Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Lecture with Computer Exercises:
Modelling and Simulating Social Systems with MATLAB

Project Report

The Impact of Party Finance and Voter Targeting

Ioannis Fousekis,
Terry Louise Jones,
Moritz Hohenfellner

Zurich
December 2016

Agreement for free-download

We hereby agree to make our source code for this project freely available for download from the web pages of the SOMS chair. Furthermore, we assure that all source code is written by ourselves and is not violating any copyright restrictions.

Ioannis Fousekis Terry Louise Jones Moritz Hohenfellner

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Declaration of originality

The signed declaration of originality is a component of every semester paper, Bachelor's thesis, Master's thesis and any other degree paper undertaken during the course of studies, including the respective electronic versions.

Lecturers may also require a declaration of originality for other written papers compiled for their courses.

I hereby confirm that I am the sole author of the written work here enclosed and that I have compiled it in my own words. Parts excepted are corrections of form and content by the supervisor.

Title of work (in block letters):

Modeling & Simulating Social Systems, Semester Project - Party Financing

Authored by (in block letters):
For papers written by groups the names of all authors are required.

Name(s):	**First name(s):**
Fousekis	Ioannis
Jones	Terry Louise
Hohenfellner	Moritz

With my signature I confirm that
- I have committed none of the forms of plagiarism described in the 'Citation etiquette' information sheet.
- I have documented all methods, data and processes truthfully.
- I have not manipulated any data.
- I have mentioned all persons who were significant facilitators of the work.

I am aware that the work may be screened electronically for plagiarism.

Place, date	**Signature(s)**
Zurich, 18.12.2016	[illegible]
	T. Jones
	[illegible]

For papers written by groups the names of all authors are required. Their signatures collectively guarantee the entire content of the written paper.

Contents

1 Abstract **6**

2 Individual contributions **6**

3 Introduction and Motivations **7**
3.1 Fundamental questions 7

4 Description of the Model **8**
4.1 Strategies 9
4.1.1 Random Selection 9
4.1.2 Target the Convincibles 9
4.1.3 Protect Your Own 10

5 Implementation **10**
5.1 Network 10
5.2 Experiments 10
5.2.1 Experiment 1: Model Application to our Network 10
5.2.2 Experiment 2: Strategy Implementation 11
5.2.3 Experiment 3: Opinion Formation in Focus Areas 11

6 Simulation Results and Discussion **12**
6.1 Experiment 1: Model Verification 12
6.2 Experiment 2: Strategy Implementation 13
6.3 Experiment 3: Opinion Formation in the Focus Areas 14
6.4 Comparison of the Different Strategies 15
6.5 Conclusion 16

7 Summary and Outlook **17**

8 Bibliography **18**

A Java Code - Model **19**
A.1 Package Controller 19
A.1.1 Class AnalyseNetwork 19
A.1.2 Class MultipleMain 20
A.1.3 Class SimpleRun 24
A.2 Package Model 27
A.2.1 Class InitializeAgentsOpinions 27
A.2.2 Class MassMediaCarlos 28

A.2.3 Class Strategy 31
A.3 Package Model.Strategies 32
A.3.1 Class TakeRandom 32
A.3.2 Class TakeConvincible 33
A.3.3 Class TakeProtect 35
A.4 Package Network 37
A.4.1 Class Agent 37
A.4.2 Class ClusterFinder 39
A.4.3 Class MEdge 41
A.4.4 Class Network 42
A.4.5 Class NetworkEvaluator 47
A.4.6 Class WattsStrogatz 48
A.5 Package Util 51
A.5.1 Class CSVReader 51
A.5.2 Class CSVWriter 53
A.5.3 Class GlobalRandom 56
A.5.4 Class JasonReader 57
A.5.5 Class ParamGridReader 59
A.5.6 Class ParamSet 60
A.5.7 Class Record 61
A.5.8 Class Viewer 63

B R Code - Analysis **65**
B.1 Evaluate Set 65
B.2 Generate Standard Plots 66
B.3 Compare Strategies 69
B.4 Generate Parameter Grid 72
B.5 Evaluate Temporal Dynamics 74
B.6 Evaluate Input Signals (for diagnosis) 75
B.7 Plot Design Configuration 76

1 Abstract

We set out to study what effect a parties finances can have on voting results. To do so we look at binary choices agents have the make. While they are initially assigned an opinion at random, there is the possibility for the agents to change to the opposing opinion. For this to happen two conditions must be fulfilled: first they must receive a media input of the opposing opinion and second the share of neighbours holding that opposing opinion must be above a given threshold. A high threshold is seen as an indication that the topic on which a decision is made is an emotional one, with a voters personal values and those of his social environment being of importance. With claims of individual targeting having been used in recent elections and votes, we further investigate how strategic targeting of voters can impact the voting outcome. It was found that in the model financing as well as individual targeting can have major influence on the outcome on voting results, if the initial share of proponents and opponents is close to being equal. Furthermore, the impact of differences in funding can be almost cancelled out by using targeting strategies.

2 Individual contributions

Ioannis Fousekis	Terry Louise Jones	Moritz Hohenfellner
Literature	Head of Report	Head of Implementation
Report Contribution	Motivations	Data Analysis
	Discussion of Results	Motivations
	Strategy Development	Discussion of Results
		Strategy Development
		Report Contribution

3 Introduction and Motivations

Most countries of the Western world today have a democratic political system. The people get to elect their representatives and thereby have a say in how their country is shaped. In Switzerland, with its semi-direct democracy, this even goes a step further. Swiss citizens do not only get to elect their representatives, but are directly involved in important decision making processes by voting on initiatives and referenda several times a year.
A question that has been raised many times before is to what degree campaign financing affects the outcome of an election. Having more money will enable a party to invest more in media coverage and advertisement, hence being able to capture a larger part of the voters attention. In Switzerland, however, parties do not have to disclose any information on campaign finance. A fact has been criticized by left wing parties as well as by the group of states against corruption (GRECO). Moreover, in April 2016, a collection of signatures for a popular initiative to change transparency laws has been launched.
In the past months claims of targeting of individuals leading to unexpected outcomes in elections and votes have been voiced. Allegedly, people were receiving advertisement based on big data information, and psychological profiling, therefore making it possible to use money more effectively. If this is indeed true, the difference in funding could potentially have a large impact if used by the party with higher funds, or even out the playing field if only used by the party with less financial means.

3.1 Fundamental questions

The main goal of this study is to see if and how the amount of money a party has at its disposal influences their chances on swaying the popular vote in their favour. Hereby only binary decisions are considered.
A further aspect taken into consideration is that not only the media, but also personal preference and the opinions of people in an individuals environment can influence the decision-making process. A higher importance of the environment for highly emotional topics, where decision-making is based more on morals and values the individual holds, is assumed. In contrast, topics, to which voters are likely to have a smaller emotional or personal connection, which are more abstract and rational, media and information received from experts is considered to be higher weighted.
A further aim is to investigate the effect of individual targeting on voting results using different strategies. It is difficult to determine the impact of targeting in the real world and while we do not claim any predictive power, we do hope to gain some insight into whether targeting could potentially have some effect on voting outcomes. The model described by Hegselmann et al. (2002) will be used. Hereby each agent

can make binary decisions on a topic, either in favour or oposing to the quetion asked, and receives external input, which represents input recieved through the media. The effect of financing will be investigated by using this media input as a proxy. The final opinion of an agent will depend on three factors, their initial opinion, the opinion presented to them by the media input and the opinion of its neighbours. This will help us to answer the following research questions:

- Does the difference in budget influence the decision on political topics?
- What is the effect of individual targeting?

The focus of this study will be on topics with a higher emotional component, for which it is assumed that the voters eighbours have a higher impact. Therefore, a high threshold for τ is used in the simulation, factoring in the higher weight of the number of neighbours whose opinion influences the individuals opinion.

4 Description of the Model

The number of nodes n represents the total number of agents or individuals i in the network. We denote s the time steps that the updated dynamics take place in, as discrete time steps, i.e. s= 0,1,2... We denote $o_i(s)$ the opinion of the individual i in the time step s. There are the two alternative opinions that the agents can have, i.e.

$$o_i(s) = \begin{cases} 1 & \text{if } i \text{ is a proponent} \\ 0 & \text{if } i \text{ is a opponent} \end{cases} \tag{1}$$

$$0 \leq i < n \tag{2}$$

In the initial conditions $o_i(s = 0)$, every agent in the network is given one of the two alternative opinions. That represents their starting opinion. Opinions are randomly distributed for each individual with a probability p_0 where

$$o_i(0) = \begin{cases} 1 & \text{with probability } p_0 \\ 0 & \text{with probability } (1 - p_0) \end{cases} \tag{3}$$

The signal agent i receives via advertisement in time step s is given by $I_i(s)$, where

$$I_i(s) = \begin{cases} 1 & \text{with probability } p_m \\ 0 & \text{with probability } (1 - p_m) \end{cases} \tag{4}$$

At each time step every agent in the network is confronted with one of the two alternative opinions. If agent i's previous opinion $o_i(s-1)$ coincides with the signal $I_i(s)$ he receives, then the agent keeps his opinion. If agent i's previous opinion $o_i(s-1)$ does not correspond to the $I_i(s)$, then the agent considers whether to change his current opinion based on the opinions of his neighbours. The agent changes his opinion if the fraction $\tau_{i,I}$ of his neighbours k who have $o_k(s-1) = I_i(s)$ at time step s exceeds a certain threshold. We denote τ the threshold for action change. The number of neighbours of an individual's network is denoted as N.

Mathematically:

$$\tau_{i,I} = \frac{\sum_{\forall k} (o_i(s-1) \text{ XNOR } I_i(s))}{N} \tag{5}$$

$$\tau_{i,0} = 1 - \tau_{i,1} \tag{6}$$

$$o_i(s) = \begin{cases} o_i(s-1) & \text{if } I_i(s) = o_i(s-1) \\ I_i(s) & \text{if } \tau_{i,I} \geq \tau \\ o_i(s-1) & \text{if } \tau_{i,I} < \tau \end{cases} \tag{7}$$

4.1 Strategies

Strategies change the way the media signal is assigned to the agents. While the two parties can have an influence on which agents receive a signal for their opinion, the share of agents p_m which receive a proponent signal in a time step is given. This share serves as a proxy for the differences of financial means of the two parties. Agents which have received a signal cannot receive a second signal in the same time step. Therefore it matters in which order the parties choose to treat the agents. The parties take turns at treating agents until eventually one party has reached its limit, which for the proponents would be $p_m \cdot n$. After one party has reached its limit the other party treats all the so far untreated agents.

4.1.1 Random Selection

This strategy is the same as the basic model. An agent is randomly selected from the list of untreated agents to receive media input.

4.1.2 Target the Convincibles

The first agent from the list of untreated agents, who can be convinced to vote in the favor of the party, is selected. Hereby only agents that have not yet adopted the opinion of the party are considered.

4.1.3 Protect Your Own

Each party focuses first on the agents that would be lost to the other party if they were to target them first. Once a party has secured all its voters, it then focuses on the voters that would be voting for the opposing party, but whom they are likely to convince otherwise.

5 Implementation

The model is implemented in Java using an object oriented approach. For graph applications the JUNG library is used. The code is given in the appendix A. [1] Results were evaluated using R.

5.1 Network

The network used is a co-authorship network, inferred by Thomson Reuters from "Web of Science" data (Web of Science, 2015). It was cleaned by identifying the biggest component and using only this component for the analysis. The number of nodes which were therefore not used is 289. The biggest component contains 2310 nodes. The average degree in the biggest component is 3.6 and the diameter is 18.

