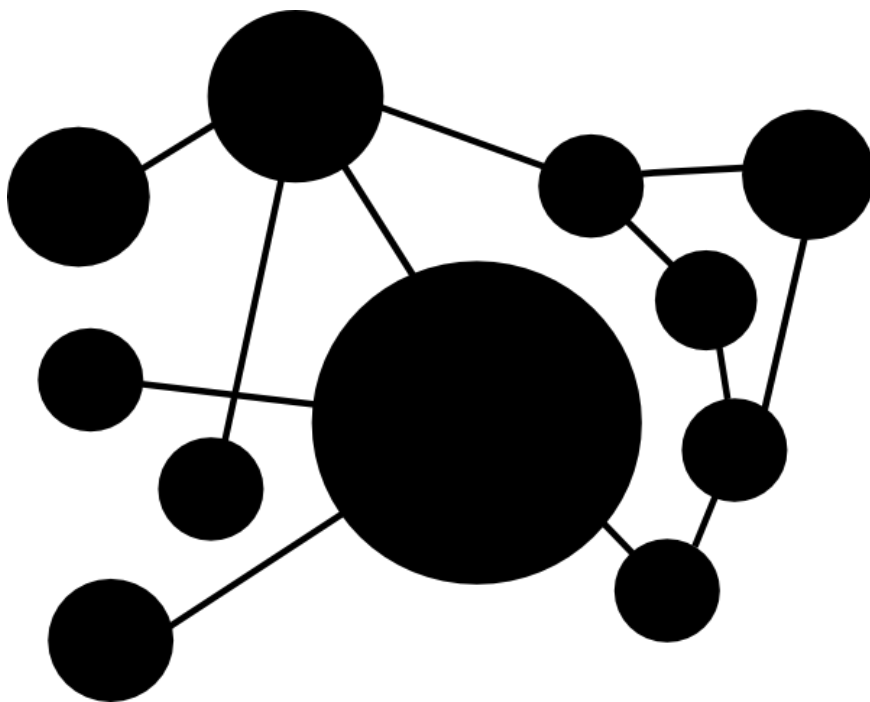




Modelling a Community: Network Software and parkrun Data



parkrun

Student Name (Student Number)	Eoin Carroll (16202781)
Module Code	UCD MIS40550
Module Title	Network Software Modelling
Assessment Title	Network Software Modelling Assignment 2
Module Co-ordinator	Dr James McDermott
Revision	0
Date Submitted	23/04/2017

Plagiarism

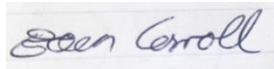
The unacknowledged inclusion of another person's writings or ideas or works, in any formally presented work (including essays, examinations, projects, laboratory reports or presentations). The penalties associated with plagiarism designed to impose sanctions that reflect the seriousness of University's commitment to academic integrity. Ensure that you have read the University's Briefing for Students on Academic Integrity and Plagiarism and the UCD Plagiarism Statement, Plagiarism Policy and Procedures, (<http://www.ucd.ie/registrar/>).

Declaration of Authorship

I declare that all material in this assessment is my own work except where there is clear acknowledgement and appropriate reference to the work of others.

Signatures

Signed:



Date: 23/04/2017

Abstract

A weekly event called parkrun has led to the growth of a passionate community of runners. This community has been modelled by predicting the relationships of runners using event result lists. A simulation of information flow was performed and the result was that if the most connected runner shared a message, they could reach 30% of the community within a short space of time.

Contents

Introduction	4
Graph Generation.....	4
Web Scraping	4
Constructing Graphs	5
Erdős–Rényi Random Graphs.....	5
Graph Properties	5
Graph Degree Plots.....	6
Information Flow	7
Discussion	8
Conclusion.....	8
References	9
Appendix.....	10
Further Research	10
Python File	10
Data location	10
Github Repository.....	10

Introduction

Parkrun is a recent social phenomenon that has spread across 14 countries since 2004. It is a weekly timed 5km fun run on Saturday mornings, is free to attend and managed entirely by volunteers. Parkrun came to Ireland in 2012 and events now take place concurrently in 61 locations across the country (parkrun.ie, 2017). The result of this phenomenon is a rapidly growing community of runners.

