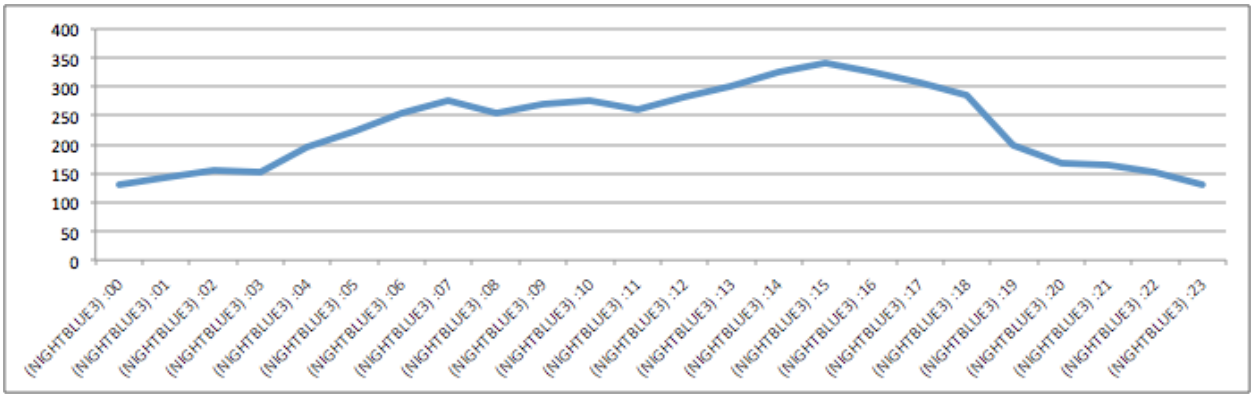
Looking at the graph for Counterstrike. We can assess that it is the least popular game in terms of viewers. Whilst Heartstone is more popular, it is not by a significant margin with both games showing similar viewer numbers. Whilst Counterstrike and Heartstone have a less steep decline in viewers. Call of Duty experiences a much sharper decline. This may be because it is a popular game, but not as popular as League of Legends, Call of Duty viewers may follow a similar schedule. Perhaps there is tighter sense of community within viewing

online for the Call of Duty fanbase and so when one comes online, a whole fanbase does. The data can become redundant in this scenario as it cannot explain why.

World of Warcraft can be seen as an example of the large fluctuation between its highest amount of viewers and lowest. This is unusual as World of Warcraft is an old game. We have concluded this is because the peak time of 7 and 8pm is when collective party attempts a run of a particularly difficult map termed “raids” and are most active in this period and so a lot of players come online to only play that. The second peak at 1am could also contribute to being a second raid time slot. However, there was a new expansion pack recently released in December 2014, after the data was collected. We could know if there was a resurgence in viewership which could be attributed to this factor as some players like to play fleetingly around the time of expansion releases.

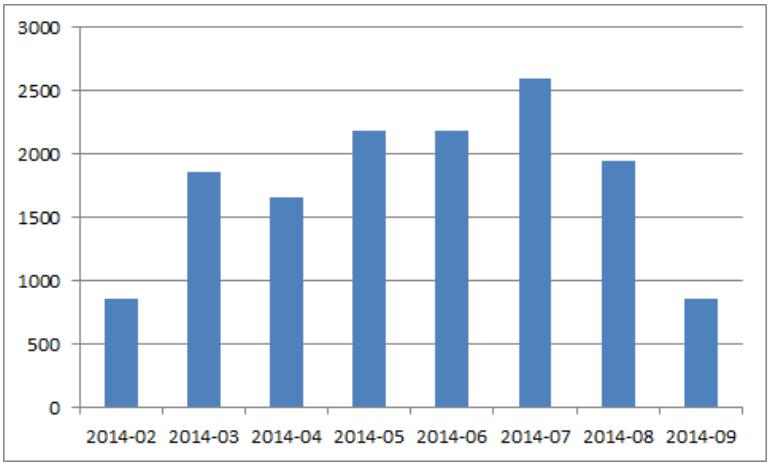
**Streamer_Timestamp**




Comparing the above graphs for the two streamers we have used in League of Legends. We can assume that Riotgames is more popular based on attainment of higher views. This correlates to the fact Riotgames is a channel hosting skilled gameplay which attracts more viewers.