5.2 Experiments

Three series of experiments are conducted to determine impact of financing and individual targeting strategies on voting outcomes.

5.2.1 Experiment 1: Model Application to our Network

The first experiment was designed in order to apply the model presented by Hegselmann et al. (2002) to our network. The probability of the share of the agents' opinions p_0 and the probability of the distribution of input p_m are set to be equal, i.e. $p = p_0 = p_m$, with $p \in [0, 1]$ and step size of 0.05. The threshold for opinion change takes values from, $\tau \in [0, 1]$ with a step of 0.05. For the experiment the number of time steps has been set to $s = 300$ with 20 runs being conducted per cell in the created grid.

[1] The variable `initPosShare` refers to p_0 and `posMassMediaShare` refers to p_m.

5.2.2 Experiment 2: Strategy Implementation

The focus of the second set of experiments was the identification of the impact of the input opinion, i.e. the financing, on the voting outcomes. The different strategies described above were tested against each other. The focus hereby lies on the conditions, in which a positive outcome for the proponent in a vote is more difficult to achieve, with the share of the agents' initial opinions being limited to the range $p_0 \in [0, 0.5]$. A step size of 0.05 was used. The probability of the distribution of inputs takes values from $p_m \in [0, 1]$ with a step size of 0.5. Additionally, the number of time steps has been set to $s = 300$ and 10 runs have been conducted per cell in the created grid. The threshold for opinion was fixed as $\tau = 0.7$.

5.2.3 Experiment 3: Opinion Formation in Focus Areas

From the second series of experiment we have identified the areas of interest, in which changing the outcome is possible. Therefore this area was studied in more detail. The experiment has been limited the range of p_0 to $p_0 \in [0.4, 0.5]$ step size has been reduced to 0.02. The range of shares of inputs has not been altered compared to the previous experiments, $p_m \in [0, 1]$ with a step size of 0.5. The number of time steps is $s = 150$, which turned out to be sufficient, however 100 runs have been conducted per cell in the created grid. The threshold for opinion was fixed as $\tau = 0.7$.

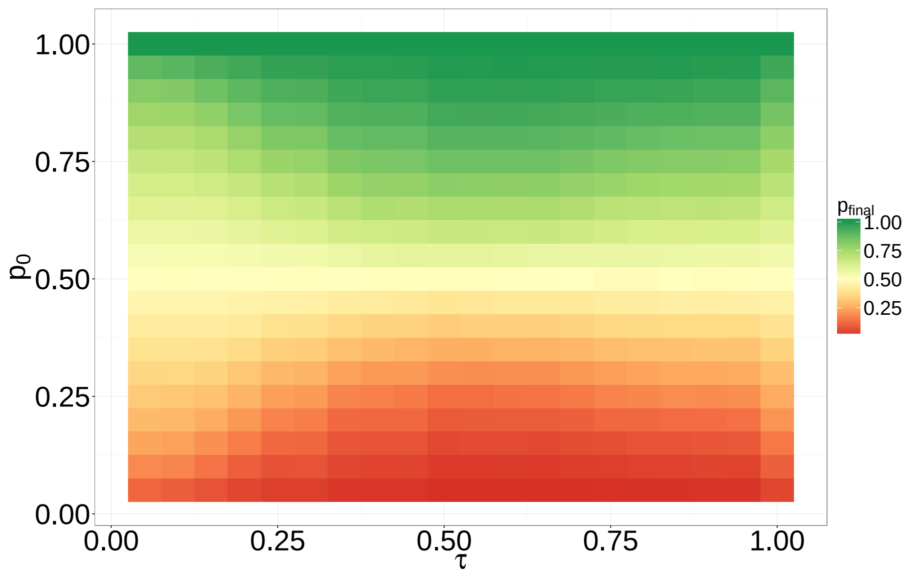

Figure 1: Mean outcome after 300 time steps. 20 runs per grid point. $p = p_0 = p_M$. Both parties playing no strategy.

6 Simulation Results and Discussion

6.1 Experiment 1: Model Verification

The model described by Hegselmann et al. (2002) was tested on our network using a parameter grid. For each of the parameters τ and $p_0 = p_m$ 20 values between 0 and 1 were tested. For each grid point 20 runs with 300 time steps each were performed. The number of time steps turned out to be sufficient since there was almost no variation of the observed opinion shares within the last time steps. While outcomes observed follows the same pattern as they do in aforementioned paper, see Figure1,a less clear distinction between the phases was achieved. This could be remidied by an increase in the number of nodes in the network.

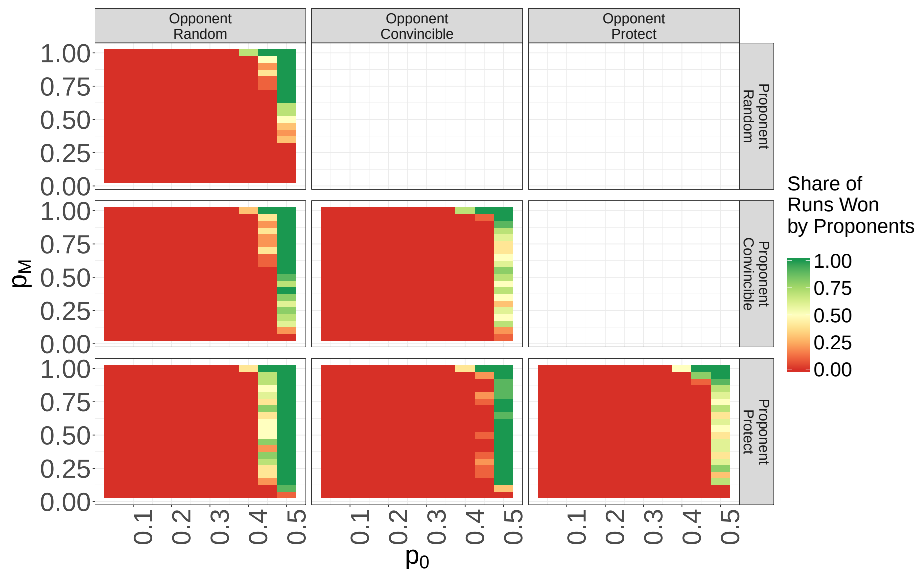

Figure 2: Comparison of different strategies against each other. With 10 runs per grid point, 300 time steps and $\tau = 0.7$.

6.2 Experiment 2: Strategy Implementation

The effect of the proponent using different strategies, while the opponent continues to use random targeting, as well as individual targeting strategies to win voters was investigated, see figure 2. Herby input shares in the range of $p_m \in [0, 1]$ and initial opinion shares in favour of proponents of $p_0 \in [0, 0.5]$ were considered. The threshold for τ was 0.7.

In a first instance both parties use random targeting. The proponents achieve a small number of wins in the area investigated, however only when the share of input is $p_m \in [0.5, 1]$ and and the initial share of opinions is $p_0 = 0.5$. This serves as a benchmark for further investigation.

Using the "target the convincibles" strategy results in a somewhat better outcome. Additionally votes can be won for lower shares of media input $p_m \in [0.2, 1]$ for $p_0 \in [0.45, 0.5]$.

With proponents using the "protect your own" strategy a high number of successful opinion changes took place for $p_0 \in [0.45, 0.5]$ for almost the entire range of input shares $p_m \in [0.1, 1]$. For $p_0 \in [0.45, 0.5]$ the proponents won (almost) every run.

With both proponent and opponent using the "target the convincibles" strategy, the results were much noisier, compared to the strategy being used by only proponent,

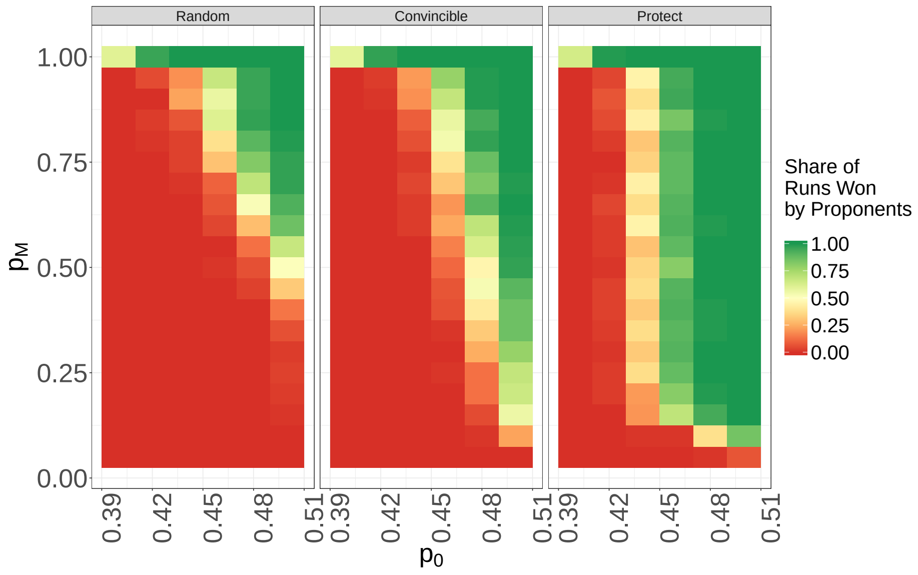

Figure 3: Comparison of different strategies in the focus area. Opponents play the random strategy. With 100 runs per grid point, 150 time steps and $\tau = 0.7$.

with opponents using random targeting. The same holds true for the case in which both parties use the "protecting your own" strategy.
In the case that proponents use the "protect your own" strategy and opponents target the convincible agents first, the initial opinion share p_0 has to be higher compared to when opponents use random targeting.

6.3 Experiment 3: Opinion Formation in the Focus Areas

As discussed in the previous section, only in the upper range of initial opinions $p_0 \in [0.39, 0.51]$ successful opinion changes were observed. Therefore this range was looked at in more detail. For the share of inputs the full range $p_m \in [0, 1]$ was studied. The results for the successful outcomes can be seen in figure 3. Further focus was on which votes the proponents win when using the different strategies.
With both opponent and proponent using random targeting, votes were only won when the share of input was in the range of $p_m \in [0.5, 1]$. While at $p_m = 0.5$ proponents can win, when $p_0 \leq 0.49$, higher shares in input are needed for lower initial opinion shares, such as $p_m \in [0.6, 1]$ for $p_0 \leq 0.47$ and $p_m \in [0.8, 1]$ for $p_0 \leq 0.45$.
The proponents using the "target the convincibles" strategy, when the share of initial

Table 1: Mean result for proponents. Opponents target randomly. 100 runs per cell, 150 time steps.

		Mean			Standard Deviation		
p_0	p_m	Random	Protect	Convincible	Random	Protect	Convincible
	0.25	0.37	0.53	0.44	0.02	0.02	0.03
0.46	0.50	0.43	0.53	0.47	0.02	0.03	0.02
	0.75	0.48	0.53	0.50	0.03	0.02	0.03
	0.25	0.40	0.56	0.47	0.02	0.02	0.03
0.48	0.50	0.46	0.56	0.50	0.03	0.02	0.02
	0.75	0.52	0.56	0.53	0.03	0.02	0.03
	0.25	0.44	0.60	0.51	0.03	0.03	0.03
0.50	0.50	0.50	0.60	0.54	0.03	0.02	0.03
	0.75	0.55	0.60	0.57	0.03	0.02	0.02

opinions is $p_0 = 0.45$, win the vote only when their share of input is above 75%, $p_m \in [0.75, 1]$. When increasing the share of initial opinions to $p_0 = 0.47$, a victory was achieved with $p_m \geq 0.5$. For $p_0 \in [0.49, 0.51]$, the proponents already win the vote at $p_m \geq 0.1$.
When using the "protect your own" strategy, initial shares of opinion in the range of $p_0 \in [0.45, 0.5]$ result in wins for the proponent for $p_m \in [0.1, 1]$.