In this project we will model the social network from one of these locations and examine the theoretical flow of information in the form of a message spreading through the community. Fortunately, parkrun places event results online for us to explore and utilise. The scenario proposed is that a decision is made to cancel an event. This message is passed from person to person based on a probability that they know each other well enough to have each other's phone number.

The event location of Marlay park in Dublin has been selected as it is the most popular event in Ireland and will provide enough data for the study. We will examine the information flow in the following six cases:

1. Social network of the first event in Marlay park
2. Social network of the first 10 events (sample network)
3. Social network of all events to date (full network, 204 events)
4. Erdős–Rényi model with 100 nodes
5. Erdős–Rényi model with 1000 nodes
6. Erdős–Rényi model with 5000 nodes



Figure 1 - parkrun Events

Graph Generation

One of the reasons parkrun was selected for the project is the availability of data. The online results for each event include each person's finishing position, time, name and associate running club if available. Furthermore, parkrun has been experimenting with an API to allow direct access to the vast amount of data available. All data files are stored in a cloud hosting service linked in the appendix.

Web Scraping

Unfortunately the API system was offline for the duration of the project and so alternative means of acquiring the dataset were explored. The next option explored was to sign up as a research partner which parkrun encourages however this was not practical due to time constraints. Next, the Python library BeautifulSoup was implemented to web scrape each results page for Marlay park. This was an interesting learning but the parkrun website politely blocked web scraping. The final option was to manually copy data from each result into locally stored files and while this was not ideal, it sufficed for the project. Each weekly result file is saved in the data folder with the name (race week).xlsx.

Table 1 - Sample Header from dataset (simplified)

Pos	parkrunner	Time	Club
1	Name 1	16:26	Raheny Shamrock AC
2	Name 2	17:22	
3	Name 3	17:31	Dundrum South Dublin Athletic Club
4	Name 4	17:36	Dundrum South Dublin Athletic Club
5	Name 5	17:52	

Constructing Graphs

An undirected, weighted graph was selected to represent the social network. Each node represents a unique name in the dataset, each edge is a connection between two people and self-loops were not applicable. The weight of each edge will represent the probability that these two people know each other well enough to have shared phone numbers.

Once the raw datasets had been saved from the results website, they could be processed into networks. An algorithm was constructed to read each name from the raw data and create nodes and weighted edges. This was completed individually for each race result set. The following attributes were used to assign probabilities of connections:

1. Two people attended the same event (everyone in each dataset)
2. Two people were in the same running club
3. Two people have the same second name
4. Two people finished with a similar finishing time
5. Two people finished with a similar position

Within the similar time and position parameters, same name and club were given more weighting to represent an increased probability. These parameters were tested with different inputs based mainly on domain knowledge. The resulting graphs were examined critically and the algorithm adjusted until a reasonable output was settled on. Note that this method allowed a small but present probability of all participants knowing each other in a given week. Parallel processing in python was implemented to speed up the network generation and the resulting network for each of the datasets was saved as a “edgelist” .csv file. Again, these data files can be seen at the attached link.

The next stage in the project was to merge the weekly networks as generate above, into a sample of the first 10 weeks and into a full dataset. All duplicate edge weights were summed together and included in the final networks. Generation and combination of networks from the datasets was completed using the functions `generate_weekly_graphs`, `combine_networks` and smaller functions which were called within these. The python code file is included with this project. The graph properties section of this report includes an insight into the networks structure. Note that before processing any edges, the edge weights are scaled to between 0 and 1 to give a probability.

Erdős–Rényi Random Graphs

Erdős–Rényi random graph models were included in the project to contrast the information flow with the real-world graphs generated. A function was included to create three Erdős–Rényi which would somewhat mimic the structure of the parkrun networks. Three graphs were created with 100, 1000 and 5000 nodes and their properties are covered in the next section.

Graph Properties

In the real world parkrun networks, there are regular participants who would have created links between each weekly result and so we will most likely see these people emerge as nodes of high centrality. Furthermore, each weekly race or each running club could be seen as a cluster within the full network. A function was included to save the properties of each network to a text file and the results are summarised in the following table.

