

(DE)STABILIZING EFFECTS OF INNOVATION DIFFUSION

AN AGENT-BASED EXPLORATION OF DIFFUSION FORCES ON COMPLEX SYSTEMS

January, 2013

Cornelis Eikelboom 1370073

Bert van Meeuwen 1359991

SPM 9555- Agent-Based Modeling of Complex Adaptive Systems

PREFACE

This report is made for the course 'Agent-Based Modelling of Complex Adaptive Systems – Advanced', or SPM9555, at Delft University of Technology. As assignment for this course, the model was developed for Joolie Kasmire, as PhD researchers at the Faculty of Technology, Policy and Management.

Before starting this course, neither of us did have any experience with Agent Based Modelling, programming (in any code whatsoever), data compression, and large scale data processing. Hence, we would like to thank Igor Nikolic, Joolie Kasmire, Andrew Bollinger, Chris Davis, Jan Jaap Treurniet and Kasper Kisjes very much for providing a (very, very) steep learning curve in Agent-Based Modeling, NetLogo, R, various R packages, PHP, data compression algorithms and diffusion theory.

January 2013,

Bert van Meeuwen & Cornelis Eikelboom

TABLE OF CONTENTS

PREFACE **2**

TABLE OF CONTENTS **3**

INTRODUCTION **5**

PROBLEM FORMULATION AND ACTOR IDENTIFICATION **6**

 WHAT IS THE PROBLEM? 6

 WHOSE PROBLEM IS IT? 7

 WHAT IS OUR ROLE..... 7

SYSTEM IDENTIFICATION AND DECOMPOSITION **7**

 ANALOGY 7

 AGENTS, PROPERTIES AND (INTER)ACTIONS 7

 ACTIONS PERFORMED BY THE AGENT TO AFFECT THEIR OWN STATE..... 8

 ENVIRONMENT 8

CONCEPT FORMALIZATION **9**

 NETLOGO LANGUAGE PRIMITIVES 9

 CONCEPT FORMALIZATION OF EV INNOVATION SYSTEM..... 9

MODEL FORMALIZATION **11**

 NARRATIVE 11

 PSEUDO-CODE..... 16

SOFTWARE IMPLEMENTATION..... **21**

 NETLOGO CODE 21

 INCONSISTENCIES BETWEEN MODEL AND NARRATIVE/PSEUDO CODE 25

MODEL VERIFICATION **26**

 RECORDING AND TRACKING AGENT BEHAVIOR 26

 SINGLE-AGENT VERIFICATION 27

 INTERACTION TESTING IN A MINIMAL ENVIRONMENT 30

 MULTI-AGENT TESTING 32

EXPERIMENTATION **37**

 EXPERIMENTAL DESIGN: MODEL HYPOTHESIS & RUN TIME 37

 EXPERIMENTAL DESIGN: PARAMETER SPACE 37

 EXPERIMENTAL DESIGN: REPLICATIONS 38

 EXPERIMENTAL SETUP: RANDOMNESS AND RUN TIME..... 38

 EXPERIMENTAL SETUP: METRICS 38

 EXPERIMENT EXECUTION: CHECKING DATA CONSISTENCY 38

FINDING AND USING A ‘METAMETRIC’ **39**

 WHAT IS STABILITY? 39

 INTRODUCING COMPRESSION AS STABILITY METRIC CONCEPT 39

 INTRODUCING DEFLATE AS STABILITY METRIC 40

 HOW TO DO COMPRESSION ANALYSIS ON NETLOGO OUTPUT 40

 PHP CODE FOR DETERMINING KOLMOGOROV COMPLEXITY..... 41

DATA ANALYSIS	43
INITIAL OUTPUT	43
KOLMOGOROV COMPLEXITY.....	45
MODEL VALIDATION	52
HISTORIC REPLAY.....	52
LITERATURE VALIDATION.....	52
FACE VALIDATIONS THROUGH EXPERT CONSULTATION.....	52
MODEL REPLICATION	52
MODEL USE	53
CONCLUSIONS	53
RECOMMENDATIONS	53
DISCUSSION ON MODEL LIMITATIONS.....	54
DISCUSSION ON THE MODELLING CYCLE EXPERIENCE	55
REFERENCES	56
APPENDIX A: FULL MODEL CODE USED FOR MODEL VERIFICATION	58
APPENDIX B: REWRITTEN CODE FOR RECORDING/TRACKING AGENT BEHAVIOR	66
APPENDIX C: COMPARISON ON DEGREE ON COMPRESSION OF # EV (HISTOGRAMS)	67
APPENDIX D: COMPARISON ON DEGREE ON COMPRESSION OF # EV (DENSITY CURVE)	69
APPENDIX E: COMPARISON ON OVERALL COMPRESSION SIZES (HISTOGRAMS)	71
APPENDIX F: COMPARISON ON OVERALL COMPRESSION SIZES (DENSITY CURVES)	73
APPENDIX G: REWORK DATA FROM INITIAL NETLOGO OUTPUT	75
APPENDIX H: R CODE FOR THE ANALYSIS OF THE INITIAL OUTPUT	76
APPENDIX I: R CODE FOR THE ANALYSIS OF KOLMOGOROV COMPLEXITY	78

INTRODUCTION

All around us, we constantly experience innovations and their diffusion over the environment. In literature, it is widely recognized across research fields that innovations often follow an S-shaped behavior in terms of their diffusion over time (Mahajan, Muller, & Srivastava, 1990; Berwick, 2003; Burke & Schumann, 1928). Question is whether diffusions are destabilizing or stabilizing forces. There is support for both, mostly based on philosophical reasoning.

In the case for destabilizing, transitions are seen as periods of change between two stable states with little recognizable change. The change is little recognizable as any changes are small scale, localized and short lived. Transitions are less stable than the states on either side because there is quite a lot of visible and obvious change, that is large scale, generalized and long lived. So the diffusion of the new technology first destabilizes the old technology in a period of struggle, and then stabilizes again. This perspective is traditionally found in the field of thermodynamics, but also recognized in the fields of transition management and technology diffusion. Hence, we will refer to the 'general perspective' when referring to this perspective.

Others, based on the work of Prigogine (1972) suggest that diffusions are actually stabilizing forces. States before and after a transition may be considered unstable as change is hard to detect as the change is not uniform. The diffusion makes change easier to see as change is going in a same direction. After the diffusion part, change is less uniform again. This perspective thus uses the uniformity of change instead of the amount of change. The more uniform the change, the more stable and predictable the system. As Prigogine is the 'founder' of this (minority) perspective, we will refer to the 'Prigogine perspective' when referring to this perspective.

Note that in our way of referring to 'general perspective' or 'Prigogine perspective' we drastically simplify the viewpoints of science. Prigogine was mostly making a case for how else stability can be seen. However, for convenience we will refer to the perspectives in this way.

In literature, it seems like the different perspectives are treated as if they include each other (Chen, 2004; Smith, Stirling, & Berkhout, 2005). However, there seems to be no solid proof that the definitions really do exclude each other. This report uses an agent-based modeling (ABM) approach to explore whether the two perspectives really exclude each other. We will base the ABM approach on the ten steps as they are described by Van Dam, Nikolic and Lukszo (2012).

PROBLEM FORMULATION AND ACTOR IDENTIFICATION

In this section, we will elaborate on a number of questions that relate to the problem at hand and the role of actors (including ourselves) in this project. In respect to the problem at hand, we will eventually form an initial hypothesis that answers the identified lack of insight. In respect to the actor field surrounding this project and the problem, we will identify our role in the actor system to identify our potential bias.

WHAT IS THE PROBLEM?

WHAT IS THE EXACT LACK OF INSIGHT THAT WE ARE ADDRESSING?

As was already elaborated upon in the introduction, it is unclear how the different perspectives on stability (i.e. the general and that of Prigogine) relate to each other.

WHAT IS THE OBSERVED EMERGENT PATTERN OF INTEREST TO US?

The observed emergent behavior is a s-shaped curve on the system level when an 'innovation' is introduced to a closed system. Note the system level element in the previous statement. No physicist expects a closed system of gas that has fully identical particles, but it is considered unimportant that some particles are slightly more energetic, warmer etc. than others if the average remains constant. Similarly, in a closed socio-technical system, it is not expected that everybody behaves exactly identical. We tend to just use descriptive statistics on the whole system.

IS THERE A DESIRED EMERGENT PATTERN?

In this study, the desired emergent pattern should be a reconstruction of the observed emergent pattern (i.e. the s-curve) when measured with the sort of measurements that diffusion studies traditionally use. The observed emergent pattern is based on descriptive statistics used in studies on diffusions of innovations and thermodynamics diffusion.

Descriptive statistics used in these studies mostly include shares of a characteristic in the system. In the case of thermodynamics, think for example of two molecules, A and B, whereby A is transformed into B. However, molecule B is a catalyst for the transformation from A to B. When a single molecule B is introduced to the system, there are many molecules of A, but only a limited number of catalysts. As the number of B grows, the transformation rate increases up until where there are many molecules of B and a limited number of A molecules. In the field of transition management, one could think of the spreading of smartphones over society for example. A lot of work in this field (that also shows the used metrics) has been done by Everett Rogers (1995).

WHAT IS THE INITIAL HYPOTHESIS?

The general perspective and the Prigogine perspective exclude each other in identification of stability.

WHOSE PROBLEM IS IT?

As this problem is relatively philosophical and abstract, there is no specific problem owner. The problem owner is any person, group or community that wants to achieve stability in some sense. They should be aware of the differences and similarities in the different stability perspectives in order to get to a solid and worthy investment on the stability they want to reach.

WHAT IS OUR ROLE

Based on the initial hypothesis, we will search for evidence that the two perspectives on stability exclude each other. However, the large but implicit assumption here is that the perspective either exclude or allow each other. The bias this brings is that we will mostly likely not recognize spectrums (i.e. grey areas) between the two perspectives.

SYSTEM IDENTIFICATION AND DECOMPOSITION

In this section, we identify the agents in our system, with its properties, its actions, and its interaction with its environment. Finally, we also describe the environment of the elements. As the problem we model is quite abstract, we first come up with an analogy of a diffusion system. Note that this text mostly describes the final output, but that the output is the result of a number of iterations over the paragraphs discussed here. An overview of the inventory structure is found in figure 2.1.

ANALOGY

For finding a workable analogy, some base points of stability theory have to be considered. Most important point here is that we are studying complex systems, whereby both stability perspectives define stability on a system level, while recognizing that the state of individual elements may vary. Secondly, we study an innovation spreading through a system. So, elements in the system share some type of link, opposed to just varying randomly.

The analogy chosen in this report is that of the introduction of electric vehicles (EVs) in a group of car owners that share social relations with each other and base their decision for a car on these social relationships. In the following paragraph, we will further elaborate and decompose this system.

AGENTS, PROPERTIES AND (INTER)ACTIONS

In the model, we will only use a single type of agents, namely car owners.

A car owners has a set of properties. First of all, a car owner has a specific car type (either EV, fossil fuel or diesel) of a certain age and lifespan. Secondly, each car owner has a certain attitude to innovation (Based on diffusion theory), that categorizes him in one of five categories of adopters (based on Rogers, 1995). Thirdly, depending on the adopter category, each car owner has a timespan for which it remembers, i.e. a memory for, the division of car types in his environment that may influence his own decision on buying a car. Also, each car owner keeps track of his own history of car types. Fourth, also based on the attitude to innovation, each car owner has a threshold that it needs to exceed for buying a different car type.

When the age of the car of a car owner reaches the lifespan of the car, the car owner has to buy a new car. The choice for a car technology is based on the car owners own car history as well as the cars types owned in the social network of the car owner. In the following paragraph, we will go in further detail on the actions of the agents that affect their own state.

ACTIONS PERFORMED BY THE AGENT TO AFFECT THEIR OWN STATE

Over time, each car owner remember the type of car he owns and sees in his own network for a certain period of time. Based on this observation, a car owner determines what type of car he will buy. This choice is also based on his change threshold and its adopter category. Any car owner has to see a car type so often that it exceeds his threshold to potentially buy the car type. The way the car owner chooses on what car type (from the car types that exceeds the threshold) will be bought, is depending on the adopter category in which the car owner falls.

Innovative minded people are eager to try 'something different'. Hence, these people preferably switch to a car type they see the least, as long as that car type still exceeds their threshold. On the other side, that are the really conservative people that have a high threshold. In the case the threshold is exceeded they will choose for the technology they see the most in their environment. We define also a middle category of car owners, that don't have a preference on car type, as long as that car type exceeds their preference.

ENVIRONMENT

In line with diffusion theory that generally describe diffusions as if they occur in closed systems, the action of the environment is limited to the introduction of electric vehicles. Innovations are then treated as a temporary opening of the system boundary. Hence, the only action of the environment will be to introduce EVs in the system after a period of time. Furthermore, the environment also comprises the initial division of diesel and petrol car owners as well as the average number of social links a car owner has. Finally, the environment will keep count of how many car owners switch to what car type over time.