6.4 Comparison of the Different Strategies

The impact of financing, represented by p_m, and using different strategies to target the agents were further investigated in the range $p_0 \in [0.39, 0.5]$.
In order to determine the impact of different strategies played, mean results for the proponent playing different strategies (random, protect your own, target the convincibles), while the opponent used random targeting, were compared, see table 1. Outcomes for three different values of p_0 as well as three different values of p_m were looked at.
The "protect your own" strategy yields the highest mean results out of the three strategies. There appears to be no effect of financing when this strategy was played, with same mean values being reached whether for both high and low shares of agents receiving the input. The random targeting has the worst results among the three strategies.
Further the actual outcome of the votes were compared, whereby the opponents used again random targeting, while the proponents used the three different strategies. For each cell on the grid the strategy which achieved the highest numbers of victories is given, see figure 4. As indicated by the results seen in table 1, the "protect your own"

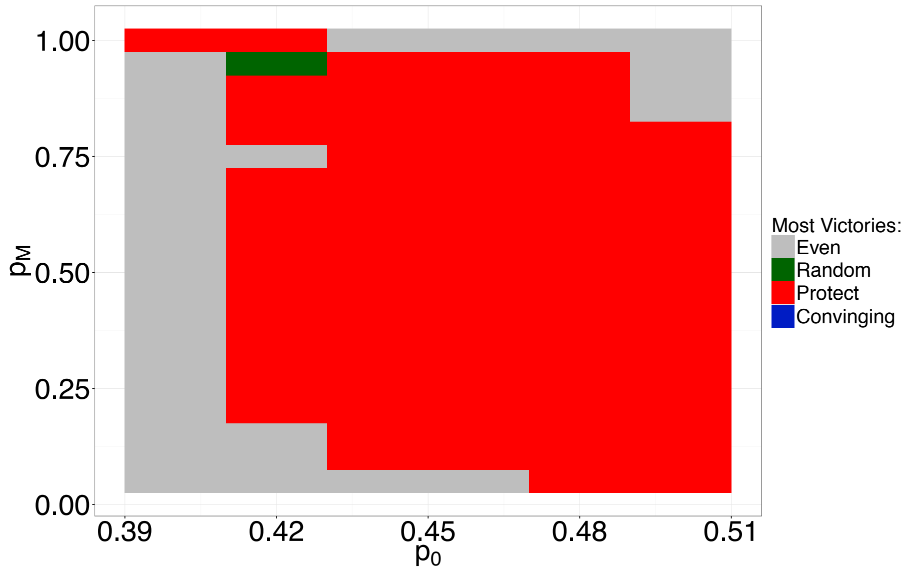

Figure 4: For each cell on the grid the strategy by the proponent yielding the highest number of victories is displayed. The opponent targets randomly. 100 runs per cell for each strategy, 150 time steps.

strategy is clearly the best out of the three options, capturing the most victories in most of the cells investigated.

6.5 Conclusion

These findings indicate, that having higher financial means than the opposing party can sway voting outcomes in favour of that party. This holds true for cases in which the initial share of opinions was 40% or higher. Hereby must also be taken into consideration that the threshold for τ was quite high (0.7), meaning whether an agent can be convinced to change his opinion depends to a high degree on his neighbours.
There is a visible difference between the outcomes for the different strategies for voter targeting used. This indicates that using specific strategies might indeed impact voting outcomes.

7 Summary and Outlook

This study aimed at answering the question whether financing affects voting outcome. The share of agents receiving an input was used as a proxy for the financial means either party has. The agents could receive the inputs either by random distribution or by individual targeting. This was the focus of the second question posed; whether targeting has an effect on voting outcome. In this study we focused on topics, which require a high share of an agents neighbours to have the opinion which the agent would switch to, in order for him to do so.
When random targeting was used, a higher than 50% share of media inputs could lead to agents being swayed to vote against their initial opinion and thus changing the outcome of the vote. A clear difference can be seen between the different strategies, with the "protect your own" strategy, in which a party focuses on securing its own voters before targeting voters, which are currently supporting the oposing party ,that they can convince to vote for them, being the clear winner. For this strategy the financial aspect, however, showed no impact when the opponent used random targeting, with votes being won even for low media shares. For the other two strategies, having higher financial means increased the positive outcome. Herefore, the share of voters holding the parties opinion had to be higher than 39% at the beginning of the run.
Overall this shows that both the financial means as well as using different strategies to target voters can swing a vote in a partys' favour. This holds even for topics in which a high weight is given to the voters environment (τ =0.7).
To answer the questions on the impact of party financing and individual targeting in more detail several extensions to the experiment could be considered. To see the effect of the financing, but especially the targeting in more detail, it might be interesting to look at changes at a cluster level rather than just the overall voting outcome.
Further the impact on outcomes in votes where a lower threshold for τ is given, i.e. in which the impact of an individuals environment is lower, could be considered. To come closer to the real world, different agents could also be assigned with different thresholds for τ. The assignment could be done both in a random manner or possibly by clusters.
Other strategies for the individual targeting could be tested. One option, would be to target the agents, which have many neighbours that could still be converted to a parties opinion first, with the aim of converting clusters rather than just individual agents. How would the results change if the targeting considered how persuadable not only the agent but its neighbours are?
So far the initial opinions were assigned to the agents randomly. Since in the real world we often see that people have similar political views as people in their envi-

ronment, it might be interesting to rerun the simulations with initial opinions being assigned in systematic way, possibly by clusters.

8 Bibliography

Hegselmann, R., U. Krause et al. (2002) Opinion dynamics and bounded confidence models, analysis, and simulation, *Journal of Artificial Societies and Social Simulation*, **5** (3).

Web of Science (2015) *Author Collaboration Network data from 1900-2014*, Thomson Reuters.

A Java Code - Model

A.1 Package Controller

A.1.1 Class AnalyseNetwork

Listing 1: Class AnalyseNetwork

```java
package controller;

import network.ClusterFinder;
import network.Network;
import util.JasonReader;

public class AnalyseNetwork {

  public AnalyseNetwork() {
    // TODO Auto-generated constructor stub
  }

  public static void main(String[] args) {
    // TODO Auto-generated method stub

    String inputFile = "input/coauthornetwork_XA_2008.jsonl";
    //inputFile="input/coauthornetwork_BU_2008.jsonl";


    //prepare network
    System.out.println("read network from file");
    JasonReader jsr = new JasonReader(inputFile);
    Network network = jsr.read();

    //Delate agents without neigbours
    network.deleteZeroNeighbourAgents();

    //Only use biggest component
    ClusterFinder cf = new ClusterFinder();
    cf.reduceToBiggestComponent(network);
    cf.identifyClusters(network);
    //Analyse the graph
    network.analyse();



  }

}
```

A.1.2 Class MultipleMain

Listing 2: Class MultipleMain

```java
package controller;

import java.io.File;
import java.util.LinkedList;

import model.MassMediaCarlos;
import model.Strategy;
import model.strategies.TakeRandom;
import model.strategies.TakeConvincible;
import model.strategies.TakeProtect;
import network.Agent;
import network.ClusterFinder;
import network.Network;
import util.CSVWriter;
import util.GlobalRandom;
import util.JasonReader;
import util.ParamGridReader;
import util.ParamSet;

public class MultipleMain {

  public MultipleMain() {
    // TODO Auto-generated constructor stub
  }

  public static void main(String[] args) {
    // TODO Auto-generated method stub
    System.out.println("lets go");

    //input file
    String gridFile = "input/grid.csv";
    gridFile = "input/peqsgrid20.csv";
    gridFile = "input/detail.csv";

    //time steps
    int maxTS = 200; // ok for small network

    ParamGridReader paramGridReader = new ParamGridReader(gridFile);


    String inputFileSmall = "input/coauthornetwork_XA_2008.jsonl";
    String inputFileBig="input/coauthornetwork_BU_2008.jsonl";


    //Init RNG
    GlobalRandom grand = new GlobalRandom();
    grand.start((long) 1);

    LinkedList<ParamSet> paramGrid = paramGridReader.sets;

    maxTS=300; //jar 3
    maxTS=150; //jar 4

    go(paramGrid, prepNetwork(inputFileSmall), maxTS, new TakeRandom(), new
        TakeRandom(), "D150Out/smallStupidStuipd/");
    go(paramGrid, prepNetwork(inputFileSmall), maxTS, new TakeProtect(), new
        TakeRandom(), "D150Out/smallSwingStupid/");
    go(paramGrid, prepNetwork(inputFileSmall), maxTS, new TakeConvincible(),
        new TakeRandom(), "D150Out/smallPromStupid/");
    go(paramGrid, prepNetwork(inputFileSmall), maxTS, new TakeProtect(), new
        TakeConvincible(), "D150Out/smallSwingProm/");
    go(paramGrid, prepNetwork(inputFileSmall), maxTS, new TakeProtect(), new
        TakeProtect(), "D150Out/smallSwingSwing/");
    go(paramGrid, prepNetwork(inputFileSmall), maxTS, new TakeConvincible(),
        new TakeConvincible(), "D150Out/smallPromProm/");
    go(paramGrid, prepNetwork(inputFileSmall), maxTS, null, null, "D150Out/
        smallNullNull/");

    /*
    go(paramGrid, prepNetwork(inputFileBig), maxTS, new StupidTakeNext(), new
         StupidTakeNext(), "out/bigStupidStuipd/");
    go(paramGrid, prepNetwork(inputFileBig), maxTS, new TakeSwing(), new
        StupidTakeNext(), "out/bigSwingStupid/");
    go(paramGrid, prepNetwork(inputFileBig), maxTS, null, null, "out/
        bigNullNull/");
    go(paramGrid, prepNetwork(inputFileBig), maxTS, new TakeSwing(), new
        TakeSwing(), "out/bigSwingSwing/");
    go(paramGrid, prepNetwork(inputFileBig), maxTS, new TakePromissing(), new
         TakePromissing(), "out/bigPromProm/");
    go(paramGrid, prepNetwork(inputFileBig), maxTS, new TakePromissing(), new
         StupidTakeNext(), "out/bigPromStupid/");
    go(paramGrid, prepNetwork(inputFileBig), maxTS, new TakeSwing(), new
        TakePromissing(), "out/bigSwingProm/");
    */

    System.out.println("all done");
  }

  static Network prepNetwork(String inputFile){
    Network network;
    //prepare network
    System.out.println("SimpleRunTest: read network from file");
    JasonReader jsr = new JasonReader(inputFile);
    network = jsr.read();

    //Delate agents without neigbours
    network.deleteZeroNeighbourAgents();

    //Only use biggest component
    ClusterFinder cf = new ClusterFinder();
    cf.reduceToBiggestComponent(network);
    cf.identifyClusters(network);
    //Analyse the graph
    //network.analyse();
    return network;

  }

  static void go(LinkedList<ParamSet> paramGrid, Network network, int maxTS,
      Strategy stratA, Strategy stratB, String outPath){
    File blankDir = new File(outPath);
    blankDir.mkdir();

    outPath = outPath + Long.toString(System.currentTimeMillis());

    blankDir = new File(outPath);
    blankDir.mkdir();

    CSVWriter writer = new CSVWriter(outPath+"/mruns");


    System.out.println("Number of runs: "+paramGrid.size());


    SimpleRun run;
    int i=0;
    for(ParamSet set : paramGrid){
      //run = new SimpleRun(100000, 4, 0.3, 0.5, 0.2, 0.6, 1, "input/
          coauthornetwork_BU_2008.jsonl");
      //run = new SimpleRun(0.5, 0.2, 0.6, 1, "input/coauthornetwork_XA_2008.
          jsonl");
      if(i%10==0){
        System.out.println("Start run "+i);
      }


      run = new SimpleRun(set.initialPosShare, set.tau, set.posMassMediaShare
          , maxTS, network, stratA, stratB);
      run.record.addConstItem("runNumber", Integer.toString(i));
      run.record.addConstItem("p", Double.toString(set.initialPosShare));
      run.record.addConstItem("theta", Double.toString(set.tau));
      run.record.addConstItem("pMedia", Double.toString(set.posMassMediaShare
          ));

      if(i==0){
        writer.writeAll(run.record.convertToLines(true, true));
      }else{
        writer.writeAll(run.record.convertToLines(true, false));
      }
      writer.flush();
      //run.writeSignalRecord(outPath, i);
      i++;
    }

    writer.close();

  }

}
```