Table 2 - Graph Properties

	First Week	First 10 Wks	Full Network	Erdős-Rényi 100	Erdős-Rényi 1000	Erdős-Rényi 5000
Order	331	1464	14070	64	1000	5000
Size	54615	356140	8514729	53	50174	6246130
Density	1.0	0.33	0.09	0.03	0.10	0.50
Clustering coefficient	1.0	0.82	0.77	0.0	0.10	0.50
Diameter	1	2	2	Unknown	3	2
Size of largest component	331	1464	14070	15	1000	5000

Graph Degree Plots

A function to create degree distribution plots was added to the program and used to create the graphs being considered. There are two plots for each network, the first shows all connections and the second adjusted plot shows only connections with greater than 50% probability of information flow between the nodes. All graphs are stored in the results folder linked and the insightful plots are shown below.

Starting with the first week parkrun dataset, all components were connected to each other and so the histogram was uninteresting. The adjusted plot showed that a small number of people had strong connections with up to four other people.

The sample parkrun dataset gave more of an insight into how the social network was developing over the first 10 events. We can see in the unadjusted graph, the majority of people were in some way connected to roughly 300 people which is the average number of people attending each event. Looking at the adjusted graph, we can see that most people are strongly connected to very few people however there are some who have up to 40 strong connections. These nodes will have high centrality measures.

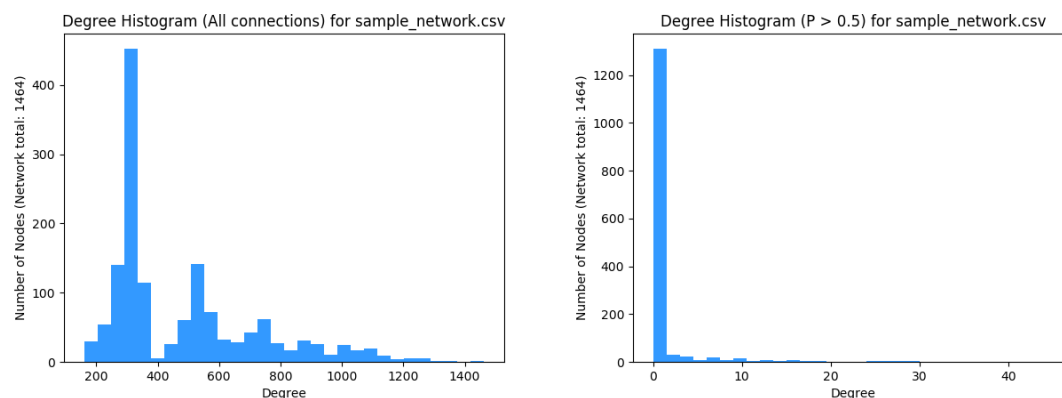


Figure 2 – Sample parkrun Network Degree Distribution

Moving on to the full parkrun network, again the majority of people are loosely connected with roughly the same number of average event attendees. This indicates that many people have only attended one event. In the full network, we can also see a nice sloping reduction in the number of people with high connectivity. Looking at the adjusted graph, we see again that few people have very strong connections but there are some nodes with high centrality.