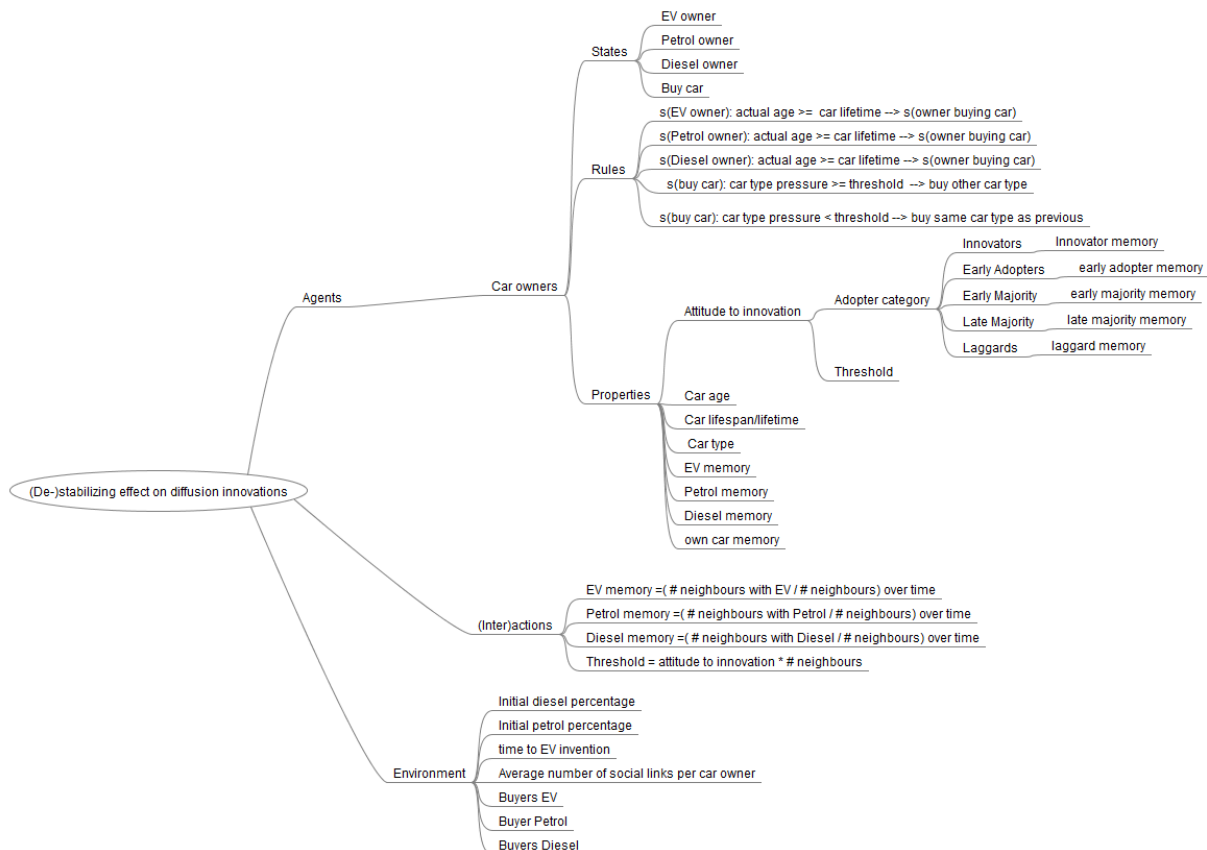


Figure 1: Structure of system inventory

CONCEPT FORMALIZATION

In this section, we will formalize the concept introduced in the previous section. In order to do this, we will use language primitives from Netlogo. In the first paragraph, we will explore the language primitives used in Netlogo. In the second paragraph, we show the formalization of the EV innovation system.

NETLOGO LANGUAGE PRIMITIVES

Netlogo in itself knows 6 types of primitives that we will shortly mention here. Netlogo can also make use of certain extensions. However, no extensions are needed for the purpose of this study. Netlogo uses the following primitives:

- Numbers: both integers as floating points
- Stings: a set of characters
- Booleans: true/false values in logic
- Turtles/patches/nodes/edges and breeds of them: (type of) agents
- Agent/patch sets: collections of agents and/or patches
- Lists: may contain any of the primitives

CONCEPT FORMALIZATION OF EV INNOVATION SYSTEM

In this section we elaborate

The car owners have:

- Electric car: Boolean
- Petrol car: Boolean
- Diesel car: Boolean
- Car age: integer ≥ 0 and \leq car lifetime
- Car lifetime: floating point > 0
- Attitude to innovation: floating point ≥ 0 and ≤ 1
- Change threshold ≥ 0 and \leq number of neighbors
- Innovator: Boolean
- Innovator memory: integer ≥ 0
- Early adopter: Boolean
- Early adopter memory: integer ≥ 0
- Early majority: Boolean
- Early majority memory: integer ≥ 0
- Late majority: Boolean
- Late majority memory: integer ≥ 0
- Laggard: Boolean
- laggard memory: integer ≥ 0
- EV list: list of integers ≥ 0 and \leq number of neighbors
- Petrol list: list of integers ≥ 0 and \leq number of neighbors
- Diesel list: list of integers ≥ 0 and \leq number of neighbors
- Car history list: list of strings ("EV", "PETROL" or "DIESEL")

The environment has:

- Initial diesel percentage: integer ≥ 0
- Time to EV invention: integer ≥ 0
- Average number of social links ≥ 0
- EV buyers: Integer ≥ 0
- Petrol buyers: Integer ≥ 0
- Diesel buyers: Integer ≥ 0

MODEL FORMALIZATION

After identification of what and who is in the model, it is now time to define who does what when. For that reason, this section describes the created model narrative (in text form), the most important modeling considerations, and the translation into pseudo-code. Again, this section shows the narrative and pseudo-code after a number of iterations.

NARRATIVE

The narrative consists of two parts. The first part contains the procedures for setting up the model. The second part contains the running procedures.

PART 1: PROCEDURES FOR SETTING UP THE MODEL

- Set background color white
- Setup car owners
 - Create number of car owners
 - Number of car owners is an input variable
 - Give each car owner a random location on the map
- Give car owners initial cars
 - Give each car owner a petrol car
 - petrol car to true, other car types to false
 - give each petrol car owner a red color
 - define car lifetime
 - car lifetime is drawn from a normal distribution with average (petrol car life * 10) and standard deviation of 20 ticks
 - average petrol car life is an input variable
 - each year corresponds to 10 ticks
 - car age to zero or random smaller than car lifetime
 - Way to set car age is an input variable
 - Assign diesel car owners
 - # diesel car owners = number of car owners * initial diesel percentage / 100
 - Initial diesel percentage is an input variable [0-100]
 - random petrol car owner to diesel car owner
 - diesel car to true, other car types to false
 - Give each diesel car owner a blue color
 - car lifetime (draw from normal distribution with average (diesel car life * 10) and standard deviation of 20 ticks
 - Average diesel car life is an input variable
 - Each year corresponds to 10 ticks
 - Repeat previous step '# diesel car owners' times
- Place car owners in a social network
 - Number of links to create = (average links per car owner * number of car owners) / 2
 - Average links per car owners is input variable
 - Number of car owners is input variable
 - Divide by two as each link connects to two car owners

- Make sure everybody has at least 1 link
- Create a link
 - Pick a random car owner
 - Choose the closest car owner that I don't have a link with
 - Create link
 - Make link color grey
- Repeat previous step as long as total number of links \leq number of links to create
- Setup innovation preferences of each car owner
 - define attitude to innovation as a number drawn from a $N(0.5;0.16)$ distribution
 - define change threshold = round (attitude to innovation * total number of neighbors)
 - If change threshold < 0 , make change threshold 0
 - If change threshold $>$ total number neighbors, make change threshold total number of neighbors
 - define adopter category
 - If attitude to innovation ≥ 0.84 , laggard to true, other adopter categories false
 - If attitude to innovation ≥ 0.50 and < 0.84 , late majority to true, other adopter categories false
 - If attitude to innovation ≥ 0.16 and < 0.50 , early majority to true, other adopter categories false
 - If attitude to innovation ≥ 0.025 and < 0.16 , early adopter to true, other adopter categories false
 - If attitude to innovation $< .025$, innovator to true, other adopter categories false
- Setup lists for each car owner
 - define ev list (0)
 - If car owner has petrol car, define petrol list (1 + number of neighbors with petrol car),
else, define petrol list (number of neighbors with petrol car)
 - If car owner has diesel car, define petrol list (1 + number of neighbors with diesel car),
else, define diesel list (number of neighbors with diesel car)
 - If car owner has petrol car, define car history list ("petrol"),
Else, if car owner has diesel car, define car history list ("diesel"),
Else, report error in car history list
- Reset run time

PART 2: RUNNING PROCEDURES

- Make car older
 - Car age = car age + 1
- Update memory for each turtle
 - If car owner has electric car, add to ev list (1 + number of neighbors with electric car),
Else, add to ev list (number of neighbors with electric car)
 - If car owner has petrol car, add to petrol list (1 + number of neighbors with petrol car),
else, add to petrol list (number of neighbors with petrol car)
 - If car owner has diesel car, set petrol list (1 + number of neighbors with diesel car),
else, add to diesel list (number of neighbors with diesel car)
 - If length of ev list is longer than (10 * memory of adopter category), remove oldest item from ev list
 - If length of petrol list is longer than (10 * memory of adopter category), remove oldest item from petrol list

- If length of diesel list is longer than ($10 * \text{memory of adopter category}$), remove oldest item from diesel list
- For each car owner, If car age \geq car lifetime, buy a new car
 - If run time $<$ ($\text{years to ev invention} * 10$), choose between diesel and petrol
 - If diesel car owner and the average of the petrol list exceeds the threshold, choose petrol technology, else, choose diesel technology.
 - If petrol car owner and the average of the diesel list exceeds the threshold, choose diesel technology, else, choose petrol technology
 - Else, choose between ev, petrol and diesel
 - If car owner has an electric car
 - If both the average of the diesel list and petrol list exceed (or are equal to) the change threshold
 - If adopter category is innovator or early adopter
 - If average of diesel list is larger than average of petrol list, become a petrol owner *[described later]*
 - If average of diesel list is shorter than average of petrol list, become a diesel owner *[described later]*
 - If average of diesel list is equal to average of petrol list, choose random between petrol car and diesel car
 - If adopter category is early majority or late majority
 - Choose random between petrol car and diesel car
 - If adopter category is laggards
 - If average of diesel list is larger than average of petrol list, become a diesel owner
 - If average of diesel list is shorter than average of petrol list, become a petrol owner
 - If average of diesel list is equal to average of petrol list, choose random between petrol car and diesel car
 - If average of diesel list is equal to or bigger than the threshold, while the average of the petrol list is smaller than the threshold, become diesel owner.
 - If average of diesel list is smaller than the threshold, while the average of the petrol list is equal or bigger than the threshold, become petrol owner
 - If the average of both the petrol and diesel list is smaller than the threshold, remain an ev owner *[described later]*
 - If car owner has a diesel car
 - If both the average of the ev list and petrol list exceed (or are equal to) the change threshold
 - If adopter category is innovator or early adopter
 - If average of ev list is larger than average of petrol list, become a petrol owner
 - If average of ev list is shorter than average of petrol list, become a ev owner
 - If average of ev list is equal to average of petrol list, choose random between petrol car and ev car
 - If adopter category is early majority or late majority
 - Choose random between petrol car and ev car

- If adopter category is laggards
 - If average of ev list is larger than average of petrol list, become a ev owner
 - If average of ev list is shorter than average of petrol list, become a petrol owner
 - If average of ev list is equal to average of petrol list, choose random between petrol car and ev car
- If average of ev list is equal to or bigger than the threshold, while the average of the petrol list is smaller than the threshold, become ev owner.
- If average of ev list is smaller than the threshold, while the average of the petrol list is equal or bigger than the threshold, become petrol owner
- If the average of both the petrol and ev list is smaller than the threshold, remain a diesel owner
- If car owner has an petrol car
 - If both the average of the ev list and diesel list exceed (or are equal to) the change threshold
 - If adopter category is innovator or early adopter
 - If average of ev list is larger than average of diesel list, become a diesel owner
 - If average of ev list is shorter than average of diesel list, become a ev owner
 - If average of ev list is equal to average of diesel list, choose random between diesel car and ev car
 - If adopter category is early majority or late majority
 - Choose random between diesel car and ev car
 - If adopter category is laggards
 - If average of ev list is larger than average of diesel list, become a ev owner
 - If average of ev list is shorter than average of diesel list, become a diesel owner
 - If average of ev list is equal to average of diesel list, choose random between diesel car and ev car
 - If average of ev list is equal to or bigger than the threshold, while the average of the diesel list is smaller than the threshold, become ev owner.
 - If average of ev list is smaller than the threshold, while the average of the diesel list is equal or bigger than the threshold, become diesel owner
 - If the average of both the diesel and ev list is smaller than the threshold, remain a petrol owner
- For becoming an ev owner
 - electric car to true, other car types to false
 - car owner color to green
 - car age to 0
 - car lifetime random from a normal distribution with the average of 10 * average ev car life and a standard deviation of 20
 - Average ev car life is an input variable

- For becoming a diesel owner
 - diesel car to true, other car types to false
 - car owner color to blue
 - car age to 0
 - car lifetime random from a normal distribution with the average of 10 * average diesel car life and a standard deviation of 20
 - Average diesel car life is an input variable
- For becoming a petrol owner
 - petrol car to true, other car types to false
 - car owner color to red
 - car age to 0
 - car lifetime random from a normal distribution with the average of 10 * average petrol car life and a standard deviation of 20
 - Average petrol car life is an input variable
- Update car history list for each turtle
 - If car owner has electric car, add EV to car history list
 - If car owner has diesel car, add DIESEL to car history list
 - If car owner has petrol car, add PETROL to car history list
- End of time step

As the procedure of buying a car might look quite complex, figure 2 shows a schematic representation of the working of this procedure. Note this is only a part of the total running procedure.