A.1.3 Class SimpleRun

Listing 3: Class SimpleRun

```java
package controller;

import javax.json.JsonReader;

import model.InitializeAgentsOpinions;
import model.MassMediaCarlos;
import model.Strategy;
import network.ClusterFinder;
import network.Network;
import network.NetworkEvaluator;
import network.WattsStrogatz;
import util.CSVWriter;
import util.GlobalRandom;
import util.JasonReader;
import util.Record;

public class SimpleRun {
  Record record = new Record();
  Network network = null;
  NetworkEvaluator evaluator = new NetworkEvaluator();
  WattsStrogatz watts = new WattsStrogatz();
  InitializeAgentsOpinions  init = new InitializeAgentsOpinions();
  MassMediaCarlos carlos;

  int ts=0;

  public SimpleRun(double initPosShare, double tau, double posMassMediaShare,
      int maxTs, Network network, Strategy stratA, Strategy stratB){
    this(0, 0, 0.0, initPosShare, tau, posMassMediaShare, maxTs, network,
        stratA, stratB);
  }

  public SimpleRun(double initPosShare, double tau, double posMassMediaShare,
      int maxTs, Network network){
    this(0, 0, 0.0, initPosShare, tau, posMassMediaShare, maxTs, network,
        null, null);
  }

  public SimpleRun(int numberOfAgents, int avgDegree, double beta, double
      initPosShare, double tau, double posMassMediaShare, int maxTs, Network
      inputNetwork, Strategy stratA, Strategy stratB) {
    // TODO Auto-generated constructor stub
    //Prepare record
    //System.out.println("SimpleRun: Start run");
    record.addRecordItem("posOpinionShare");


    //Prepare network
      if(inputNetwork == null){
        inputNetwork = watts.createSmallWorld(numberOfAgents, avgDegree, beta
            ); //if no network and no input file is given
      }else{
        this.network=inputNetwork;
      }

    //Assign initial opinions
    init.init(network, initPosShare);

    //Add inital state
    record.addTimeStep("posOpinionShare", evaluator.posOpShare(network, 0));


    //Prepare model
    carlos = new MassMediaCarlos(network, tau, posMassMediaShare);

    if(!(stratA == null || stratB == null)){
      //System.out.println("SimpleRun: Prepare Strategies");
      stratA.targetValue=1;
      stratB.targetValue=-1;
      stratA.carlos = carlos;
      stratB.carlos = carlos;
    }

    //Iterate
    ts=1;
    while(ts < maxTs){
      if(stratA == null || stratB == null){
        carlos.performTimeStep(ts);
      }else{
        performStartegyTSParallel(stratA, stratB, posMassMediaShare);
      }
      if(ts%50 == 0 || ts == maxTs-1){
        record.addTimeStep("posOpinionShare", evaluator.posOpShare(network,
            ts));
      }
      ts++;
    }
    network.checkConsitency();
    //System.out.println("SimpleRun: finished run");
  }



  private void performStartegyTSParallel(Strategy stratA, Strategy stratB,
      double posMassMediaShare){
    //Strategies take turns alternating
```

```java
    network.generateAvailableAgents(ts);
    int totalNumber = network.availableAgentsInTS.size();
    int maxA = (int) Math.round(totalNumber*posMassMediaShare); //how many is
         A allowed to hit
    int maxB = totalNumber - maxA;


    int hitA = 0;
    int hitB = 0;
      while(network.availableAgentsInTS.size() >0){
        if(true){//Take turns
        if(hitA<maxA){
          stratA.hitNextAgent(ts);
          hitA++;
        }
        if(hitB<maxB){
          stratB.hitNextAgent(ts);
          hitB++;
        }
        }else{// Draw randomly
          if(GlobalRandom.rand.nextDouble() < posMassMediaShare){
            stratA.hitNextAgent(ts);
          }else{
            stratB.hitNextAgent(ts);
          }
        }
      }
      network.checkConsitency();
    carlos.finishTS(ts);//in case of uneven agent number
    //System.out.println("SimpleRun: finished ts");
  }

  public void writeSignalRecord(String outPath, int runNumber){
    CSVWriter writer = new CSVWriter(outPath+"/zmruns"+runNumber);
    writer.writeAll(carlos.signalRecord.convertToLines(false, true));
    writer.close();
  }

}
```

A.2 Package Model

A.2.1 Class InitializeAgentsOpinions

Listing 4: Class InitializeAgentsOpinions

```java
package model;

import java.util.Random;

import javax.swing.text.StyleContext.SmallAttributeSet;

import util.GlobalRandom;
import network.Agent;
import network.Network;

public class InitializeAgentsOpinions {

  public InitializeAgentsOpinions() {
    // TODO Auto-generated constructor stub
  }

  public void init(Network swNetwork, double sharePos){
    Random rand = GlobalRandom.rand;
    for(Agent agent : swNetwork.agentMap.values()){
      agent.clearRecord();
      if(rand.nextDouble() < sharePos){
        agent.setOpinion(1);
      }else{
        agent.setOpinion(-1);
      }
    }
  }

}
```

A.2.2 Class MassMediaCarlos

Listing 5: Class MassMediaCarlos

```java
package model;

import java.util.Random;

import util.GlobalRandom;
import util.Record;
import network.Agent;
import network.Network;

public class MassMediaCarlos {
  double posMassMedShare;
  public double tau;
  public Network network;
  public Record signalRecord = new Record();

  public MassMediaCarlos(Network network, double tau, double posMassMedShare)
      {
    // TODO Auto-generated constructor stub
    this.network=network;
    this.tau=tau;
    this.posMassMedShare=posMassMedShare;

    signalRecord.addRecordItem("ts");
    signalRecord.addRecordItem("AgentID");
    signalRecord.addRecordItem("inputSignal");
    signalRecord.addRecordItem("previousOpinion");
    signalRecord.addRecordItem("tauPos");
  }

  public int performTimeStep(int ts){
    if(ts < 1){//ts 0 is the initial state. Calling the function for ts 0
        would try to access ts -1
      System.out.println("MassMediaCarlos: WARNING: Called perform TimeStep
          for ts 0");
      return -1;
    }
    //System.out.println("MassMediaCarlos: perform ts "+ts);
    for(Agent agent : network.getAllAgents()){
      double opa = getOpinionToApply(); //opinion to apply
      hitAgent(agent, opa, ts);
    }
    return 0;
  }

  public double getOpinionToApply(){
    Random rand = GlobalRandom.rand;
    if(rand.nextDouble() < posMassMedShare){
      return 1;
    }else{
      return -1;
    }
  }

  public int hitAgent(Agent agent, double opa, int ts){
    //if(agent.getOpinionRecord().size()-1 >= ts){  //Deactivated to save
        time
    //  System.out.println("MassMediaCarlos: WARNING: Tried to hit agent
        twice");
    //  return -200;//Agent already has an opinion for this ts
    //}
    //System.out.println("MassMediaCarlos: hit agent "+agent.id);
    double opAgent = agent.getOpinionAt(ts-1);
    double tauPos = network.getTauPos(agent, ts-1);

    if(ts%50==0 || ts<5){
      signalRecord.addTimeStep("ts", ts);
      signalRecord.addTimeStep("AgentID", agent.id);
      signalRecord.addTimeStep("inputSignal", opa);
      signalRecord.addTimeStep("previousOpinion", opAgent);
      signalRecord.addTimeStep("tauPos", tauPos);
    }

    //Remove from lists (this might cost time)
    network.availableAgentsInTS.remove(agent); //Everybody only once per ts
    if(opAgent==1){
      network.availablePosAgentsInTS.remove(agent);
    }else{
      network.availableNegAgentsInTS.remove(agent);
    }

    //Check whether opinions coincide
    if(opa == opAgent){//Agent already has this opinion
      agent.setOpinion(opAgent); //Keep your opinion
      return 0;
    }

    //check for adoption
    if(opa == 1 && tauPos > tau){
      agent.setOpinion(opa);
      //System.out.println("MassMediaCarlos: op changed to pos");
      return 1;//changed opinion to 1
    }
    if(opa == -1 && (1-tauPos)>tau){
      agent.setOpinion(opa);
      //System.out.println("MassMediaCarlos: op changed to neg");
      return -1;//changed opinion to -1
    }

    agent.setOpinion(opAgent); //Keep your opinion

    return 0;//no change
  }

  public void finishTS(int ts){
    //Set ts opinion for all agents who do not have one yet
    int shortCounter = 0;
    for(Agent agent : network.getAllAgents()){
      if(agent.getOpinionRecord().size()-1 < ts){//agent does not have an
          opinion for the ts
        agent.setOpinion(agent.getCurrentOpinion()); //keep opinion
        shortCounter++;
      }
    }
    //System.out.println("MassMediaCarlos: Number of untreated agents: "+
        shortCounter);
  }


}
```

A.2.3 Class Strategy

Listing 6: Class Strategy

```java
package model;

import network.Agent;
import network.Network;

public class Strategy {
  public double targetValue=0; //Opinion value the stratgy should apply
  public MassMediaCarlos carlos;
  public Strategy() {
    // TODO Auto-generated constructor stub

  }

  public Agent selectNextAgent(int ts){
    return null;
  }

  public void performTS(int contingent, int ts){
    int i=0;
    while(i<contingent){
      Agent agent = selectNextAgent(ts);
      if(hitAgent(agent, ts) != -200){//-200 >> agent was already hit, take
          an other one
        //but the selecting algorithm should only select available agents
        i++;
      }
    }
  }

  public int hitNextAgent(int ts){
    if(hitAgent(selectNextAgent(ts), ts) == -200){//try to hit the next
        selcted agent until you find one
      System.out.println("Strategy: Tried to hit unavailable agent");
    }
    return 0;//0 >> io; -1 >> no more agents
  }

  public int hitAgent(Agent agent, int ts){
    return carlos.hitAgent(agent, targetValue, ts);
  }


}
```

A.3 Package Model.Strategies

A.3.1 Class TakeRandom

Listing 7: Class TakeRandom

```java
package model.strategies;

import util.GlobalRandom;
import network.Agent;
import model.MassMediaCarlos;
import model.Strategy;

public class TakeRandom extends Strategy {

  public TakeRandom() {
    super();
    // TODO Auto-generated constructor stub
  }
  @Override
  public Agent selectNextAgent(int ts){
    int count = this.carlos.network.availableAgentsInTS.size();
    int index = GlobalRandom.rand.nextInt(count);


    return this.carlos.network.availableAgentsInTS.get(index);
  }

}
```