Another observation from the comparison of the graphs tells us that whilst Riotgames attracts more views, Nightblue3 has a more consistent amount with a rough average of 175. This tells us that Nightblue3 might have a constant live stream in which there are always a small number of viewers watching. Or that any old clips are popular to be looked at by its fanbase for entertainment. Riotgames on the other hand has a larger fluctuation in viewers which can range from 100 to 1200. This tells us that whilst they have a peak with a large amount of viewers, they do not have as loyal fanbase in terms of keeping a steady amount of viewers on the channel.




We could not establish any particular trend to do with streamers and their timestamp other than the graph on the left which is the monthly views of Riotgames. One attribute we can pick out is the peak of viewership in July of 2014 where it exceeds 2500 viewers. We have concluded this is because of the tournament Riotgames participating in and so attracted higher amount of views. This validates our hypothesis that tournaments draw in larger viewing numbers.

## Constraining factors

**Unidentified Streamers:** Some games can be streamed by robots or malwares which can affect viewership in terms of how attractive the channel appears. In such cases we do not have any a control in the data we received. However there are possible ways to limit such errors as to have a database that pre-defines regular genuine streamers for games to enable us to compare our data with the database and remove any irregularities.

**Game_Delay:** We ran our .Csv files and found out that the delay contained 0 values throughout the dataset. We created a MapReduce job that writes out relevant data which contained delay fields with values greater than 0 but was subsequently not able to gain any results. So there is no way to use the delay as a factor to analyse the popularity of a game as the data is abnormal because delay values cannot be zero.

**Quantity of data collected**: The dataset we were given to analyse would offer good information in terms of the data collected but a limiting factor would be if too much data was collected making it difficult to properly analyse and conclude any patterns and trends. Such example as when we tried to analyse the timestamp in terms of days. Having 8 months of

periods resulting in 224 - 248 different days. This was impossible to properly graph and see a popular trend resulting in missed opportunities to conclude invaluable reasonings.

**Data filtration complications:** Another problem we encountered was the data we collected had been heavily shaved off. The total number of viewers in the stremer_viewer MapReduce job for Riotgames equates to over 21 million yet in the streamer_timestamp, through filtering out data which did not include a timestamp we lost nine tenths of the data needed. This has had heavy consequence on the final results yielded. When graphing the Riotgames viewer results, one member of the group noticed the results was too low given the immense popularity of the channel and game.

## Conclusion

In conclusion, The first of the two main aspects we set out to find are “How will you detect the top streamers/games, vs the non- popular ones (in MapReduce terms)”. We successfully implemented a MapReduce job to group the games with their views in a table and show the highest ranking games, the top of which were League of Legends and DOTA2. This job providing a main basis for creating the skeleton of the rest of the MapReduce jobs.

The next question given was “What factors you think you can extract from the data/additional sources that might have an influence in popularity? Keep in mind that over the 10 month dataset, we collect all the active channels every 30 mins.” We think factors such as popular trends can cause rises in popularity. League of Legends is currently in a boom period of players, similar to how World of Warcraft was at its peak a decade ago. the games popularity can be exponentially increased if is a “fad” amongst gamers. Another factor that could affect popularity is whether there is a tournament being held. As we saw in the Riotgames viewers MapReduce results analysis, The viewer numbers increased during a time where Riotgames was participating in a League of Legends tournament. This is seen as a significant factor in determining the popularity of not just a game, but more importantly the streamer.

## References


Twitch.tv dataset
Found on the QMUL hadoop server
hadoop fileserver location - /data/twitch/twitch-all-channels.csv

WordCount.jar and related source code modified for use in project
Found on the QMUL big data processing module page
http://qmplus.qmul.ac.uk/course/view.php?id=4916