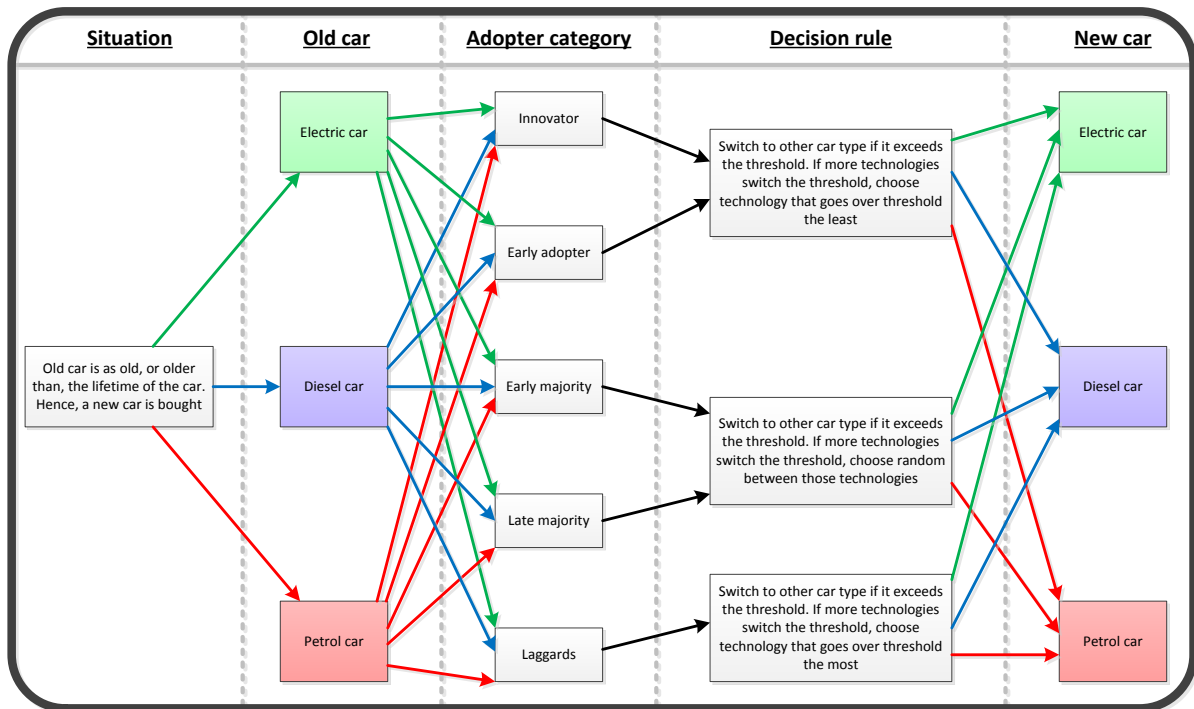


figure 2: schematic overview of car buying procedure

PSEUDO-CODE

In the pseudo code, we describe the exact flow of the algorithms we are about to program. This is very useful, as it is the last moment before programming. Hence, it is now relatively easy to make adaptations. Furthermore, this step gives us great insight in how each car owners makes its decision. Here, we first define variables each car owner owns. Subsequently we provide pseudo code for setting up the model. We will end this section with the pseudo code for the running procedures

PART 1: VARIABLES THAT TURTLES OWN

All turtles own

- Electric-car?
- Petrol-car?
- Diesel-car?
- Innovator?
- Early-adopter?
- Early-majority?
- Late-majority?
- Laggards?
- Car-age
- Car-lifetime
- Attitude-to-innovation
- Change-threshold
- Ev-list
- Petrol-list
- Diesel-list
- Car-history-list

PART 2: PSEUDO CODE FOR SETTING UP THE MODEL

To setup

- Clear-all
- Ask patches (set color white)
- setup nodes
 - Set default shape turtles "circle"
 - Create number-of-car-owners
 - Setxy random x-cor random y-cor
- setup initial car
 - Ask turtles set size 2
 - Ask turtles become-PETROwner
 - Ask n-of $((\text{number-of-car-owners} * \text{initial-diesel-percentage}) / 100)$ turtles with petrol-car become-DIESEowner
 - ask turtles
 - if initial-car-age = 0 (set car-age 0)
 - if initial-car-age = "random" (set car-age random car-lifetime)


```
setup network
  let num-links (average-links-per-car-owner * number-of-car-owners) / 2
  ask turtles (if count link with neighbors <= 0, create link with smallest of (other turtles
    distance to myself)
  while count links <= num-links
    (ask one of turtles (pick choice of
      (other turtles with no link to neighbor and myself)
      (based on distance from turtle to myself))
    If choice is not nobody, create-link-with choice)
  Ask links set color grey
Setup innovation preferences
  Ask turtles
    Set attitude-to-innovation to random-normal 0.5 0.16
    Set change-threshold to rounded (attitude-to-innovation * count link-neighbors))
    If change-threshold < 0, set change-threshold to 0
    If change-threshold > countlink-neighbors, set change-threshold count link-neighbors
    If attitude-to-innovation >= 0.84 , set laggards? to True
    If attitude-to-innovation < 0.84 , set late-majority? to True
    If attitude-to-innovation < 0.50 , set early-majority? to True and set late-majorityfalse
    If attitude-to-innovation < 0.16 , set early-adopter? True and set early-majority false
    If attitude-to-innovation < 0.0.25 , set innovator? to True and set early-majority false
Setup lists
  Ask turtles
    set ev-list to (list 0)
    ifelse petrol-car? (set petro- list (list (1+ count link-neighbors with petrol-car?)))
      (set petrol-list (list count link neighbors with petrol-car?))
    ifelse diesel-car? (set diesel-list (list (1+ count link-neighbors with diesel-car?)))
      (set diesel-list (list count link-neighbors with diesel-car?))
    ifelse petrol-car? (set car-history list (list "PETROL"))
      (ifelse diesel-car? (set car-history-list (list "DIESEL))
        (show "ERROR in car history list))
Reset ticks
```

PART 3: PSEUDO CODE FOR RUNNING PROCEDURES

To go

```
Ask turtles set car-age car-age + 1
Ask turtles update memory
  ifelse electric-car? (add (1 + count link-neighbors with electric-car?) to end of list)
    (add (count link-neighbors with electric-car?) to end of list)
  ifelse petrol-car? (add (1 + count link-neighbors with petrol-car?) to end of list)
    (add (count link-neighbors with petrol-car?) to end of list)
  ifelse diesel-car? (add (1 + count link-neighbors with diesel-car?) to end of list)
    (add (count link-neighbors with diesel-car?) to end of list)
Reduce list length if list is longer than memory
  If innovator? (
    If length of ev-list > (innovator-memory * 10) (remove item 0 from list)
    If length of petrol-list > (innovator-memory * 10) (remove item 0 from list)
```

```

        If length of diesel-list > (innovator-memory * 10) (remove item 0 from list)
    If early-adopter? (
        If length of ev-list > (early-adopter-memory * 10) (remove item 0 from list)
        If length of petrol-list >(early-adopter-memory*10) (remove item 0 from list)
        If length of diesel-list >(early-adopter-memory*10) (remove item 0 from list)
    If early-majority? (
        If length of ev-list > (early-majority-memory * 10) (remove item 0 from list)
        If length of petrol-list>(early-majority-memory*10) (remove item 0 from list)
        If length of diesel-list>(early-majority-memory*10) (remove item 0 from list)
    If late-majority? (
        If length of ev-list > (late-majority-memory * 10) (remove item 0 from list)
        If length of petrol-list >(late-majority-memory*10) (remove item 0 from list)
        If length of diesel-list >(late-majority-memory*10) (remove item 0 from list)
    If laggards? (
        If length of ev-list > (laggards -memory * 10) (remove item 0 from list)
        If length of petrol-list >(laggards-memory*10) (remove item 0 from list)
        If length of diesel-list >(laggards-memory*10) (remove item 0 from list)
    Ask turtles with car-age >= car-lifetime to buy car
        Ifelse ticks < years-to-ev-invention * 10 (
            Ifelse diesel-car? (
                ifelse average of petrol-list >= change-threshold (become-PETROLowner)
                    (become-DIESELowner)
                )
            (ifelse petrol-car? (
                Ifelse average of diesel-list >= change-threshold
                    (become-DIESELowner)
                    (become-PETROLowner))
                (show "DECISION ERROR IN BUYING CAR BEFORE EV INVENTION")
            ))
        ( ifelse electric-car? (
            If average of both diesel-list and petrol-list >= change-threshold (
                If innovator? or early-adopter? (
                    if average of diesel-list > average of petrol-list
                        become-PETROLowner
                    if average of diesel-list < average of petrol-list
                        become-DIESELowner
                    if average of diesel-list = average of petrol-list
                        choose random between become-PETROLowner
                            or become-DIESELowner)
                if early-majority? or late-majority? (
                    choose random between become-PETROLowner or
                        become-DIESELowner)
                if laggards? (
                    if average of diesel-list > average of petrol-list
                        become-DIESELowner
                    if average of diesel-list < average of petrol-list
                        become-PETROLowner
                    if average of diesel-list = average of petrol-list
                        choose random between become-PETROLowner
                            or become-DIESELowner) )
            )
        )
    )

```

```
if average of diesel-list >= change-threshold but average of petrol-list not
    (become-DIESELowner)
If average of petrol-list >= change-threshold but average of diesel-list is not
    (become-PETROLowner)
If average of both diesel-list and petrol-list < change-threshold
    (become-EVowner)
)
(
ifelse Diesel-car? (
    If average of both ev-list and petrol-list >= change-threshold (
        If innovator? or early-adopter? (
            if average of ev-list > average of petrol-list
                become-PETROLowner
            if average of ev-list < average of petrol-list
                become-EVowner
            if average of ev-list = average of petrol-list
                choose random between become-
                PETROLowner or become-EVowner)
        if early-majority? or late-majority? (
            choose random between become-PETROLowner
            or become-EVowner)
        if laggards? (
            if average of ev-list > average of petrol-list
                become-EVowner
            if average of ev-list < average of petrol-list
                become-PETROLowner
            if average of ev-list = average of petrol-list
                choose random between become-
                PETROLowner or become-EVowner) )
    if average of ev-list >= change-threshold but average petrol-list not
        (become-EVowner)
    If average of petrol-list >= change-threshold but average ev-list not
        (become-PETROLowner)
    If average of both ev-list and petrol-list < change-threshold
        (become-DIESELowner)
    )
)
(ifelse Petrol-car? (
    If average of ev-list and diesel-list >= change-threshold (
        If innovator? or early-adopter? (
            if average of ev-list > mean of diesel -list
                become-DIESELowner
            if average of ev-list < mean of diesel -list
                become-EVowner
            if average of ev-list = mean of diesel -list
                choose random between
                become-DIESELowner or
                become-EVowner)
        if early-majority? or late-majority? (
            choose random between become-
            DIESELowner or become-EVowner)
```


SOFTWARE IMPLEMENTATION

The narrative and pseudo code are implemented in an modeling environment called NetLogo. NetLogo is a free program that allows for relatively easy implementation of agent-based models. It comes with a large number of examples and a huge online community. In this way, NetLogo is an ideal tool for starting programmers like ourselves. The first paragraph of this section shows the code as it is implemented in NetLogo. The second paragraph identifies the inconsistencies between the code and the model narrative/pseudo code.