A.3.2 Class TakeConvincible

Listing 8: Class TakeConvincible

```java
package model.strategies;

import java.util.LinkedList;

import network.Agent;
import model.Strategy;

public class TakeConvincible extends Strategy {

  public TakeConvincible() {
    // TODO Auto-generated constructor stub
  }

  @Override
  public Agent selectNextAgent(int ts){
    LinkedList<Agent> list;
    list=this.carlos.network.availableAgentsInTS;
    //For performance
    if(this.targetValue == 1 && this.carlos.network.availableNegAgentsInTS.
        size() >0){
      list=this.carlos.network.availableNegAgentsInTS; // You dont want to
          work on those who like you anyways
      //System.out.println("TakePromessing: Take Neg fast list");
    }
    else if(this.targetValue == -1 && this.carlos.network.
        availablePosAgentsInTS.size() >0){
      list=this.carlos.network.availablePosAgentsInTS; // You dont want to
          work on those who like you anyways
    }


    for(Agent agent : list){
      //if(agent.getCurrentOpinion()==targetValue){ //Should be coverd by the
            list now
      //  continue; //makes it damn slow!
      //}
      double tauPos = carlos.network.getTauPos(agent, ts-1);
      if(targetValue == 1 && tauPos>carlos.tau){
        //System.out.println("TakePromissing: found +1 agent");
        return agent;
      }
      if(targetValue == -1 && (1-tauPos)>carlos.tau){
        //System.out.println("TakePromissing: found -1 agent");
        return agent;
      }

```

```
        }

        return this.carlos.network.availableAgentsInTS.getFirst();
    }

}
```

A.3.3 Class TakeProtect

Listing 9: Class TakeProtect

```java
package model.strategies;

import java.util.LinkedList;

import network.Agent;
import model.Strategy;

public class TakeProtect extends Strategy {

  public TakeProtect() {
    // TODO Auto-generated constructor stub
  }

  @Override
  public Agent selectNextAgent(int ts){
    //first safe yours which could go to the other side
    //then steel from the others

    LinkedList<Agent> list;
    LinkedList<Agent> yours = null;
    LinkedList<Agent> others = null;
    list=this.carlos.network.availableAgentsInTS;

    if(this.targetValue == 1){
      yours = this.carlos.network.availablePosAgentsInTS;
      others = this.carlos.network.availableNegAgentsInTS;
    }
    else if(this.targetValue == -1){
      others = this.carlos.network.availablePosAgentsInTS;
      yours = this.carlos.network.availableNegAgentsInTS;
    }

    //safe yours
    for(Agent agent : yours){
      double tauPos = carlos.network.getTauPos(agent, ts-1);
      if(targetValue == 1 && (1-tauPos) < carlos.tau){
        continue; //safe
      }

      if(targetValue == -1 && (tauPos) < carlos.tau){
        continue; //safe
      }

      //System.out.println("TakeSing: found one");
      return agent;
    }

    //Steel others which you can get
    for(Agent agent: others){
      double tauPos = carlos.network.getTauPos(agent, ts-1);
      if(targetValue == 1 && (tauPos) < carlos.tau){
        continue; //lost anyway
      }

      if(targetValue == -1 && (1-tauPos) < carlos.tau){
        continue; //lost anyway
      }
      //System.out.println("TakeSing: found one");
      return agent;
    }

    return this.carlos.network.availableAgentsInTS.getFirst();
  }

}
```

A.4 Package Network

A.4.1 Class Agent

Listing 10: Class Agent

```
package network;

import java.util.LinkedList;

public class Agent {
public int id;
public int cluster = -1;
public double tauPos = -1.0;
private LinkedList<Double> opinionRecord = new LinkedList<Double>();
//Using an array would probably result in better performance but
//the linked list is comfortable for coding since we do not
//Need to define the length at the beginning and we dont need to
//Pass the time step to the agent
//BUT we have to make sure that each agent gets his opinin changed
//exactly once per time step


  public Agent(int id) {
    // TODO Auto-generated constructor stub
    this.id = id;
  }

  public double getCurrentOpinion(){ //If used during opinion update this can
       be one TS a head
    return opinionRecord.getLast();
  }

  public void setOpinion(double d) {
    // TODO Auto-generated method stub
    opinionRecord.add(d);
  }

  public double getOpinionAt(int ts){
    return opinionRecord.get(ts);
  }

  public int getSizeOpinionList(){
    return opinionRecord.size();
  }

  public void clearRecord(){
    opinionRecord = new LinkedList<Double>();
  }

```

```java
  public LinkedList<Double> getOpinionRecord(){
    return this.opinionRecord;
  }

}
```

A.4.2 Class ClusterFinder

Listing 11: Class ClusterFinder

```java
package network;

import java.util.LinkedList;

public class ClusterFinder {
  Network network;
  int[] clusters;
  int maxId = 0;
  int maxSize = 0;

  public ClusterFinder() {
    // TODO Auto-generated constructor stub
  }

  public void identifyClusters(Network network){
    System.out.println("ClusterFinder: start identifying");
    //Use -Xss5M to aviod stack overflow >> huge recursion
    this.network=network;
    clusters = new int[network.graph.getVertexCount()]; //Worth case ervry
        agent on its own
    java.util.Arrays.fill(clusters, 0);
    int currentCluster = 0;

    for(Agent agent : network.getAllAgents()){
      agent.cluster=-1;
    }


    while(true){
      Agent start = null;
      //boolean everyNodeInACluster = true;
      for(Agent agent : network.getAllAgents()){
        if(agent.cluster == -1){//Agent is not yet in a cluster
          start = agent;
        }
      }
      if(start == null){
        break; //Every agent is in a cluster
      }
      mark(start, currentCluster);

      currentCluster++;
    }

    //Print and identify biggest component
    for(int i = 0; i<clusters.length; i++){
      //System.out.println("Cluster "+i+" size: "+clusters[i]);
      if(clusters[i] > maxSize){
        maxSize=clusters[i];
        maxId=i;
      }
      if(clusters[i]==0){
        break;
      }
    }
    System.out.println("ClusterFinder: fineshed identifying");
  }

  void mark(Agent agent, int clusterID){
    agent.cluster=clusterID;
    clusters[clusterID]++;
    for(Agent neighbor : network.graph.getNeighbors(agent)){
      if(neighbor.cluster == -1){
        mark(neighbor, clusterID);
      }
    }
  }

  public void reduceToBiggestComponent(Network network){
    System.out.println("ClusterFinder: start reducing");
    this.identifyClusters(network);
    LinkedList<Agent> agentsToRemove = new LinkedList<Agent>();
    for(Agent agent : network.getAllAgents()){
      if(agent.cluster != this.maxId){
        agentsToRemove.add(agent);
      }
    }
    network.deleteAgents(agentsToRemove);
    System.out.println("ClusterFinder: fineshed reducing");
  }

}
```

A.4.3 Class MEdge

Listing 12: Class MEdge

```java
package network;

public class MEdge {

  public MEdge() {
    // TODO Auto-generated constructor stub
  }

}
```

A.4.4 Class Network

Listing 13: Class Network

```java
package network;

import java.util.Collection;
import java.util.HashMap;
import java.util.LinkedList;
import java.util.Map;

import com.google.common.base.Function;

import edu.uci.ics.jung.algorithms.metrics.Metrics;
import edu.uci.ics.jung.algorithms.shortestpath.DistanceStatistics;
import edu.uci.ics.jung.graph.SparseGraph;
import edu.uci.ics.jung.graph.UndirectedSparseGraph;


public class Network {
  public HashMap<Integer, Agent> agentMap = new HashMap<Integer, Agent>();
  public UndirectedSparseGraph<Agent, MEdge> graph = new
      UndirectedSparseGraph<Agent, MEdge>();
  public LinkedList<Agent> availableAgentsInTS = new LinkedList<Agent>(); //
      holds agents which were not hit yet
  public LinkedList<Agent> availablePosAgentsInTS = new LinkedList<Agent>();
      //holds agents which were not hit yet
  public LinkedList<Agent> availableNegAgentsInTS = new LinkedList<Agent>();
      //holds agents which were not hit yet
  int tauPosTS=-1; //Network calculates all tau pos for this ts. If a ts !=
      this is requested they have to be recalculated

  public Network() {
    // TODO Auto-generated constructor stub
  }

  public void addAgent(Agent agent){
    agentMap.put(agent.id, agent);
    graph.addVertex(agent);
  }

  public int addEdge(Agent a, Agent b){
    if(graph.isNeighbor(a, b)){
      return -1;
    }
    if(a == b){
      return -1;
    }
    graph.addEdge(new MEdge(), a, b);
    return 1;
  }

  public int addEdgeByIDs(int a, int b){
    if(!this.agentMap.containsKey(a) || !this.agentMap.containsKey(b)){
      return -1;
    }
    Agent aA = this.agentMap.get(a);
    Agent aB = this.agentMap.get(b);
    return this.addEdge(aA, aB);
  }

  public Collection<Agent> getAllAgents(){
    return agentMap.values();
  }

  public Agent getAgent(int id){
    return agentMap.get(id);
  }

  public int checkConsitency(){
    int lengthOpList = 0;
    for(Agent agent : getAllAgents()){
      //Check length of opinion list
      if(lengthOpList == 0){
        lengthOpList=agent.getSizeOpinionList();
      }else{
        if(lengthOpList != agent.getSizeOpinionList()){
          System.out.println("Network: WARNING: Size of opinion records differ!");
          return-1;
        }
      }

      //check opinion
      if(agent.getCurrentOpinion() < -1 || agent.getCurrentOpinion() > 1){
        System.out.println("Network: WARNING: Opinion out of range -1 to 1");
        return-1;
      }

    }

    //System.out.println("Network:consistency check ok");
    return 0;
  }

  public double getTauPos(Agent agent, int ts){
    if(this.tauPosTS != ts){
      //System.out.println("Network: Calculate t pos for ts" + ts);
      for(Agent currentAgent : this.getAllAgents()){
        currentAgent.tauPos=this.getTauPosCalc(currentAgent, ts);
      }
      this.tauPosTS=ts;
    }
    return agent.tauPos;
  }

  public double getTauPosCalc(Agent agent, int ts){
    //Returns the share of neighbours having opinion larger than zero in ts
    int neighbours = 0;
    int posNeighbours = 0;
    for(Agent neighbour : graph.getNeighbors(agent)){
      neighbours ++;
      if(neighbour.getOpinionAt(ts) > 0){
        posNeighbours++;
      }
    }
    if(neighbours == 0){
      System.out.println("Network: WARNING: Calculated tau for agent without
          neighbours");
      return 0;
    }
    //System.out.println(posNeighbours+" "+neighbours);
    return 1.0*posNeighbours/neighbours;
  }

  public void deleteZeroNeighbourAgents(){
    int count = 0;
    LinkedList<Agent> removeAgents = new LinkedList<Agent>();
    for(Agent agent : getAllAgents()){
      if(graph.getNeighborCount(agent) == 0){
        removeAgents.add(agent);
        count++;
      }
    }

    deleteAgents(removeAgents);

    System.out.println("Network: Number of removed agents: "+count);
  }

  public void deleteAgents(LinkedList<Agent> agentDelList){
    for(Agent agent : agentDelList){
      graph.removeVertex(agent);
      agentMap.remove(agent.id);
    }
  }

  public void printAllCurrentOpinions(){
    System.out.println("opinions");
    for(Agent agent : getAllAgents()){
      System.out.print(agent.getCurrentOpinion()+1+"\t");
    }
    System.out.println("---");
    return;
  }

  public void analyse(){
    System.out.println("Network: Calculate diameter");
    Double diameter = DistanceStatistics.diameter(graph);
    System.out.println("Network: Calculate Clustering Coefficients");
    Map<Agent, Double> clusteringCoeff = Metrics.clusteringCoefficients(graph
        );
    System.out.println("Netowrk: Diameter: " + diameter);

    System.out.println("Network: Calculate average distances");
    Function<Agent, Double> avgDist = DistanceStatistics.averageDistances(
        graph);

    double overAllAvgDist = 0;
    double totalAvgDists = 0;
    double avgClusterCoeff = 0;
    double totalOfClusterCoeff=0;
    double degreeSum = 0;
    double averageDegree = 0;
    for(Agent agent : this.getAllAgents()){
      //System.out.println("Network: Agents avg Dist: "+avgDist.apply(agent))
          ;
      totalAvgDists=totalAvgDists+avgDist.apply(agent);
      //System.out.println("Network: Agents clust coeff: "+avgDist.apply(
          agent));
      totalOfClusterCoeff=totalOfClusterCoeff+clusteringCoeff.get(agent);
      degreeSum=degreeSum+graph.degree(agent);
    }
    overAllAvgDist=totalAvgDists/graph.getVertexCount();
    avgClusterCoeff=totalOfClusterCoeff/graph.getVertexCount();
    averageDegree=degreeSum/graph.getVertexCount();
    System.out.println("Network: Average of average distances "+
        overAllAvgDist);
    System.out.println("Network: Average of local clustering coefficients "+
        avgClusterCoeff);
    System.out.println("Network: Average Degree "+averageDegree);

  }

  public void generateAvailableAgents(int ts){ // collect all the agents
      which got not yet hit
    availableAgentsInTS = new LinkedList<Agent>();
    availablePosAgentsInTS = new LinkedList<Agent>();
    availableNegAgentsInTS = new LinkedList<Agent>();

    for(Agent agent : getAllAgents()){
      if(agent.getOpinionRecord().size()-1 < ts){
        availableAgentsInTS.add(agent);
        if(agent.getCurrentOpinion()==1){
          availablePosAgentsInTS.add(agent);
          //System.out.println("Network: Added agent to pos fast list");
        }else{
          availableNegAgentsInTS.add(agent);
          //System.out.println("Network: Added agent to neg fast list");
        }

      }
    }
  }

}
```