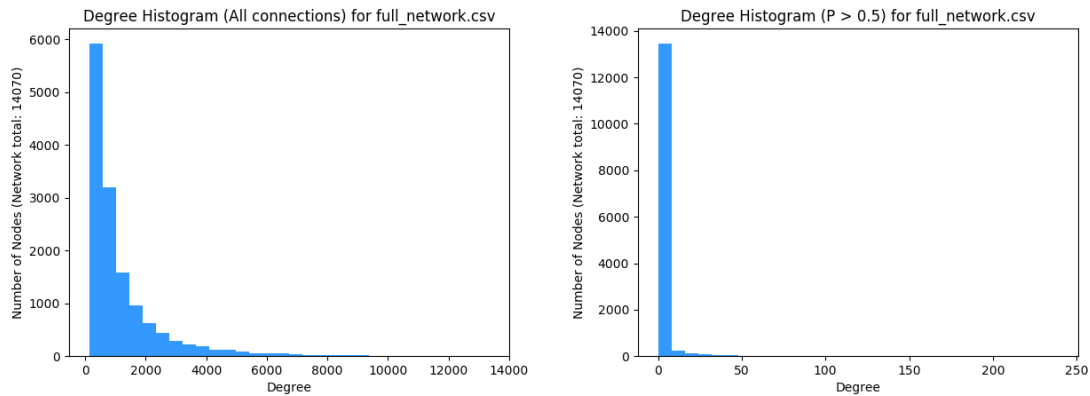


Figure 3 - Full parkrun Network Degree Distribution

Similar to the single parkrun event, the Erdős–Rényi graph with 100 nodes was too small of a sample size to obtain meaningful information from the plots. The 1000 and 5000 node Erdős–Rényi models however showed an almost normal distribution for all connections. The adjusted graphs resulted in most nodes having no connections and only a small number with 1 degree of connection for the 1000 node network. The 5000 node network was similar again but had some 2 degree connections.

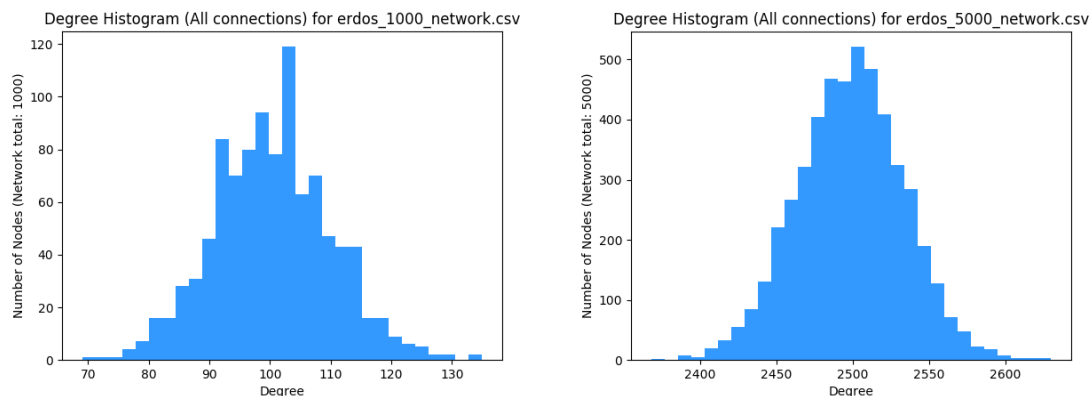


Figure 4 - Erdős–Rényi 1000 & 5000 Degree Distribution

Information Flow

The scenario being considered is that a decision to cancel an event is made by the most connected person in each network (regardless of edge weight). This decision happens at 9am with the race start time originally scheduled at 9.30am. The simulation function of the program iterates through time steps and sets the state of each node to indicate if they have received the cancellation message. We will let a time step equate to 5 minutes for the purpose of the project. The probability of the message passing through each edge is based on the strength of the relationship as modelled from the datasets and encoded in the edge weights. It is likely that the edge weights will dictate the rate of information flow along with the density and clustering coefficient to a lesser extent.

Each node can be in state 0 which means that they have not received a message, state 1 meaning they received a message and should pass it on to their connections or state 2 where they have gone back to bed on a Saturday morning. The result of this simulation was examined by plotting the number in each network per state over time. The message failed to propagate significantly through each of the

networks from first race, sample network, Erdős–Rényi 100 and 1000. The results are included in the results folder for reference.

The full parkrun network and Erdős–Rényi 5000 models were far more interesting to examine. The message caught traction and was passed to a significant portion of the community. In both cases we can see that a steady state was reached after relatively few iterations.