NETLOGO CODE

```

1  globals [
2    buyers-EV
3    buyers-Petrol
4    buyers-Diesel
5  ]
6
7
8  turtles-own
9  [
10   Electric-Car?           ;; if true, the turtle has an Electric-Vehicle
11   Petrol-Car?            ;; if true, the turtle has an Petrol-Car
12   Diesel-Car?           ;; if true, the turtle has an Diesel-Car
13
14   Innovator?            ;; if true, turtle falls in innovator category
15   Early-adopter?        ;; if true, turtle falls in early adopter category
16   Early-majority?       ;; if true, turtle falls in early majority category
17   Late-majority?        ;; if true, turtle falls in late majority category
18   Laggards?            ;; if true, turtle falls in laggards category
19
20   car-age               ;; number of ticks since the car was bought
21   car-lifetime          ;; number of ticks the car 'survives'
22
23   attitude-to-innovation ;; Determines the innovation category & change threshold of turtle
24   change-threshold      ;; The willingness of an agent to change its technology
25
26   ev-list               ;; Holds data on the amount of neighbors with an electric car for a specified period of ticks
27   petrol-list          ;; Holds data on the amount of neighbors with a petrol car for a specified period of ticks
28   diesel-list          ;; Holds data on the amount of neighbors with a diesel car for a specified period of ticks
29
30   car-history-list      ;; Holds data on the history of car types of each turtle
31 ]
32
33 ;;;;;;;;;;;;;;;;;;;;;;;;;;
34 ;; Setup Procedures ;;
35 ;;;;;;;;;;;;;;;;;;;;;;;;;;
36
37 to setup
38   clear-all
39   ask patches [set pcolor white]
40   setup-nodes
41   setup-initial-car
42   setup-network
43   setup-innovation-preferences
44   setup-lists
45   set buyers-EV 0
46   set buyers-Petrol 0
47   set buyers-Diesel 0
48   reset-ticks
49 end
50
51 to setup-nodes
52   set-default-shape turtles "circle"
53   crt number-of-car-owners[
54     setxy (random-xcor * 0.90) (random-ycor * 0.90) ]    ;; don't put any nodes real close to the edges
55 end
56
57 to setup-initial-car
58   ask turtles [become-PETROLowner]
59   ask n-of ((number-of-car-owners * initial-diesel-percentage) / 100) turtles with [Petrol-Car?] [become-DIESELowner]
60   ;; it might be possible that a laggard start with an insuperior technology (i.e. diesel), while being surrounded by other technologies
61
62   ask turtles [
63     if initial-car-age = 0 [set car-age 0]
64     if initial-car-age = "random" [set car-age random car-lifetime]
65   ]
66
67   ;; this sets the turtles car properties
68 end
69

```

```

70 to setup-network
71 let num-links (average-links-per-car-owner * number-of-car-owners) / 2           ;; divide by 2 as each link connects two nodes
72 ask turtles [if count link-neighbors <= 0
73 [create-link-with min-one-of (other turtles with [not link-neighbor? myself]) [distance myself]] ;; make sure every node has at least 1 link
74 while [count links <= num-links ]
75 [ask one-of turtles
76 [ [ let choice (min-one-of (other turtles with [not link-neighbor? myself])
77 [distance myself])
78 if choice != nobody [ create-link-with choice ]           ;; add connect links if node is unconnected
79 ]
80 ]
81
82 ask links [ set color grey ]
83
84 repeat 5
85 [layout-spring turtles links 0.5 (world-width / (sqrt number-of-car-owners)) 1] ;; make network look bit nicer
86 end
87
88 to setup-innovation-preferences
89 ask turtles [
90 set attitude-to-innovation random-normal 0.5 0.16           ;; normal distribution N(0.5;0.16), this follows theory
91 set change-threshold (round (attitude-to-innovation * count link-neighbors)) ;; threshold is now set to number of neighbors
92 if change-threshold < 0 [set change-threshold 0]
93 if change-threshold > count link-neighbors [set change-threshold count link-neighbors]
94
95 ;; Make sure no turtle is categorized yet
96 set Innovator? false
97 set Early-adopter? false
98 set Early-majority? false
99 set Late-majority? false
100 set Laggards? false
101
102 ;; arrange categorization of turtles
103 if attitude-to-innovation >= 0.84 [set Laggards? true]
104 if attitude-to-innovation < 0.84 [set Late-majority? true]
105 if attitude-to-innovation < 0.50 [set Early-majority? true set late-majority? false]
106 if attitude-to-innovation < 0.16 [set Early-adopter? true set early-majority? false]
107 if attitude-to-innovation < 0.025 [set Innovator? true set early-adopter? false]
108
109 ]
110 end
111
112 to setup-lists
113 ask turtles [
114 ;; record initial environment of turtle + own car type
115 set ev-list (list 0)
116 ifelse Petrol-Car? [set petrol-list (list (1 + count link-neighbors with [Petrol-Car?]))] [set petrol-list (list count link-neighbors with [Petrol-Car?])]
117 ifelse Diesel-Car? [set diesel-list (list (1 + count link-neighbors with [Diesel-Car?]))] [set diesel-list (list count link-neighbors with [Diesel-Car?])]
118
119 ;; record initial car type of turtle
120 ifelse Petrol-Car? [set car-history-list (list "PETROL")]
121 [ifelse Diesel-Car? [set car-history-list (list "DIESEL")]
122 [show "ERROR in car history list"]
123 ]
124 ]
125 end
126
127
128
129 ;;::::::::::::::::::::::::::
130 ;; Go Procedures ;;
131 ;;::::::::::::::::::::::::::
132
133 to go
134 set buyers-EV 0
135 set buyers-Petrol 0
136 set buyers-Diesel 0
137 ask turtles [set car-age car-age + 1]
138 ask turtles [update-memory]
139 ask turtles with [car-age >= car-lifetime] [buy-car]
140 tick
141 end
142
143 to update-memory           ;; 10 ticks equal 1 year, this section does not cope with shortening 'car-owner-memory' during run time adequately
144 ifelse Electric-Car? [set ev-list lput (1 + count link-neighbors with [Electric-Car?]) ev-list] [set ev-list lput count link-neighbors with [Electric-Car?] ev-list]
145 ifelse Petrol-Car? [set petrol-list lput (1 + count link-neighbors with [Petrol-Car?]) petrol-list] [set petrol-list lput count link-neighbors with [Petrol-Car?] petrol-list]
146 ifelse Diesel-Car? [set diesel-list lput (1 + count link-neighbors with [Diesel-Car?]) diesel-list] [set diesel-list lput count link-neighbors with [Diesel-Car?] diesel-list]
147
148
149 if Innovator? [
150 ;; limit list for innovators
151 if length ev-list > (innovator-memory * 10) [set ev-list remove-item 0 ev-list]
152 if length petrol-list > (innovator-memory * 10) [set petrol-list remove-item 0 petrol-list]
153 if length diesel-list > (innovator-memory * 10) [set diesel-list remove-item 0 diesel-list]
154 ]
155

```

```

156 if Early-adopter? [
157   ;; limit list for early-adopters
158   if length ev-list > (early-adopter-memory * 10) [set ev-list remove-item 0 ev-list]
159   if length petrol-list > (early-adopter-memory * 10) [set petrol-list remove-item 0 petrol-list]
160   if length diesel-list > (early-adopter-memory * 10) [set diesel-list remove-item 0 diesel-list]
161 ]
162
163 if Early-majority? [
164   ;; limit list for early-majority
165   if length ev-list > (early-majority-memory * 10) [set ev-list remove-item 0 ev-list]
166   if length petrol-list > (early-majority-memory * 10) [set petrol-list remove-item 0 petrol-list]
167   if length diesel-list > (early-majority-memory * 10) [set diesel-list remove-item 0 diesel-list]
168 ]
169
170 if Late-majority? [
171   ;; limit list for late-majority
172   if length ev-list > (late-majority-memory * 10) [set ev-list remove-item 0 ev-list]
173   if length petrol-list > (late-majority-memory * 10) [set petrol-list remove-item 0 petrol-list]
174   if length diesel-list > (late-majority-memory * 10) [set diesel-list remove-item 0 diesel-list]
175 ]
176
177 if Laggards? [
178   ;; limit list for laggards
179   if length ev-list > (laggard-memory * 10) [set ev-list remove-item 0 ev-list]
180   if length petrol-list > (laggard-memory * 10) [set petrol-list remove-item 0 petrol-list]
181   if length diesel-list > (laggard-memory * 10) [set diesel-list remove-item 0 diesel-list]
182 ]
183 end
184
185
186 to buy-car
187   ifelse ticks < (years-to-ev-invention * 10) [
188     ;; procedure in case ev is not evented yet
189     ifelse Diesel-Car? [
190       ifelse mean petrol-list >= change-threshold [become-PETROLowner] [become-DIESELowner]]
191     [ifelse Petrol-Car? [
192       ifelse mean diesel-list >= change-threshold [become-DIESELowner] [become-PETROLowner]]
193     [show "DECISION ERROR IN BUYING CAR BEFORE EV INVENTION"]
194     ]
195     ]
196
197     [
198       ;; procedure in case ev is invented
199       ifelse Electric-Car? [
200         if (mean diesel-list >= change-threshold) and (mean petrol-list >= change-threshold) [
201           if Innovator? or early-adopter? [
202             if mean diesel-list > mean petrol-list [become-PETROLowner]
203             if mean diesel-list < mean petrol-list [become-DIESELowner]
204             if mean diesel-list = mean petrol-list [ifelse random-float 1 <= 0.50 [become-DIESELowner] [become-PETROLowner]]]
205           if early-majority? or late-majority? [
206             ifelse random-float 1 <= 0.50 [become-DIESELowner] [become-PETROLowner]]
207           if laggards? [
208             if mean diesel-list < mean petrol-list [become-PETROLowner]
209             if mean diesel-list > mean petrol-list [become-DIESELowner]
210             if mean diesel-list = mean petrol-list [ifelse random-float 1 <= 0.50 [become-DIESELowner] [become-PETROLowner]]]]]
211         if (mean diesel-list >= change-threshold) and (mean petrol-list < change-threshold) [become-DIESELowner]
212         if (mean diesel-list < change-threshold) and (mean petrol-list >= change-threshold) [become-PETROLowner]
213         if (mean diesel-list < change-threshold) and (mean petrol-list < change-threshold) [become-EVowner]]
214
215       [ifelse Diesel-Car? [
216         if (mean ev-list >= change-threshold) and (mean petrol-list >= change-threshold) [
217           if Innovator? or early-adopter? [
218             if mean ev-list > mean petrol-list [become-PETROLowner]
219             if mean ev-list < mean petrol-list [become-EVowner]
220             if mean ev-list = mean petrol-list [ifelse random-float 1 <= 0.50 [become-EVowner] [become-PETROLowner]]]
221           if early-majority? or late-majority? [
222             ifelse random-float 1 <= 0.50 [become-EVowner] [become-PETROLowner]]
223           if laggards? [
224             if mean ev-list < mean petrol-list [become-PETROLowner]
225             if mean ev-list > mean petrol-list [become-EVowner]
226             if mean ev-list = mean petrol-list [ifelse random-float 1 <= 0.50 [become-EVowner] [become-PETROLowner]]]]]
227         if (mean ev-list >= change-threshold) and (mean petrol-list < change-threshold) [become-EVowner]
228         if (mean ev-list < change-threshold) and (mean petrol-list >= change-threshold) [become-PETROLowner]
229         if (mean ev-list < change-threshold) and (mean petrol-list < change-threshold) [become-DIESELowner]]
230

```

```

231 [ifelse Petrol-Car? {
232   if (mean ev-list >= change-threshold) and (mean diesel-list >= change-threshold) {
233     if Innovator? or early-adopter? {
234       if mean ev-list > mean diesel-list [become-DIESELowner]
235       if mean ev-list < mean diesel-list [become-EVowner]
236       if mean ev-list = mean diesel-list [ifelse random-float 1 <= 0.50 [become-EVowner] [become-DIESELowner]]]
237     if early-majority? or late-majority? {
238       ifelse random-float 1 <= 0.50 [become-EVowner] [become-DIESELowner]]
239     if laggards? {
240       if mean ev-list < mean diesel-list [become-DIESELowner]
241       if mean ev-list > mean diesel-list [become-EVowner]
242       if mean ev-list = mean diesel-list [ifelse random-float 1 <= 0.50 [become-EVowner] [become-DIESELowner]]]
243     if (mean ev-list >= change-threshold) and (mean diesel-list < change-threshold) [become-EVowner]
244     if (mean ev-list < change-threshold) and (mean diesel-list >= change-threshold) [become-DIESELowner]
245     if (mean ev-list < change-threshold) and (mean diesel-list < change-threshold) [become-PETROLowner]
246 [show "DECISION ERROR IN BUYING CAR AFTER EV INVENTION"]
247 ]
248 ]
249 ]
250
251 if Electric-Car? [set car-history-list lput "EV" car-history-list]
252 if Diesel-Car? [set car-history-list lput "DIESEL" car-history-list]
253 if Petrol-Car? [set car-history-list lput "PETROL" car-history-list]
254 end
255
256 ;;;;;;;;;;;;;;
257 ;; Car owner Procedures ;;
258 ;;;;;;;;;;;;;;
259
260 to become-EVowner
261   set Electric-Car? true
262   set Petrol-Car? false
263   set Diesel-Car? false
264   set color lime
265   set size 2
266   set buyers-EV buyers-EV + 1
267
268
269   set car-age 0
270   set car-lifetime random-normal (average-ev-car-life * 10) 20 ;;10 ticks equal 1 year --> car-lifetime now follows N(average car life, 2) year distribution
271 end
272
273 to become-PETROLowner
274   set Electric-Car? false
275   set Petrol-Car? true
276   set Diesel-Car? false
277   set color red
278   set size 2
279   set buyers-Petrol buyers-Petrol + 1
280
281
282   set car-age 0
283   set car-lifetime random-normal (average-petrol-car-life * 10) 20 ;;10 ticks equal 1 year --> car-lifetime now follows N(average car life, 2) year distribution
284 end
285
286 to become-DIESELowner
287   set Electric-Car? false
288   set Petrol-Car? false
289   set Diesel-Car? true
290   set color sky
291   set size 2
292   set buyers-Diesel buyers-Diesel + 1
293
294
295   set car-age 0
296   set car-lifetime random-normal (average-diesel-car-life * 10) 20 ;;10 ticks equal 1 year --> car-lifetime now follows N(average car life, 2) year distribution
297 end

```


INCONSISTENCIES BETWEEN MODEL AND NARRATIVE/PSEUDO CODE

There are three inconsistencies between the code presented in the previous paragraph and the narrative and pseudo code of the previous section. Firstly, some code lines are added to function as metric in the analysis of the model results. This applies to code line 1 to 5 and the lines 266, 279 and 292. These will be further elaborated upon in further sections.