A.4.5 Class NetworkEvaluator

Listing 14: Class NetworkEvaluator

```java
package network;

import edu.uci.ics.jung.graph.Graph;

public class NetworkEvaluator {

  public NetworkEvaluator() {
    // TODO Auto-generated constructor stub
  }

  public double posOpShare(Network network, int ts){
    int pos = 0;

    for(Agent agent : network.getAllAgents()){
      if(agent.getOpinionAt(ts)>0){
        pos++;
      }
    }

    return 1.0*pos/network.graph.getVertexCount();
  }





}
```

A.4.6 Class WattsStrogatz

Listing 15: Class WattsStrogatz

```java
package network;

import java.util.Random;

import javax.swing.text.StyleContext.SmallAttributeSet;

import util.GlobalRandom;

public class WattsStrogatz {
  int n;
  int k;
  double beta;
  Network swNetwork = new Network();

  public WattsStrogatz() {
    // TODO Auto-generated constructor stub
  }

  public Network createSmallWorld(int n, int k, double beta){//Number of
      nodes, average degree (should be even?), spatial param.
    this.n=n;
    this.k=k;
    this.beta=beta;

    if(k % 2 == 1){
      System.out.println("WattsTrogatz: k is univen that might cause a
          problem");
    }



    //Create n agents
    for(int i=0; i<n; i++){
      swNetwork.addAgent(new Agent(i));
    }

    //Create circular edges
    System.out.println("WatzStrogatz: start building circular edges");
    for(int i=0; i<n; i++){//Not efficient not all of them needed anyways
        doubles and loops are refused by network class
      for(int dist = -k/2; dist<= k; dist++){
        int other = circleId(i+dist);
        swNetwork.addEdgeByIDs(i, other);

      }
      //System.out.println("Edges for verted "+i+" done");
    }


    //Rewire
    System.out.println("WattsStrogatz: start rewireing");
    Random bethaRand = GlobalRandom.rand;
    for(int i=0; i<n; i++){//Not efficient not all of them needed anyways
        doubles and loops are refused by network class
      Agent current = swNetwork.agentMap.get(i);
      //System.out.println("Rewire vertex "+i);
      for(MEdge ege : swNetwork.graph.getOutEdges(current)){
        Agent neighBour = swNetwork.graph.getOpposite(current, ege);
        if(neighBour.id<i){
          //System.out.println("backward link");
          continue;
        }
        //System.out.println("forward link");
        if(bethaRand.nextDouble()<this.beta){
          reWire(ege, current);
        }
      }
    }

    return swNetwork;
  }

  int circleId(int id){
    if(id >= 0 && id<n){
      return id;
    }
    if(id<0){
      return n+id;
    }
    if(id>n-1){
      return id-n;
    }
    return 0;
  }

  void reWire(MEdge oldE, Agent baseVertex){
    //Base vertex gets reconected to some other vertex
    Random rand = GlobalRandom.rand;

    //Remove old edge
    this.swNetwork.graph.removeEdge(oldE);

    //Choose uniform between vertexes until you find a valid one (No self
        loop no double)
    while(true){
      int newId = rand.nextInt(n);
```

```
        int success = swNetwork.addEdgeByIDs(newId, baseVertex.id);
        if(success == 1){
          return;
        }
      }


  }

}
```

A.5 Package Util

A.5.1 Class CSVReader

Listing 16: Class CSVReader

```java
package util;



import java.io.BufferedReader;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;
import java.util.HashMap;
import java.util.LinkedList;

public class CSVReader {
  boolean jsonl = false;
  public  LinkedList<String[]> readCSV(String fileName){
    return readCSV(fileName, "\t");
  }

  public  LinkedList<String[]> readJSONL(String fileName){
    jsonl=true;
    return readCSV(fileName, ",");
  }


  public  LinkedList<String[]> readCSV(String fileName, String trennzeichen){
    BufferedReader reader = null;
    LinkedList<String[]> liste = new LinkedList<String[]>();
    try {
      reader = new BufferedReader(new FileReader(fileName));
    } catch (FileNotFoundException e1) {
      // TODO Auto-generated catch block
      e1.printStackTrace();
    }
    String zeile = "";

    try {
      zeile = reader.readLine();
      while ((zeile = reader.readLine()) != null) {
        if(jsonl){
          zeile = zeile.replace("[", "").replace("]", "");
        }
        String[] felder = zeile.split(trennzeichen,0);
        if(felder.length>1 || true){  // Wenn nur eine spalte 0 setzen //
            jsonl file has empty agents >> true
          liste.add(felder);
        }
      }
    } catch (IOException e) {
      // TODO Auto-generated catch block
      e.printStackTrace();
    } //header
    try {
      reader.close();
    } catch (IOException e) {
      // TODO Auto-generated catch block
      e.printStackTrace();
    }
    return liste;
  }
}
```

A.5.2 Class CSVWriter

Listing 17: Class CSVWriter

```java
package util;


import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.util.LinkedList;

public class CSVWriter {
  File file;
  FileWriter fileWriter;


  public CSVWriter(String fileName){
    file=new File (fileName+".csv");
    fileWriter = null;
    try {
      fileWriter = new FileWriter(file);
      //DataIng_Envi.debug_print("CSV Wrtier: opened file "+fileName);
      //System.out.println("File offen");
    } catch (IOException e) {
      // TODO Auto-generated catch block
      //DataIng_Envi.debug_print("CSV Wrtier: open file "+fileName+" failed")
          ;
      e.printStackTrace();
    }
  }

  public void writeAll(LinkedList<LinkedList<String>> zeilen){
    for (LinkedList<String> zeile : zeilen){
      String zeilenText = null;
      for(String spalte : zeile){
        if(zeilenText!=null){
          zeilenText=zeilenText+",";
        }else{
          //System.out.println("neuer zeilentext");
          zeilenText="";
        }
        if(spalte==null){
          spalte=" ";
        }


        zeilenText=zeilenText+spalte;
      }
      writeLine(zeilenText);
    }

  }

  public void writeLine(String text){
    try {
      fileWriter.write(text+"\n");
    } catch (IOException e) {
      // TODO Auto-generated catch block
      e.printStackTrace();
    }
  }

  public void writeLine(LinkedList<String> zeile){
    String zeilenText = null;
    for(String spalte : zeile){
      if(zeilenText!=null){
        zeilenText=zeilenText+",";
      }else{
        //System.out.println("neuer zeilentext");
        zeilenText="";
      }
      if(spalte==null){
        spalte=" ";
      }


      zeilenText=zeilenText+spalte;
    }
    writeLine(zeilenText);
  }

  public void close(){
    try {
      fileWriter.close();
    } catch (IOException e) {
      // TODO Auto-generated catch block
      e.printStackTrace();
    }
  }

  public void flush(){
    try {
      fileWriter.flush();
    } catch (IOException e) {
      // TODO Auto-generated catch block
      e.printStackTrace();
    }
  }

```

```
}
```

A.5.3 Class GlobalRandom

Listing 18: Class GlobalRandom

```
package util;

import java.util.Random;

public class GlobalRandom {
  //Holds global random generator to be able to controll for seeds
  public static Random rand;
  public static long seed;
  public GlobalRandom() {
    // TODO Auto-generated constructor stub


  }
  public void start(long seed){
    this.seed=seed;
    this.rand = new Random(seed);
  }

}
```

A.5.4 Class JasonReader

Listing 19: Class JasonReader

```java
package util;

import java.util.LinkedList;

import network.Agent;
import network.Network;


public class JasonReader {

  String fileName;

  public JasonReader(String fileName) {
    this.fileName = fileName;
  }

  public Network read(){
    CSVReader reader = new CSVReader();
    LinkedList<String[]> input = reader.readJSONL(fileName);
    Network jNetwork = new Network();

    int agentCount = input.size();

    System.out.println("Number of agents in file: "+agentCount);


    for(int i = 0; i<agentCount; i++){
      jNetwork.addAgent(new Agent(i));
    }

    int id = 0;
    for(String[] line : input){
      for(int j = 0; j<line.length; j++){
        if(line[j].equals("")){
          continue;
        }
        int secondID = Integer.parseInt(line[j]);
        if(secondID > agentCount){
          System.out.println("Link to missing agent (id to high): "+secondID)
              ;
        }else{
          jNetwork.addEdgeByIDs(id, secondID);
        }
      }
      id++;
    }

    System.out.println("done reading");
    return jNetwork;
  }

}
```

A.5.5 Class ParamGridReader

Listing 20: Class ParamGridReader

```java
package util;

import java.util.LinkedList;

public class ParamGridReader {
  public LinkedList<ParamSet> sets = new LinkedList<ParamSet>();
  public ParamGridReader(String gridFileName) {
    // TODO Auto-generated constructor stub
    CSVReader csv = new CSVReader();
    LinkedList<String[]> importList = csv.readCSV(gridFileName, ",");

    boolean head = true;
    for(String[] line : importList){
      if(head){
        head=false;
        continue;
      }

      ParamSet set = new ParamSet();
      set.initialPosShare = Double.parseDouble(line[1]); //index 0 is run
          number
      set.tau = Double.parseDouble(line[2]);
      set.posMassMediaShare = Double.parseDouble(line[3]);
      sets.add(set);
    }


  }

}
```