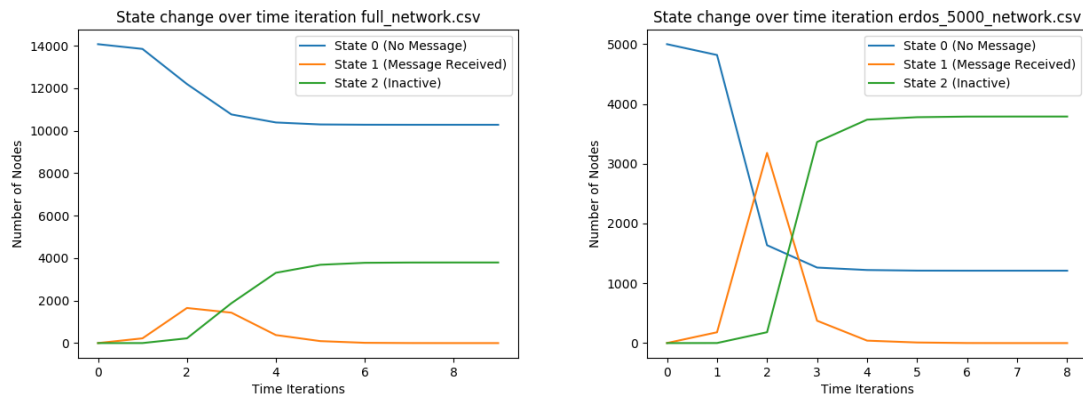


Figure 5 - Full and Erdős–Rényi 5000 Simulation Results

In addition to this real world case, a message was seeded at random locations in each graph. This resulted in varying degrees of propagation however in both graphs, the message spread far less than shown above.

Discussion

The simulation results for the smaller networks were uninspiring. There was not enough interconnectivity for the information to flow between nodes. The two large networks however provided very useful information. If the assumptions for creation of the models was correct, then a message sent by the most connected person would propagate through the network and reach a steady state within 35 minutes. By this time, 4000 out of 14000 ($\approx 30\%$) people in the parkrun network would have received the message. Likewise, 4000 out of 5000 ($\approx 80\%$) of people in the Erdős–Rényi graph would have received the message and know they can go back to sleep on Saturday morning. It is likely that the significantly increased density of the Erdős–Rényi graph allowed more information flow in comparison to the full parkrun network.

In hindsight, too much emphasis was placed on generating the networks from real world datasets rather than on examining how different graph types and parameters would affect the simulation. This was a result of ambitiously setting out with a scope that did not match the project brief. An alternative path could have been to mention that real data is available but then use random graphs for simulation while focussing on the sensitivity of the model to network parameters such as density.

Conclusion

A real world case study has been completed on a parkrun dataset. Information flow was modelled to simulate an event cancellation message spreading throughout the community starting at 9am. Our conclusion is that roughly 30% of people in the network will have received the cancellation message before the original race start time of 9.30am. Network density will increase information flow.

References

1. Network clipart, , accessed 2017.
<http://www.clker.com/cliparts/G/G/F/Y/Y/U/network-md.png>
2. Parkrun, accessed 2017.
<http://www.parkrun.ie>
3. Parkrun API, accessed 2017.
<http://www.parkrun.com/api/>

Appendix

Further Research

Some possibilities for further work were recorded. Firstly, a more thorough study could merge the networks from other locations to form a complete network. Due to the quantity of data this would need to be done through the API facility if it comes back online. Further adjustments to the graph generation parameters could be made to more accurately reflect the real world community. Perhaps a survey or similar could be used in this area. One limitation on the project was the extreme run times. This could be reduced by changing data types and improvements to the code. Lastly, the deterministic nature of the randomness was not explored fully, by running the simulation a number of times and recording the average results, we may obtain more useful insights.

Python File

The python program has been included and can be run to verify any findings. The estimated run time for each function is listed and so it is advised to check this before running. At a minimum, the python file needs to be placed in a folder along with a data folder containing raw data and an empty results folder. Alternatively, all data can be downloaded from the cloud and stored to run only specific functions.

Data location

This publicly shared folder will be available until grades for the assignment have been released. This folder includes the following files:

- Data folder
 - Raw data files for each set of weekly results
 - Network data file generated from each raw data file
 - Sample network and full network files
 - 3x Erdős–Rényi random graphs
 - 6x simulation results files
- Results folder
 - Network generation and combination log files
 - 6x graph properties files
 - 6x full network distribution histograms
 - 6x adjusted network distribution histograms
 - 6x simulation result plots
- Python Code
- Report and assignment brief

<https://drive.google.com/drive/folders/0B9kelMwrrRsROF9UZFY5SZZUzg>

Github Repository

The project was made available on Github with the following URL. Note that the data files are too big to store on Github.

https://github.com/eoincUCD/Modelling_Parkrun_Community