The two other relate to the way the model is graphically represented in NetLogo. When the model is set up in the way it is explained in the narrative and pseudo code, is quite possible that car owners are located really close to the edges of the system representation or (partly) overlap on their location (see figure 3 for an example).

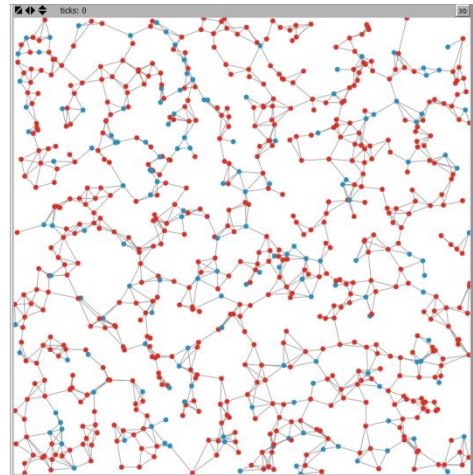


Figure 3: graphic representation without extra code

For the first reason, we multiplied the coordinates of each turtle with a factor 0.9 (line 54 of the code). For the second reason we repeat the following algorithm five times (line 84 and 85 of the code):

Repeat 5

[layout-spring turtles links 0.5 (world-width / (sqrt number-of-car-owners)) 1]

In this way, the graphical representation looks a bit nicer (see figure 4).

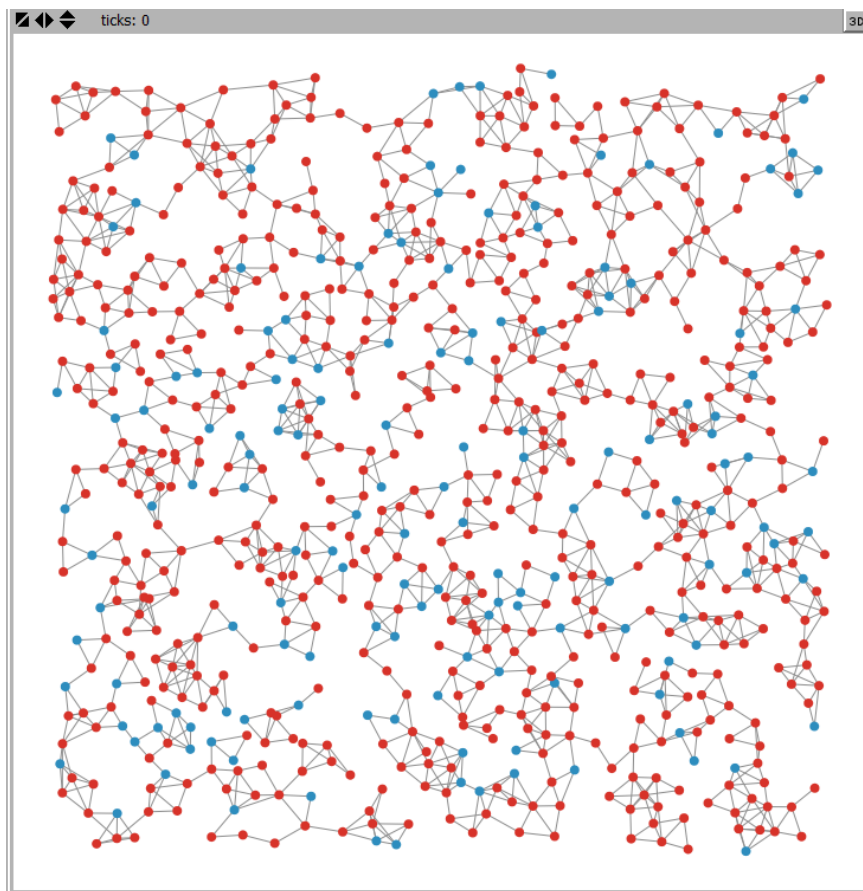


figure 4: Graphic representation with extra code

MODEL VERIFICATION

In this section, we check the model against its conceptual design. In other words, we check whether we built the model in the way we intended it. We will verify the model at three levels. First, we discuss relevant variables for verification and describe a way to measure them in the model. In the second paragraph, we'll perform single-agent verification checks, in which the behavior of car owners is verified. The third paragraph is about minimal model interaction verification, in which we'll verify the interaction between a minimal set of agents. The fourth paragraph is about multi-agent verification, in which we verify the emergent behavior of multiple agents. The NetLogo model that is used for the verification (including the verification coding) can be found in appendix A.

RECORDING AND TRACKING AGENT BEHAVIOR

In NetLogo, we define a variable called `log-car-owner-stats` (a Boolean) to (de)activate the procedures for the verification as we describe them here. Concerning verification, two main aspects of the model are of fundamental importance. First, this is that a car owner buys a car after that his car is equal to or over the car lifetime. For this reason, we add the following line of code in the `go` procedure (between line 137 and 138 of the code given in the previous section):

```
ask turtles [if Log-car-owner-stats? [print (word self "my car is now " car-age ", while the lifetime of my car is" car-lifetime)]]
```

Secondly, and most importantly, it is important that the right new car is bought, i.e. the decision procedure is followed correctly. In order to measure this, we need some more code. First we define two new variables that turtles own: `'car-type'` and `'adopter-category'`. Next, we add the following lines of code in the car buying procedures. Between line 260 and 261 of the code in the previous section:

```
if Log-car-owner-stats? [print (word self "I buy EV, as my change threshold is " change-threshold " and my memory values are " mean-ev-list " " mean-petrol-list " " mean-diesel-list " for ev, petrol and diesel respectively)]]
```

```
if Log-car-owner-stats? [print (word self "By the way, " adopter-category " is my adopter category and I had a " car-type "car")]]
```

Between line 273 and 274 of the code in the previous section:

```
if Log-car-owner-stats? [print (word self "I buy Petrol, as my change threshold is " change-threshold " and my memory values are " mean-ev-list " " mean-petrol-list " " mean-diesel-list " for ev, petrol and diesel respectively)]]
```

```
if Log-car-owner-stats? [print (word self "By the way, " adopter-category " is my adopter category and I had a " car-type "car")]]
```

Between line 286 and 287 of the code in the previous section:

```
if Log-car-owner-stats? [print (word self "I buy Diesel, as my change threshold is " change-threshold " and my memory values are " mean-ev-list " " mean-petrol-list " " mean-diesel-list " for ev, petrol and diesel respectively)]]
```

```
if Log-car-owner-stats? [print (word self "By the way, " adopter-category " is my adopter category and I had a " car-type "car")]]
```

However, for these lines of code to work, we need to rewrite some other code as well. The way the code is rewritten is shown in appendix B. Note that by using this code, we also gain insight in the change threshold and memories of a car owner.

SINGLE-AGENT VERIFICATION

In this type of verification, we explore the behavior of a single agent. For that, we'll use various types of tests. The first are theoretical prediction and sanity checks. The second focusses on extreme values for breaking the agent, i.e. define the edges of normal behavior.

THEORETICAL PREDICTION AND SANITY CHECKS

Here, we test the output of the agents to normal operating inputs. Hence, we provide well defined inputs to the agents. We check the outputs against predefined theoretical predictions. Any deviation from the theory directly points at an implementation error.

Buy new car if car reaches lifetime

- If the age of the car is as big as, or exceeds the lifetime of the car, a car owner should buy a new car. **Confirmed**

Change threshold should be zero

- As the car owner has no social network, his change threshold should be zero, as his attitude to innovation is multiplied by 0. **Confirmed**

Car buying procedure without influencing memory

- With a change threshold of zero, and EVs not yet invented, car owners should switch their car type from petrol to diesel and vice versa every time they buy a car. **Error found**. EVs still occurred. This was traced back to the check of the time step where EVs are invented, i.e. EVs were not taken into account when 'years-to-ev-invention' * 10 <= ticks. Logically, this should be the other way around: ticks <= 'years-to-ev-invention' * 10. In the code presented in the section on software implementations, this was already adopted. **Fixed, reverified and confirmed.**
- With a change threshold of two, and EVs not yet invented, car owners should never switch from their initial car type as the threshold is not exceeded. **Confirmed**
- With a change threshold of zero, and EVs invented, car owners should choose randomly between the car type they do not have, as both other car type exceed the threshold (of zero). **Confirmed**
- With a change threshold of two, and EVs invented, car owners should never switch from their initial car types as the threshold is not exceeded. **Confirmed**

Influence of adopter category after EV invention

- When adopter category is innovator or early-adopter, when EVs are invented and the change threshold is zero, car owners will switch from technology continuously as long as their memory is shorter than the car life (i.e. they can only remember their current car. Hence, there is no significant difference between the other car types). **Confirmed**
- When adopter category is early-majority or late-majority, when EVs are invented and the change threshold is zero, car owners will switch from technology continuously as long as their memory is shorter than the car life. **Confirmed**
- When adopter category is laggards, when EVs are invented and the change threshold is zero, car owners will switch from technology continuously as long as their memory is shorter than the car life. **Confirmed**

BREAK THE AGENT & EXTREME VALUE TESTS

What we try to establish in these kinds of tests is to find out what parameters make the agent break and/or perform unintended actions. This shows the limitations of the used code (or implementations errors). Subsequently, we need to make sure an agent never receives this value. The structure of this paragraph is similar to the previous one. We will not use dynamic signal testing for this verification, because of a lack of knowledge on matlab (that is needed to generate different signal series).

Negative change threshold value

- In case of a negative change threshold value, the change threshold is always met (as the car owners cannot have a negative memory). Hence, car owners base their decisions based on their adopter category. **Confirmed**
- The only way possible to get to a negative change threshold value is to draw a negative number from the normal distribution on 'attitude to innovation' (and a negative number of neighbors is impossible). **Limitation found.** As adaption, we set the change threshold to 0 when the change threshold is below 0. In the code presented in the section on software implementations, this was already adopted. **Fixed, reverified and confirmed.**
- The only way possible to get to a change threshold that exceeds the number of links in the car owners network, is to draw a number bigger than 1 from the normal distribution on 'attitude to innovation'. **Limitation found.** As adaption, we set the change threshold to the number of links a car owners has, when the change threshold exceeds the number of links of a car owner. In the code presented in the section on software implementations, this was already adopted. **Fixed, reverified and confirmed.**

Negative memory

- When an agent receives a negative input on its memory, this is just incorporated in the overall effect of the memory. In other words, an agent can work with a negative input value on its memory. **Confirmed**
- As the memory consists of a count of links with a characteristic on the other end, the memory value cannot get negative by itself. **Confirmed**

Extreme car ages (note that for this test the only-buying-procedure? Variable has to be off)

- When a car receives a negative car age, this is no problem for the agent. It will still count till the car lifetime is met and then buy a new car. In other words, the effect is that the effective car lifetime is longer than the originally given car lifetime. **Confirmed**
- When a car receives a car age that exceeds the car lifetime, the car owner buys a new car in the next tick. **Confirmed**

Length of memory

- When amount of information in the car owner memory exceeds the maximum length of a memory (for example when the memory decreases over time), the length of information will not automatically shorten as well. This is because per time step, only a single value is added and only a single value can be removed from the memory. **Limitation found.** The consequence of this limitation is that the model is not capable of handling a shortened memory over time. Increasing the memory over time is possible.