A.5.6 Class ParamSet

Listing 21: Class ParamSet

```java
package util;

public class ParamSet {
  public double initialPosShare;
  public double tau;
  public double posMassMediaShare;

  public ParamSet() {
    // TODO Auto-generated constructor stub
  }

}
```

A.5.7 Class Record

Listing 22: Class Record

```java
package util;

import java.util.HashMap;
import java.util.LinkedList;


public class Record {
  HashMap<String, LinkedList<String>> map = new HashMap<String, LinkedList<
      String>>();
  //<Name of Variable, <Values over time>

  public Record() {
    // TODO Auto-generated constructor stub
  }

  public void addConstItem(String name, String value){
    LinkedList<String> constV = new LinkedList<String>();
    for(int ts = 0; ts < map.values().iterator().next().size(); ts++){
      constV.add(value);
    }
    this.addRecordItem(name, constV);
  }

  public void addRecordItem(String name){
    this.addRecordItem(name, new LinkedList<String>());
  }

  public void addRecordItem(String name, LinkedList<String> values){
    map.put(name, values);
  }

  public void addTimeStep(String itemName, double value){
    this.map.get(itemName).add(Double.toString(value));
  }

  public void addRecordItemDouble(String name, LinkedList<Double> values){
    LinkedList<String> stringList = new LinkedList<String>();
    for(double value : values){
      stringList.add(Double.toString(value));
    }
    this.addRecordItem(name, stringList);
  }

  public LinkedList<LinkedList<String>> convertToLines(boolean
      generateTimeSteps, boolean writeHeads){
    LinkedList<LinkedList<String>> lines = new LinkedList<LinkedList<String
```

```
        >>();
    LinkedList<String> heads = new LinkedList<String>();

    //Create ts column
    if(generateTimeSteps){
      LinkedList<Double> timeSteps = new LinkedList<Double>();
      for(int ts = 0; ts < map.values().iterator().next().size(); ts++){
        timeSteps.add(ts*1.0);
      }
      this.addRecordItemDouble("ts", timeSteps);
    }

    //Get heads
    for(String head : map.keySet()){
      heads.add(head);
    }
    if(writeHeads){
      lines.add(heads);
    }

    //Get lines
    for(int i = 0; i < map.get(heads.getFirst()).size(); i++){
      LinkedList<String> line = new LinkedList<String>();
      for(String head : heads){
        line.add(map.get(head).get(i));
      }
      lines.add(line);
    }

    return lines;
  }

}
```

A.5.8 Class Viewer

Listing 23: Class Viewer

```java
package util;

import java.awt.Color;
import java.awt.Dimension;
import java.awt.Paint;
import java.awt.Shape;
import java.awt.geom.AffineTransform;
import java.awt.geom.Ellipse2D;

import javax.swing.JFrame;

import com.google.common.base.Function;

import edu.uci.ics.jung.algorithms.layout.CircleLayout;
import edu.uci.ics.jung.graph.Graph;
import edu.uci.ics.jung.visualization.VisualizationViewer;
import edu.uci.ics.jung.visualization.control.CrossoverScalingControl;
import edu.uci.ics.jung.visualization.control.ScalingControl;
import network.Agent;
import network.MEdge;
import network.Network;

public class Viewer {
  VisualizationViewer vv;
  public Viewer() {
    // TODO Auto-generated constructor stub
  }

  public void showGraph(Network network){
    // Color Transformer
        Function<Agent, Paint> vertexPaintTrans = new Function<Agent, Paint
            >() {
          public Paint apply(Agent a) {
            if(a.getCurrentOpinion()>0){
              return Color.GREEN;
            }
            return Color.RED;
          }
        };

        // Shape Transformer
        Function<Agent,Shape> vertexSize = new Function<Agent,Shape>(){
            public Shape apply(Agent a){
                Ellipse2D circle = new Ellipse2D.Double(-5, -5, 10, 10);
                // in this case, the vertex is twice as large
                return AffineTransform.getScaleInstance(.7, .7).
```

```
                        createTransformedShape(circle);
                }
            };


            Dimension preferredSize = new Dimension(1000, 1000);

            JFrame jf = new JFrame();
        Graph g = network.graph;
        CircleLayout<Agent, MEdge> myLayout = new CircleLayout<Agent, MEdge>(g);
        myLayout.setSize(preferredSize);

        vv = new VisualizationViewer(myLayout);
        vv.getRenderContext().setVertexFillPaintTransformer(vertexPaintTrans);
        vv.getRenderContext().setVertexShapeTransformer(vertexSize);
        vv.setPreferredSize(preferredSize);

        jf.getContentPane().add(vv);
            jf.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
            jf.pack();
            jf.setVisible(true);

    }

    public void reDraw(){
        vv.repaint();
    }

}
```

B R Code - Analysis

B.1 Evaluate Set

Listing 24: main.R

```r
rm(list=ls(all=TRUE))
setwd("~/git/mss/MSS/R")
library("ggplot2")
library("plyr")
library("RColorBrewer")
source("defineBeamerPlot.R")



runDir = "testRun";
runDir = "peqs20";
runDir = "t70fix20";
runDir = "t70fix20PromVsStupid";
runDir = "t790fix20SwingVsStupid";
runDir = "1481575424202";
runDir = "1481584054667";

setwd(paste("~/git/mss/MSS/output/", runDir, sep=""))

setwd("~/git/mss/MSS/output/1481660744993/")
setwd("~/git/mss/MSS/D150Out/smallSwingStupid/1481926279694/")

source("~/git/mss/MSS/R/generateStandardPlots.R")




# Check whether number of time steps was sufficient ----------------------------------
maxTs = max(record$ts)
plot(density(ddply(record, ~runNumber, summarize, diff12=posOpinionShare[1]-posOpinionShare[2])$diff12))
plot(density(ddply(record, ~runNumber, summarize, diff56=posOpinionShare[maxTs-1]-posOpinionShare[maxTs])$diff56))
```

B.2 Generate Standard Plots

Listing 25: generateStandardPlots.R

```

# Prepare Data ————————————————————————————————————
#Load
record <- read.csv(file="mruns.csv",sep=",",head=TRUE)

#Look only at final TS
finalStep = subset(record, record$ts==max(record$ts))


#Aggregate Runs with similar Input
finalStep$combinationIdentifier = finalStep$theta*100000+finalStep$p*1000+
    finalStep$pMedia*10
finalStep$Win =  ifelse(finalStep$posOpinionShare>0.5,1,0)

aggregated = ddply(finalStep,~combinationIdentifier,summarise,pEnd=mean(
    posOpinionShare),sdPEnd=sd(posOpinionShare), p=mean(p), s=mean(pMedia), t
    =mean(theta), n=length(theta), pWin = mean(Win))
aggregated$Win = ifelse(aggregated$pEnd>0.5,1,0)




# Mean Outcome P0 vs T ————————————————————————————————
#Means
ggplot(aggregated, aes(x=t, y=p))+
  geom_tile(aes(fill=pEnd))+
  #scale_fill_gradientn(name="Share of Proponents \nin Final Time Step\n ",
      colours=brewer.pal(9,"RdYlGn"))+
  scale_fill_gradientn(name=expression(p[final]) ,colours=brewer.pal(9,"
      RdYlGn"))+
  beamerPlot+
  xlab(expression(tau))+
  #ylab(paste(expression(p[0]), "Initial Share of Proponents"))
  ylab(expression(p[0]))
#  geom_abline(slope=1, intercept= 0)+
#  geom_abline(slope=-1, intercept= 1)+
#  geom_abline(intercept = 0.7, slope = 0)
ggsave(file="MeanOtucomePvsT.pdf", width=16, height=10, dpi=100)



# Mean Outcome P0 vs S ————————————————————————————————
#Means
ggplot(aggregated, aes(x=p, y=s))+
  geom_tile(aes(fill=pEnd))+
  scale_fill_gradientn(name=expression(p[final]) ,colours=brewer.pal(9,"
      RdYlGn"))+
  beamerPlot+
  xlab(expression(p[0]))+
  ylab(expression(p[M]))
ggsave(file="MeanOtucomePvsS.pdf", width=16, height=10, dpi=100)



# Chance To Win P0 vs S ----------------------------------------------------------
ggplot(aggregated, aes(x=p, y=s))+
  geom_tile(aes(fill=pWin))+
  scale_fill_gradientn(name="Share of \nRuns Won\nby Proponents\n" ,colours=
      brewer.pal(9,"RdYlGn"))+
  beamerPlot+
  xlab(expression(p[0]))+
  ylab(expression(p[M]))
ggsave(file="PWinPvsS.pdf", width=16, height=10, dpi=100)




# Win Loose P0 vs S --------------------------------------------------------------
ggplot(aggregated, aes(x=p, y=s))+
  geom_tile(aes(fill=as.factor(Win)))+
  #scale_fill_discrete(name=expression(p[final]))+
  scale_fill_manual(name="Proponents:", values=c("0"="red", "1"="darkgreen"),
       labels=c("0"="Loose", "1"="Win"))+
  beamerPlot+
  xlab(expression(p[0]))+
  ylab(expression(p[M]))
ggsave(file="WinPvsS.pdf", width=16, height=10, dpi=100)


# n  p0 vs s ---------------------------------------------------------
ggplot(aggregated, aes(x=p, y=s))+
  geom_tile(aes(fill=n))+
  scale_fill_continuous(name=expression(n))+
  beamerPlot+
  xlab(expression(p[0]))+
  ylab(expression(p[M]))
ggsave(file="nPvsS.pdf", width=16, height=10, dpi=100)


# sd  p0 vs s --------------------------------------------------------
ggplot(aggregated, aes(x=p, y=s))+
  geom_tile(aes(fill=sdPEnd))+
  scale_fill_continuous(name=expression(sigma(p)["final"]))+
  beamerPlot+
  xlab(expression(p[0]))+
  ylab(expression(p[M]))
```

```
ggsave(file="sdPvsS.pdf", width=16, height=10, dpi=100)


# Standard Dev p vs t ------------------------------------------------
ggplot(aggregated, aes(x=t, y=p))+
  geom_tile(aes(fill=sdPEnd))+
  scale_fill_continuous(name=expression(sigma(p)["final"]))+
  beamerPlot+
  xlab(expression(tau))+
  ylab(expression(p[M]))
ggsave(file="sdpvst.pdf", width=16, height=10, dpi=100)








# p(s) over time -----------------------------------------------------
#ggplot(record)+
#  geom_line(aes(x=ts, y = posOpinionShare, group = runNumber, color=as.
    factor(pMedia)))
```