Car buying procedure with influencing of EV memory and change threshold

- With a change threshold of zero, EVs invented, the adopter category being innovator or early-adopter, and a car owner with a strong (positive) memory on EVs, a car owner will switch from car type, but not to EV. **Confirmed**
- With a change threshold of zero, EVs invented, the adopter category being early-majority of late-majority, and a car owner with a strong (positive) memory on EVs, a car owner will randomly switch from car type. **Confirmed**
- With a change threshold of zero, EVs invented, the adopter category being laggard, and a car owner with a strong (positive) memory on EVs, a car owner will switch car type to EV (and remain EV for the rest of the run time). **Confirmed**

Car buying procedure with influencing of Petrol memory and change threshold

- With a change threshold of zero, EVs not invented, the adopter category being innovator or early-adopter, and a car owner with a strong (positive) memory on petrol cars, a car owner will switch to diesel (and remain that for the rest of the run time). **Confirmed**
- With a change threshold of zero, EVs not invented, the adopter category being early-majority of late-majority, and a car owner with a strong (positive) memory on petrol cars, a car owner will randomly switch from car type (between diesel and petrol obviously). **Confirmed**
- With a change threshold of zero, EVs not invented, the adopter category being laggard, and a car owner with a strong (positive) memory on petrol cars, a car owner will switch car type to petrol (and remain petrol for the rest of the run time). **Confirmed**
- With a change threshold of zero, EVs invented, the adopter category being innovator or early-adopter, and a car owner with a strong (positive) memory on petrol cars, a car owner will switch from car type, but not to petrol. **Confirmed**
- With a change threshold of zero, EVs invented, the adopter category being early-majority of late-majority, and a car owner with a strong (positive) memory on petrol cars, a car owner will randomly switch from car type. **Confirmed**
- With a change threshold of zero, EVs invented, the adopter category being laggard, and a car owner with a strong (positive) memory on petrol cars, a car owner will switch car type to petrol (and remain petrol for the rest of the run time). **Confirmed**

Car buying procedure with influencing of diesel memory and change threshold

- With a change threshold of zero, EVs not invented, the adopter category being innovator or early-adopter, and a car owner with a strong (positive) memory on diesel cars, a car owner will switch to petrol (and remain that for the rest of the run time). **Confirmed**
- With a change threshold of zero, EVs not invented, the adopter category being early-majority of late-majority, and a car owner with a strong (positive) memory on diesel cars, a car owner will randomly switch from car type (between diesel and petrol obviously). **Confirmed**
- With a change threshold of zero, EVs invented, the adopter category being innovator or early-adopter, and a car owner with a strong (positive) memory on diesel cars, a car owner will switch from car type, but not to diesel. **Confirmed**
- With a change threshold of zero, EVs invented, the adopter category being early-majority of late-majority, and a car owner with a strong (positive) memory on diesel cars, a car owner will randomly switch from car type. **Confirmed**
- With a change threshold of zero, EVs invented, the adopter category being laggard, and a car owner with a strong (positive) memory on diesel cars, a car owner will switch car type to diesel (and remain petrol for the rest of the run time). **Confirmed**

INTERACTION TESTING IN A MINIMAL ENVIRONMENT

Here, we explore the behavior of a minimal set of agents. As we only use one type of agents in this model, we include 2 car owners. Furthermore, we use the same tests as in the previous step, i.e. theoretical prediction and sanity checks and break the agent tests. In all experiments, we start with a car owner that owns a diesel car and a car owner that has a petrol car.

THEORETICAL PREDICTION AND SANITY CHECKS

Change thresholds

- When a car owner is innovator, early-adopter, or early-majority, his threshold should be 0. **Confirmed**
- When a car owners is late-majority or laggard, his threshold should be 1. **Confirmed**

Effect of adopter categories when EVs are not yet invented

- When both car owners are innovator or early adopter and EVs are not yet invented, both will switch car type independent of the other car owner. **Confirmed**
- When both car owners are early majority and EVs are not invented yet, both will switch independent of the other car owner. **Confirmed**
- When both car owners are late majority and EVs are not invented yet, the car owner that has to buy a new car first will switch car type. The other car owner will only switch is he buys a new car in the same tick. In other words, one technology will be extinguished. **Confirmed**
- When both car owners are laggard and EVs are not invented yet, the car owner that has to buy a new car first will switch car type. The other car owner will only switch is he buys a new car in the same tick. **Confirmed**
- When a single car owner is innovator, early-adopter or early-majority, and the other is late-majority or laggard, and EVs are not yet invented, the first car owner will change car type every time he buys a new car, the latter car owner will only change car type when over the length of his memory, the first car owner did not buy a new car. **Confirmed**

Effect of adopter categories when EVs are invented

- When both car owners are innovator and EVs are invented, both will switch to another car type that the other does not have at that moment. **Confirmed**
- When at least one of the car owners is early adopter, and the other innovator or early adopter, and EVs are invented, the early adopters will switch to the technology they do not have themselves and did see the least (note: this thus depends on memory length) at the other car owner. **Confirmed**
- When both car owners are early majority and EVs are invented, they will switch their car type randomly, as both other technologies exceed the threshold (of 0). **Confirmed**
- When both car owners are late-majority and EVs are invented, only the car owner that has to buy a car first will buy the car type of the other car owner. The other car owner only switches if he buys a car in the same tick. So, in nearly all cases, a single technology 'wins'. EVs will not be introduced. **Confirmed**
- When one car owner is early-majority and the other is late-majority and EVs are invented, the early-majority randomly chooses between the other technologies . The late-majority car owner will only follow the early-adopter car owner when the memory of the late majority car owner is filled with a single car type of the early adopter (i.e. he did not buy a new car for as long as the memory of the late majority car owner knows). **Confirmed**

- When both car owners are laggards and EVs are invented, only the car owner that has to buy a car first will buy the car type of the other car owner. The other car owner only switches if he buys a car in the same tick. So, in nearly all cases, a single technology ‘wins’. EVs will not be introduced. **Confirmed**

Effect of average car lives and memory lengths

- When a car owner has a threshold of 0, and the memory is just a single tick, the car life does not matter for choice of car type. **Confirmed**
- As explained, in the verification of ‘effect of adopter categories when EVs are invented’, the car owner memory may have influence on the choice of car type. **Confirmed**
- The average car life has effect on the variation of car types in the memory of a car owner. **Confirmed**

BREAK THE AGENT & EXTREME VALUE TESTS*Car buying procedure with influencing of EV memory and change threshold*

- With a change threshold of three, EVs invented, the adopter category being innovator or early-adopter, and a car owner with a strong (positive) memory on EVs, a car owner will switch to EV (and remain EV for the rest of the run time). **Confirmed**
- With a change threshold of three, EVs invented, the adopter category being early-majority of late-majority, and a car owner with a strong (positive) memory on EVs, a car owner will switch to EV (and remain EV for the rest of the run time). **Confirmed**
- With a change threshold of three, EVs invented, the adopter category being laggard, and a car owner with a strong (positive) memory on EVs, a car owner will switch to EV (and remain EV for the rest of the run time). **Confirmed**

Car buying procedure with influencing of Petrol memory and change threshold

- With a change threshold of three, EVs invented, the adopter category being innovator or early-adopter, and a car owner with a strong (positive) memory on petrol cars, a car owner will switch to petrol (and remain petrol for the rest of the run time). **Confirmed**
- With a change threshold of three, EVs invented, the adopter category being early-majority of late-majority, and a car owner with a strong (positive) memory on petrol cars, a car owner will switch to petrol (and remain petrol for the rest of the run time). **Confirmed**
- With a change threshold of three, EVs invented, the adopter category being laggard, and a car owner with a strong (positive) memory on petrol cars, a car owner will switch to petrol (and remain petrol for the rest of the run time). **Confirmed**

Car buying procedure with influencing of diesel memory and change threshold

- With a change threshold of three, EVs invented, the adopter category being innovator or early-adopter, and a car owner with a strong (positive) memory on diesel cars, a car owner will switch to diesel (and remain petrol for the rest of the run time). **Confirmed**
- With a change threshold of three, EVs invented, the adopter category being early-majority of late-majority, and a car owner with a strong (positive) memory on diesel cars, a car owner will switch to diesel (and remain petrol for the rest of the run time). **Confirmed**
- With a change threshold of three, EVs invented, the adopter category being laggard, and a car owner with a strong (positive) memory on diesel cars, a car owner will switch to diesel (and remain petrol for the rest of the run time). **Confirmed**

MULTI-AGENT TESTING

In multi-agent testing, we run the entire model. What we are testing for is the coherence of emergent patterns in the model. Therefore, in this paragraph, we will perform some variability testing, verify some input distributions and do a timeline sanity check.

VARIABILITY TESTING

By variability testing, we explore the variability of the output in different regions of the parameter space. Therefore, we determine a number of output variables. Furthermore, we take a 100 repetitions across the parameter space. Subsequently, we check whether the outcomes match the expected results and perform some statistical examination on the (variability) in the results. Finally, we check for strange outcomes. In the case outcomes cannot be explained through model logic, implementation errors are likely.

Determine output variables

As output variables we will use the amount of EVs as this is the key performance indicator we are interested in following traditional stability theory. We will take the amount of EVs after 50, 1050, 3050 and 5050 ticks to get insight in the emergent behavior.

Parameter space

In this verification experiment, we choose the reference model setting as follows:

- Number of car owners: 750
- Average links per car owner: 4
- Years to EV invention: 100
- Initial diesel percentage: 29%
- Innovator memory: 0.1 year
- Early-adopter memory: 2 years
- Early majority memory: 4 years
- Late majority memory: 6 years
- Laggard memory: 8 years
- Average petrol car life: 12 years
- Average diesel car life: 12 years
- Average EV car life: 12 years

As variation in the parameter space we will change one variable at a time. We test for the average links per car owner set to 2, the average links per car owner set to 6 and the average EV car life set to 18.

Expected results

Table 1 shows the expected results of the model variation

Input	Expected output (average number of EVs on 100 replications)			
	Tick 50	Tick 1050	Tick 3050	Tick 5050
<i>Reference</i>	-	-	-	-
<i>Average links to 6</i>	No difference	Less	Less	Less
<i>Average links to 2</i>	No difference	Much more	Much more	Much more
<i>Ev car life 18</i>	No difference	Slightly more	more	more

Table 1: Expected results on verification variability tests.

Outcome of analysis

The outcome over the replications is given in table 2. As it seems, this all corresponds to our expectations.

Input	output (average number of EVs on 100 replications)			
	Tick 50	Tick 1050	Tick 3050	Tick 5050
Reference	0	5.26	13.8	13.8
Average links to 6	0	1.82	3.52	3.38
Average links to 2	0	30.3	106	111
Ev car life 18	0	5.66	26.1	28.5

Table 2: Expected results on verification variability tests.

We will now further examine the statistics of the outcomes, more specifically the variance and skewness of the results. High variances don't necessarily mean a problem. However, combined with a high skewness, this may suggest large outliers. Hence, closer inspection is needed to identify we deal with non-linear chaotic model behavior or if an artifact is present. For this analysis, we'll use the variance and skewness of the distributions over the repetitions.

Skewness is a measure of asymmetry of the probability distribution. The skewness value can be positive, negative and zero. Skewness is a sign for the distribution of values on both sides of the mean. Therefore, the skewness value might help us notes significant changes in the distributions outcome, implying the existence of an artifact. As measure for skewness, we will use a relatively simple calculation as suggested by Pearson (Weisstein, N.D.):

$$\text{Skewness coefficient} = \frac{(\text{mean} - \text{median})}{\text{standard deviation}}$$

The results of the analysis are shown in table 3.

Input	Analysis	Analysis output			
		Tick 50	Tick 1050	Tick 3050	Tick 5050
Reference	Variance	0	6.14	24.47	24.98
	Skewness	-	1.31	0.16	0.16
Average links to 6	Variance	0	1.79	4.53	4.44
	Skewness	-	-0.13	0.24	0.18
Average links to 2	Variance	0	32.28	158.18	230.69
	Skewness	-	0.05	0.06	0.07
Ev car life 18	Variance	0	5.78	91.92	105.77
	Skewness	-	0.28	0.11	-0.05

Table 3: Analysis output

The most important conclusions are that in the reference case, the skewness is quite big (and positive) directly after the EV invention, what indicates that most values are on the left of the mean, but some extreme values are on the right. The exact implication on the model is not directly clear, it seems that 4 links per car owner on average is quite a critical number. With fewer links, EV diffusion barely comes from the ground, while with 2 links the diffusion is quite significant. Based on this reasoning, the skewness of the reference case is not found in contradiction with the model specification.

Another notion is the difference in variance in the situation where car owners have two links on average for the ticks 3050 and 5050. Although the total number of EV owners barely changes, the variance gets 1.5 times bigger. However, the underlying skewness is barely different. Hence, the underlying distribution appears to be similar. To investigate this further, one could look at the Kurtosis, an indicator used in distribution analysis as a sign of flattening or 'peakedness' in a distribution. However, this is not done here.

VERIFICATION OF INPUT DISTRIBUTIONS

While experimenting with the model, it seemed that the defined random distribution in the model code do not match the actual distributions in the model setup. Let us specify this further for the case of the variable 'attitude-to-innovation', as this variable is the only random variable for which this notion imposes a restriction on the model experiments.

Traditionally, diffusion literature sees diffusion as an s-curve in a closed system with, in the case of social systems, a variety of types of people, as is shown in figure 6. In a broader sense, these types of people would mean a number of phases.

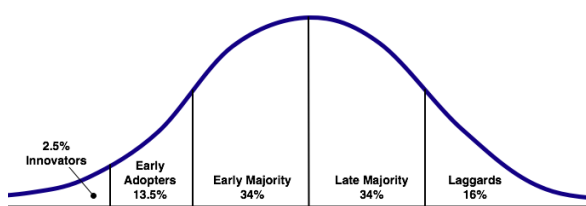


figure 6: diffusion 'phases'

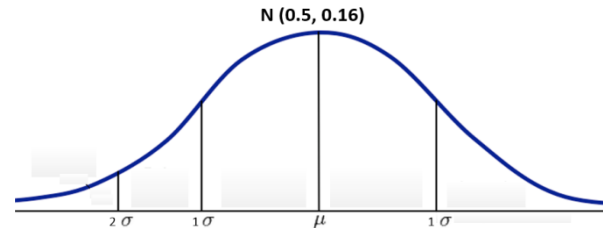


figure 5: supposed attitude to innovation distribution

Following figure 5, we expect a curve on a normal distribution with the average of .5 and standard deviation .16 to follow the same distribution. In terms of the model, when we use 750 car owners, we would expect about 2.5% of them (about 19 car owners) to be innovator. However, figure 7 shows a bar graph on the expected and real variation in adopter categories after on the average of 25 model setups.

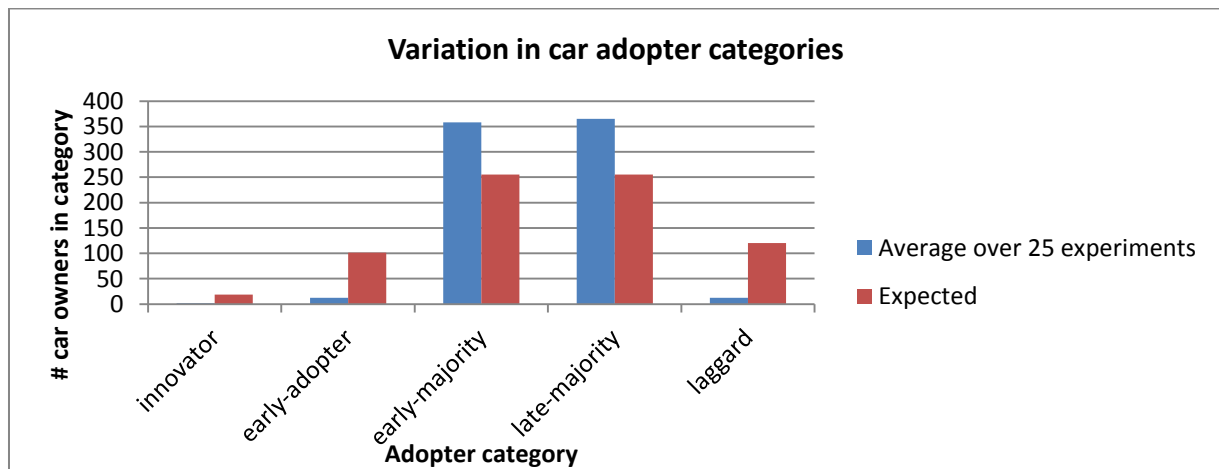


figure 7: average vs expected variation in adopter categories of car owners

On the basis of figure 7 we conclude that the attitude to innovation is not drawn according to the predefined distribution. The average over 25 experiments still seems to follow a normal distribution, but with significant smaller standard deviation. This variation does not directly threat the working of the model. On top of that, adapting the code for this weakness is quite complicated. When the attitude to innovation would only be used for determining the adopter categories, it would make sense to just assign car owners to a specific category. However, the underlying function is also important to determine the threshold value. It is therefore chosen to accept this weakness into the model. Consequences is that EV technology will diffuse less easily, as there are relatively few innovators and early adopters to spread the usage of EVs.

TIMELINE SANITY TEST

In this final test, we check whether model outputs can be explained by reasoning through the model logic. Therefore, we will perform several runs with the default parameter settings to check whether there are any patterns that cannot be explained. We will therefore use a runtime of 10.000 ticks.

Default parameter settings

We will use the following parameter settings:

- Number of car owners: 750
- Average links per car owner: 3
- Years to EV invention: 100
- Initial diesel percentage: 50%
- Innovator memory: 0.1 year
- Early-adopter memory: 2 years
- Early majority memory: 4 years
- Late majority memory: 6 years
- Laggard memory: 8 years
- Average petrol car life: 12 years
- Average diesel car life: 12 years
- Average EV car life: 15 years

Analysis results

We performed 9 replications to check for behavior. Nine graphs with the amount of car types are shown in figure 8. What we notice is that the S-curve starts off relatively quickly and ends relatively slowly, in comparison to diffusion theory. However, this is not imposing a threat to the usability of the model. For the rest of the model run, the behavior seems to be in line with the expected results.

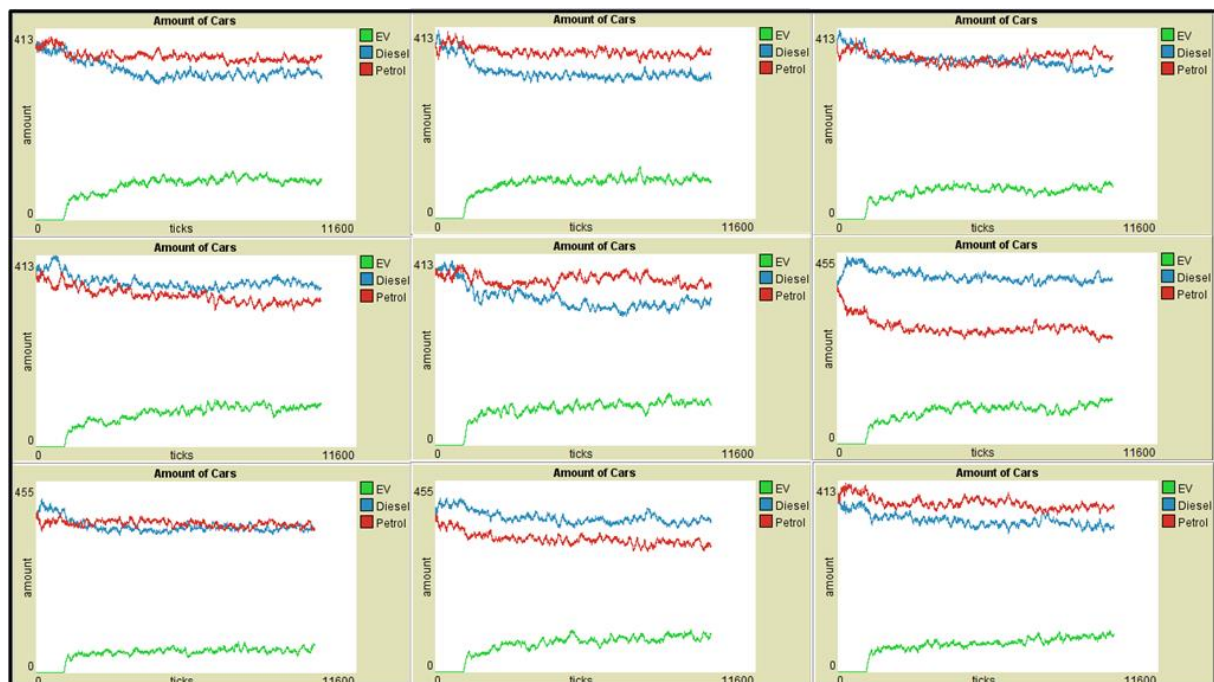


figure 8: Timeline sanity check with 10.000 ticks (9 replications)

As final verification, we will also check the modeling results for an extreme runtime of 100.000 ticks. Note that this is just a single, but illustrative, replication. The results of this test are depicted in figure 9. Again, no significant difference is found from the expected behavior.

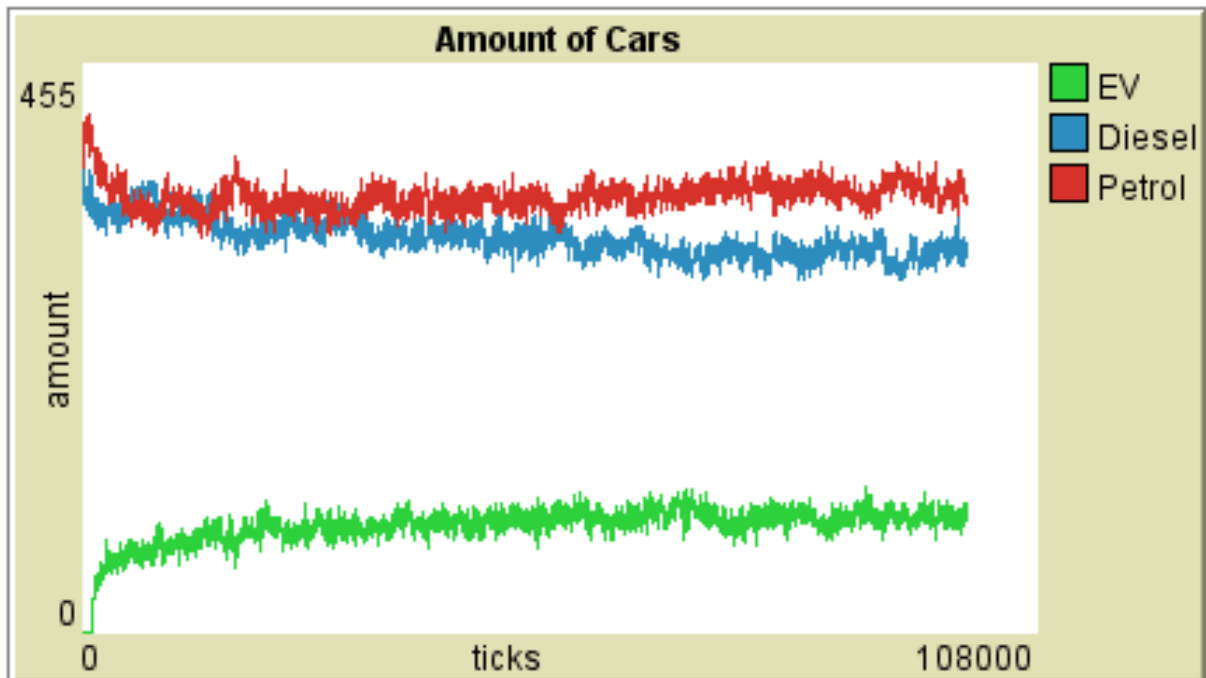


figure 9: long verification run (100000 ticks), note this is only 1 (but illustrative) replication

EXPERIMENTATION

For creating the experimental design, we first discuss the time over which we will run our model. Secondly comes the parameter space that we are interested in. Third, we will say something about the number of repetitions to perform. Fourth, we discuss the experimental setup. With that, also comes a short calculation on the expected run time based. For determining the experimental setup, we use the provided guide to setup an experimental setup in NetLogo (Kasmire, 2011).

EXPERIMENTAL DESIGN: MODEL HYPOTHESIS & RUN TIME

In this study, time is ill defined. This means that we not necessarily know how long the model takes before interesting emergent behavior can be observed, and when it is no longer relevant to study the behavior further (i.e. determine the end of a model run). In fact, the model built just has to function to provide us input for the analysis. The data analysis should help us falsify/confirm our hypothesis: *'The general perspective and the Prigogine perspective exclude each other in identification of stability'*. Therefore, we need to study the transition, and the equilibrium phase after that.

In the verification, we already studied the model behavior over long run time. On the basis of that experience, we here decide that we run a model for 6000 ticks. In this way, we are able to study both the transition, but also the equilibrium phase after that. Depending on the computer we use, a model run takes about 45 seconds on average on a desktop computer.

EXPERIMENTAL DESIGN: PARAMETER SPACE

As the output of this experiment does not have a direct impact on practice, extensively exploring the parameter space (i.e. the scenario space) is of less interest. Instead, a number of variations over the parameters may help us bring a variation of different s-curves. These can then be used for testing the initial hypothesis.

In the verification we already saw that the average number of links of a car owner can strongly influence the implementation of EVs. Hence we will vary our model over the use with 2, 4, and 6 links per car owner on average.

Second we will vary the average car life of EVs as also this proved to have an effect on the diffusion pattern in the verification. We will therefore vary the average car life of EVs between 12, 15 and 18 years, while the average car life of the other car types remain the same (12 years).

Thirdly, we vary with the amount of initial diesel owners. We mainly do this because we wonder whether diffusion patterns differ when two other technologies 1) share the market equally, or 2) compete whereby one is technology possesses a significant bigger extent of the market. For that reason, we vary between the initial diesel percentage being 50% and 29%, the latter representing the car types on the Dutch road (CBS, 2012).

Overall, this gives 18 possible combinations ($3 \cdot 3 \cdot 2$). We will use this information in the next paragraph to discuss the size (especially the run time) of the total experiment considering a number of repetitions.

EXPERIMENTAL DESIGN: REPLICATIONS

Perform a single run for each of the 18 experiments does not give a result that can be trusted, as ABM models use randomness. The question therefore is how many repeats we need to get a statistically reliable sample for a certain confidence level. There are three factors that determine the repeats necessary: desired confidence level, confidence interval and population. In general, more replications is better. But we are limited by computational constraints. Apart from that, we also have to deal with computational constraints when analyzing the data. Considering the more replications the better, we start of planning to do 100 repetitions for each experiment and see whether that fits the computation power available.

EXPERIMENTAL SETUP: RANDOMNESS AND RUN TIME

Ideally, we would run a full factorial design to get the best big picture look at the overall experiment. If this is not possible, we might use Latin Hypercube Sampling for reducing the number of experiments. If even this proves hard in terms of computational memory, we might want to use Monte Carlo techniques.

However, using a full factorial for 18 unique experiments with 100 replications, the total amount of experiments would be 1800. Considering the fact that an average run of the model takes about 45 seconds, and we have a quad core computer available at TPM, running the overall experiment will take 5,5 to 6 hours. However, as we experienced ourselves, the TPM computer room is quite busy during examination periods and the power is killed at night. Hence, we spread the experiments over three quad core computers, which reduces the runtime to under 2 hours. In this way, we are able to run the model in the TPM computer room in the early evening.