B.3 Compare Strategies

Listing 26: Compare.R

```

# Load Data ———————————————————————————————————————————————
rm(list=ls(all=TRUE))
setwd("~/git/mss/MSS/R")
library("ggplot2")
library("plyr")
library("RColorBrewer")
library("xtable")
source("defineBeamerPlot.R")


#Stupid Stupid
setwd("~/git/mss/MSS/D150Out/smallStupidStuipd/1481908670856/")
source("~/git/mss/MSS/R/generateStandardPlots.R")
StupidStupid = aggregated
StupidStupid$Game="StupidStupid"

#Swing Stupid
setwd("~/git/mss/MSS/D150Out/smallSwingStupid/1481926279694/")
source("~/git/mss/MSS/R/generateStandardPlots.R")
SwingStupid = aggregated
SwingStupid$Game="SwingStupid"

#Prom Stupid
setwd("~/git/mss/MSS/D150Out/smallPromStupid/1481955685931/")
source("~/git/mss/MSS/R/generateStandardPlots.R")
PromStupid = aggregated
PromStupid$Game="PromStupid"

setwd("~/git/mss/MSS/R/tables/")
# Merge ———————————————————————————————————————————————————
Merged=rbind(StupidStupid, SwingStupid, PromStupid)




# Table ———————————————————————————————————————————————————
#Identifiers
tabelIdentA = 0.7*100000+0.46*1000+c(0.25,0.5,0.75)*10
tabelIdentB = 0.7*100000+0.48*1000+c(0.25,0.5,0.75)*10
tabelIdentC = 0.7*100000+0.5*1000+c(0.25,0.5,0.75)*10
allIdent = c(tabelIdentA, tabelIdentB, tabelIdentC)

TableData = subset(Merged, (Merged$combinationIdentifier%in%allIdent))
TabelRduced = TableData[,c()]

```

```
rawTable = ddply(TableData, ~combinationIdentifier, summarise, p0=mean(p), pm
    =mean(s))

TabelDataSort = TableData[(order(TableData$combinationIdentifier)),]

#Means
rawTable$MeanRandom=0
rawTable[(order(rawTable$combinationIdentifier)),]$MeanRandom = TabelDataSort
    [(TabelDataSort$Game=="StupidStupid"),]$pEnd

rawTable$MeanProtect=0
rawTable[(order(rawTable$combinationIdentifier)),]$MeanProtect =
    TabelDataSort[(TabelDataSort$Game=="SwingStupid"),]$pEnd


rawTable$MeanConvincible=0
rawTable[(order(rawTable$combinationIdentifier)),]$MeanConvincible =
    TabelDataSort[(TabelDataSort$Game=="PromStupid"),]$pEnd

#SD

rawTable$SDRandom=0
rawTable[(order(rawTable$combinationIdentifier)),]$SDRandom = TabelDataSort[(
    TabelDataSort$Game=="StupidStupid"),]$sdPEnd

rawTable$SDProtect=0
rawTable[(order(rawTable$combinationIdentifier)),]$SDProtect = TabelDataSort
    [(TabelDataSort$Game=="SwingStupid"),]$sdPEnd


rawTable$SDConvincible=0
rawTable[(order(rawTable$combinationIdentifier)),]$SDConvincible =
    TabelDataSort[(TabelDataSort$Game=="PromStupid"),]$sdPEnd






hlines=c(-1,0,3,6,9)
MeanTable = xtable(rawTable[,c(2:9)], caption="Mean result for proponents.
    Oponents play stupid. \\rot{10 runs per cell}", digits = 2)
align(MeanTable) <- "|l|l|l|ccc|ccc"
print.xtable(x=MeanTable, hline.after = hlines, include.rownames = FALSE,
    vlines.after=vlines, caption.placement = "top")#, file = "meanTable.txt")



# Winner Plot ------------------------------------------------------------
Winner = ddply(Merged, ~combinationIdentifier, summarize, winnerIs=Game[which
    .max(pWin)], p0=mean(p), pm=mean(s), MaxpWin = max(pWin), sd=sd(pWin),
    diff=max(pWin)-min(pWin))
Winner$winnerIs=ifelse(Winner$diff < 0.0001, "non", Winner$winnerIs)

ggplot(Winner, aes(x=p0, y=pm))+
  geom_tile(aes(fill=as.factor(winnerIs)))+
  #scale_fill_discrete(name=expression(p[final]))+
  scale_fill_manual(name="Most Victories:", drop=FALSE, values=c("SwingStupid
      "="red", "StupidStupid"="darkgreen", "PromStupid"="yellow", "non"="gray
      "),
                      labels=c("SwingStupid"="Protect", "StupidStupid"="Random"
                          , "PromStupid"="Convincible", "non"="Even"))+
  beamerPlot+
  xlab(expression(p[0]))+
  ylab(expression(p[M]))
ggsave(file="WinnerPlot.pdf", width=16, height=10, dpi=100)





# Chance To Win P0 vs S ------------------------------------------------
#Order
Merged$Game=factor(Merged$Game,
        levels = c("StupidStupid", "PromStupid", "SwingStupid"),
        labels = c("Random", "Convincible", "Protect"))

ggplot(Merged, aes(x=p, y=s))+
  geom_tile(aes(fill=pWin))+
  scale_fill_gradientn(name="Share of \nRuns Won\nby Proponents\n" ,colours=
      brewer.pal(9,"RdYlGn"))+
  beamerPlot+
  xlab(expression(p[0]))+
  ylab(expression(p[M]))+
  theme(axis.text.x = element_text(angle = 90, hjust = 1))+
  #facet_grid(Game~.)+
  facet_grid(~Game)+
  theme(panel.spacing.x=unit(0.5, "lines"),panel.spacing.y=unit(1, "lines"))+
  theme(strip.text.x = element_text(size = 18))+
  theme(strip.text.y = element_text(size = 18))
ggsave(file="PWinPvsS.pdf", width=16, height=10, dpi=100)
```

B.4 Generate Parameter Grid

Listing 27: genGrid.R

```r

# Grid all paramters combinations once ----------------------------------------
p = c(0.25,0.5,0.75)
t = c(1:5)*0.2
s = c(1:5)*0.2

grid = expand.grid(p,t,s)

names(grid)[1]="p"
names(grid)[2]="t"
names(grid)[3]="s"


write.csv(grid, "../input/grid.csv")



# fixed combination multiple times ---------------------------------------------
numbers = c(1:150)

sets = data.frame(Number = numbers)
sets$p=0.25
sets$t=0.4
sets$s=0.6
sets$Number=NULL
write.csv(sets, "../input/grid.csv")



# p equals s -------------------------------------------------------------------
p = c(1:20)*0.05
t = c(1:20)*0.05
s = c(1:20) # >> gives 20 points per combinations s will be set to p later

grid = expand.grid(p,t,s)

names(grid)[1]="p"
names(grid)[2]="t"
names(grid)[3]="s"

grid$s = grid$p
length(grid$p)

write.csv(grid, "../input/peqsgrid20.csv")


```

```

# p vs s; tau 0.7 ---------------------------------------------------------------
p = c(1:10)*0.05
t = 0.7
s = c(1:20)*0.05 # >> gives 20 points per combinations s will be set to p
    later
n=c(1:10)

grid = expand.grid(p,t,s,n)

names(grid)[1]="p"
names(grid)[2]="t"
names(grid)[3]="s"

length(grid$p)

write.csv(grid, "../input/tfixed1.csv")



# Detailed Grid ------------------------------------------------------------------



# p vs s; tau 0.7 ---------------------------------------------------------------
p = c(20:25)*0.02
t = 0.7
s = c(1:20)*0.05
n=c(1:100)

grid = expand.grid(p,t,s,n)

names(grid)[1]="p"
names(grid)[2]="t"
names(grid)[3]="s"

length(grid$p)

write.csv(grid, "../input/detail.csv")
```

B.5 Evaluate Temporal Dynamics

Listing 28: recordEval.R

```r
rm(list=ls(all=TRUE))
library("ggplot2")

record <- read.csv(file="../output/1481580003092/mruns.csv",sep=",",head=TRUE
    )


ggplot(record)+
  geom_line(aes(x=ts, y = posOpinionShare, group = runNumber, color=as.factor
      (theta)))



ggplot(record)+
  geom_line(aes(x=ts, y = posOpinionShare, group = runNumber, color=as.factor
      (pMedia)))


ggplot(record[(record$pMedia==0.2),])+
  geom_line(aes(x=ts, y = posOpinionShare, group = runNumber, color=as.factor
      (theta)))

ggplot(record[(record$ts>30),])+
  geom_line(aes(x=ts, y = posOpinionShare, group = runNumber, color=as.factor
      (theta)))


ggplot(record[(record$theta==0.4 & record$pMedia>0.5),])+
  geom_line(aes(x=ts, y = posOpinionShare, group = runNumber, color=as.factor
      (pMedia)))+
  xlim(c(1000, 1010))


ggplot(record)+
  geom_smooth(aes(x=ts, y = posOpinionShare, color=as.factor(theta)))

max(record[(record$ts>30),]$posOpinionShare)
max(record[(record$posOpinionShare>0),]$ts)

hist(record[(record$ts==18),]$posOpinionShare)
```

B.6 Evaluate Input Signals (for diagnosis)

Listing 29: evalinputSignal.R

```
signalRecord <- read.csv(file="mruns40.csv",sep=",",head=TRUE) #40
#signalRecordB <- read.csv(file="mruns0.csv",sep=",",head=TRUE)

signalRecord$inputBinary = ifelse(signalRecord$inputSignal == 1,1,0)



mean(signalRecord$inputBinary)
table(signalRecord$inputSignal, signalRecord$previousOpinion)

prop.table(table(signalRecord$ts,signalRecord$inputBinary), margin = 1)
prop.table(table(signalRecord$ts,signalRecord$previousOpinion), margin = 1)

ggplot()+
  geom_density(aes(x=signalRecord$tauPos, group=signalRecord$ts, color=
      signalRecord$ts))


duplicated(signalRecordB$AgentID)
agent = unique(signalRecordB$AgentID[duplicated(signalRecordB$AgentID)])[5]
table(signalRecordB$inputSignal, signalRecordB$previousOpinion)
signalRecord[(signalRecord$AgentID==agent),]
signalRecordB[(signalRecordB$AgentID==agent),]
```

B.7 Plot Design Configuration

Listing 30: defineBeamerPlot.R

```
library(ggplot2)
library("grid", lib.loc="/Library/Frameworks/R.framework/Versions/3.2/
    Resources/library")
beamerPlot =
  theme_bw()+
  #theme_light()+
  #theme_minimal()+
  theme(axis.text = element_text(size = rel(3)))+
  theme(axis.title = element_text(size = rel(3)))+
  theme(legend.text=element_text(size=25), legend.title=element_text(size=25)
      )+
  theme(legend.key.size = unit(2, 'lines'))+
  theme(axis.title.y=element_text(vjust=2))#+
  #theme(axis.title.x=element_text(vjust=-1))
  #theme(axis.title.y=element_text(margin=margin(0,20,0,0)))

beamerWidth=1



# Multiplot ———————————————————————————————————————————————
multiplot <- function(..., plotlist=NULL, file, cols=1, layout=NULL) {
  library(grid)

  # Make a list from the ... arguments and plotlist
  plots <- c(list(...), plotlist)

  numPlots = length(plots)

  # If layout is NULL, then use 'cols' to determine layout
  if (is.null(layout)) {
    # Make the panel
    # ncol: Number of columns of plots
    # nrow: Number of rows needed, calculated from # of cols
    layout <- matrix(seq(1, cols * ceiling(numPlots/cols)),
                     ncol = cols, nrow = ceiling(numPlots/cols))
  }

  if (numPlots==1) {
    print(plots[[1]])

  } else {
    # Set up the page
    grid.newpage()
    pushViewport(viewport(layout = grid.layout(nrow(layout), ncol(layout))))

    # Make each plot, in the correct location
    for (i in 1:numPlots) {
      # Get the i,j matrix positions of the regions that contain this subplot
      matchidx <- as.data.frame(which(layout == i, arr.ind = TRUE))

      print(plots[[i]], vp = viewport(layout.pos.row = matchidx$row,
                                      layout.pos.col = matchidx$col))
    }
  }
}

# ----------------------------------------------------------------------
```