EXPERIMENTAL SETUP: METRICS

Now the question remains on what we are actually going to measure. To compare the two perspectives on stability, we need to use other metrics than those traditionally used in stability theory. Best would be use a kind of 'metametric'. However, these are not readily available. Anyway, considering the point attempted to be made in this document,

Hence, in the following section we define the metric used for analysis in some more depth. For now, it is sufficient to mention that the model output for each tick of each run should be the amount of EVs, the amount of petrol cars, and the amount of diesel cars.

EXPERIMENT EXECUTION: CHECKING DATA CONSISTENCY

To check whether we have the data intended we perform a short check on the three comma separated value files that we retrieve from the experiments. Each file should have 3.600.600 rows of data ($6 \times 100 \times 6001$), excluding the information rows on top (this proved to be 6 and a header row makes 7 rows) of each csv file. As this checked out, we verified that we have all the data.

FINDING AND USING A 'METAMETRIC'

As was already discussed in the introduction and problem identification, traditional metrics on diffusion patterns (and the stability of the diffusion) mostly focus on an amount or 'market share' of a characteristic. In these traditional s-curved graphs, it is relatively easy to see the difference between the different experiments. In this section we search for some type of 'metametric' to see whether the perspectives exclude each other in terms of stability. In order to do that, we wanted make a deepened analysis on the output of the metrics that are generally used in diffusion studies. In other words, we want to use a metric that is directly based on the traditional metrics, while the outcome of the analysis with the metric is (/can) not be derived directly from the traditional metrics. Therefore, we'll first take a closer look on what stability actually is. Second, we introduce compression as a metric concept before introducing DEFLATE as the compression algorithm and metric in the third paragraph. The fourth paragraph will explain on how we used compression on the NetLogo output. The fifth and final paragraph will then explain the PHP code used to determine the Kolmogorov complexity.

WHAT IS STABILITY?

Stability is a central term in multiple academic disciplines, like physics, chemistry, geomorphology, ecology, economics, game and decision theory, international affairs and political science (Hansson & Helgesson, 2003). As there are numerous concepts in circulation, several attempts have been made to clarify the concepts (Boulding, 1978; Grofman & Uhlner, 1985; McCoy & Shrader-Frechette, 1992; Clausen, 1998). Based on a literature study by Hansson & Helgesson (2003) over various fields, three basic concepts surrounding stability can be defined: constancy, robustness, and resilience. In diffusion studies, constancy seems to be the main concept used for naming stability. Hence, we consider this concept as our basic starting point.

Constancy, is what is the meaning that is often found when dictionaries give synonyms for stability (i.e. stationary, not changing, and constant). Constancy is 'whenever something remains unchanged or is changed only to a limited degree' (Hansson & Helgesson, 2003). Hence, constancy seems to be related to the terms predictability and complexity.

INTRODUCING COMPRESSION AS STABILITY METRIC CONCEPT

Considering the idea that stability, seems to be related to the notions of predictability and stability, we might find an appropriate metric for the analysis in these fields. The field algorithmic information theory, focusses on how much computational resources are needed to specify an object. One of the key concepts here is Kolmogorov complexity and data compression. The Kolmogorov complexity gives a number on the complexity of a string, the complexity of a string is the length of the shortest possible description of the string in some fixed language.

There are two types of data compression: lossy and lossless. Lossy data compression is an encoding method that compresses data by losing some of it. The focus lies on minimizing the amount of data needed. Typically, quite a significant amount of data can be discarded before the degradation has a significant effect. Hence, lossy compression is often used for compression of multimedia data. Think for example of JPEG and MP3. Lossless compression allows for the exact original data to be reconstructed from the compressed data. Hence, this type of compression is often used for text and data files. A well-known method here is ZIP.

Lossless compression techniques provides an ideal way to measure stability in terms of predictability and complexity as it does not lose any of the information in between.

INTRODUCING DEFLATE AS STABILITY METRIC

As already said, ZIP is a commonly used lossless compression type. Igor Nikolic raised the idea of compressing parts of data to a ZIP file, read out the sizes of the files, and use the file sizes as indication for the complexity. However, on windows operating machines, this is not as trivial as it looks. Therefore, we decided to search further.

ZIP (but also PNG for example) is based on Deflate, a lossless data compression algorithm developed by Phil Katz. Deflate itself is based on Huffman coding and LZ77, two entropy coding algorithms. As Deflate is just one of the many algorithms around, we will not extensively discuss the pros and cons, but just mention the biggest pro: The Deflate algorithm is standard available in the PHP! As such, this removes all the barrier encountered by running on a windows operating machine. However, it still required us to learn PHP.

We now have a metric, i.e. the Kolmogorov complexity, that can be used to analyze the variety in stability perspectives. Kolmogorov complexity in our case defines the length of the string (in bytes) of the shortest possible description of the string in the DEFLATE language.

However, before we go into detail on the PHP coding, let us first explore on what we actually want to measure.

HOW TO DO COMPRESSION ANALYSIS ON NETLOGO OUTPUT

The NetLogo output consists of a CSV file that includes the input variable values and the output metrics for each run on each tick. What we have to do with this data for compression is to determine which part of the curves we will compress. This is not a trivial thing, as the elements of the curve, i.e. the start and length of the diffusion as well as the 'stable' phase thereafter, cannot easily be determined. In addition, according to Kolmogorov theory, the size of the compressed string (i.e. the Kolmogorov complexity) is strongly dependent on the length of the string.

To solve these two difficulties, we will divide each output string in a number of blocks (i.e. a certain amount of ticks) and define a step size (\leq block size) for a new block to start. In this way, we create some overlap over the blocks. Advantage of this approach is that every block has the same size (in terms of number of measurements) and blocks are therefore comparable. We can now also use a number of blocks to compare.

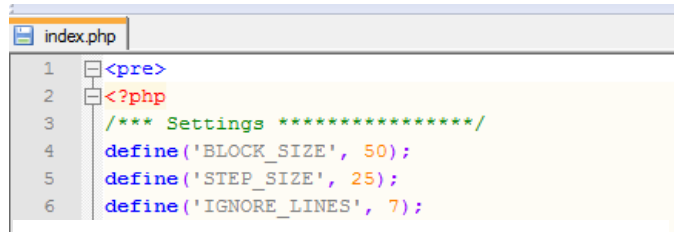
Note that we will pack the data per (subelement of a) run, and not over a number of runs, for reasons of comparability. Initially we will use a block size of 50 ticks and a step size of 25 ticks. This is quite an arbitrary choice that balances on 2 ideas. First of all, we want to limit the block size to get as much information as possible. On the other hand we want to have a bigger block to be able to measure the compression effect better. As this choice is relatively arbitrary, in the analysis phase we will also pay some attention to the effect of the block size and step size on the analysis outcomes.

The question now remains on what we are actually going to measure. We will use the traditional diffusion metrics for two types of analysis. First we will analyze the stability perspectives on the implementation of the innovation (i.e. the introduction of EVs). Literature typically considers this 'individual' happening and describes it as an event on an overall system. However, literature does rarely pay attention to the effect of other elements in a system. Still, the system comprises all of the elements. Therefore we will also do a second analysis on the compressibility of the overall system. For this analysis, we will compress the amount of EVs, petrol cars and diesel cars separately, and then sum up their separate Kolmogorov complexities to an overall complexity number of the whole environment. In this way, we are able to explore the stability perspectives on a system level during the diffusion.

PHP CODE FOR DETERMINING KOLMOGOROV COMPLEXITY

In this section, we will explain the PHP code used. The code is written in Notepad ++ and executed by simulating a server using XAMPP. In XAMPP, the memory limit is put to 3072 M and the maximum server execution time is put to 3000s (this is done in the XAMPP control panel by selecting the Apache configuration and the php.ini file). The PHP code consists of four main elements that we will go through in this paragraph.

First, we start by defining the block and step size that we defined before, and the number of lines that can be ignored. The latter is there to exclude the standard information given in the NetLogo output, i.e. the code should only include the actual data in its analysis. This part of the code is shown in figure 10.



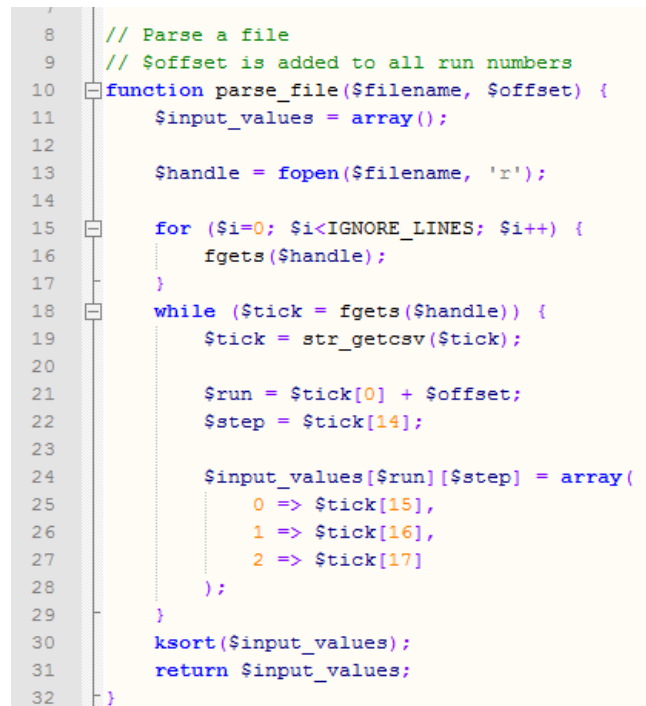
```

1 <pre>
2 <?php
3 /** Settings *****/
4 define('BLOCK_SIZE', 50);
5 define('STEP_SIZE', 25);
6 define('IGNORE_LINES', 7);

```

figure 10: Definition of elements

In the second element, we open the CSV file and select the relevant information. As we performed multiple runs at the same time (i.e. using quad core computers), we also sort the data according to run number and tick number. This element of the code is shown in figure 11.



```

8 // Parse a file
9 // $offset is added to all run numbers
10 function parse_file($filename, $offset) {
11     $input_values = array();
12
13     $handle = fopen($filename, 'r');
14
15     for ($i=0; $i<IGNORE_LINES; $i++) {
16         fgets($handle);
17     }
18     while ($tick = fgets($handle)) {
19         $tick = str_getcsv($tick);
20
21         $run = $tick[0] + $offset;
22         $step = $tick[14];
23
24         $input_values[$run][$step] = array(
25             0 => $tick[15],
26             1 => $tick[16],
27             2 => $tick[17]
28         );
29     }
30     ksort($input_values);
31     return $input_values;
32 }

```

figure 11: Select and sort data

The third part of the code defines the compressions that have to be made. This starts with defining that each run should be processed separately, i.e. no blocks from different runs should be used. Subsequently, substrings of the initial output are defined. The number of substrings depends on the step size. There are three types of substrings, one for each car type (i.e. 'string_e', 'string_p', and 'string_d'). The length of the substring depends on the block size.

The substrings are then compressed using the deflate algorithm. The length of compression file (in bytes) is then saved as a new value, together with run and ticks that define the block. This part of the code is shown in figure 12.

```

34 // Calculate output for given input values and write to output file
35 function calculate_output($input_values) {
36     foreach ($input_values as $run => $run_values) {
37         $ticks = sizeof($run_values);
38
39         for ($i = 0; $i < $ticks; $i += STEP_SIZE) {
40             $string_e = '';
41             $string_p = '';
42             $string_d = '';
43
44             for ($j = $i; $j < $i+BLOCK_SIZE; $j++) {
45                 if (isset($run_values[$j][0])) {
46                     $string_e .= $run_values[$j][0].' ';
47                     $string_p .= $run_values[$j][1].' ';
48                     $string_d .= $run_values[$j][2].' ';
49                     $last = $j;
50                 }
51             }
52             if ($last == ($i + BLOCK_SIZE - 1)) {
53                 $output_line = "".$run."".$i."-".$last."".$strlen(gzdeflate($string_e)).".".$strlen(gzdeflate($string_p)).".".$strlen(gzdeflate($string_d)).".".$n";
54             }
55             file_put_contents('output.csv', $output_line, FILE_APPEND);
56         }
57     }
58 }

```

figure 12: definition and compression of substrings

The fourth and final part of the code initiates the procedures. It opens a new output CSV file (and defines the headers), and it initiates the definition and compression of all substrings for all 1800 runs based on the three different CSV input files (as we used 3 computers for our experiment). In defining the input files, a value is summed up with the run number (either 0, 600, or 1200). This is done as each input file holds the run numbers 1 to 600, but we would like to be able to distinguish the various runs again afterward. The fourth part of the code is shown in figure 13.

```

59 // Initialize file
60 file_put_contents('output.csv', '"run","string","size EV","size Petrol","size Diesel"."\n');
61
62 // Parse input files and calculate output
63 calculate_output(parse_file('Experiment_table_2.csv', 0));
64 calculate_output(parse_file('Experimenten_table_4.csv', 600));
65 calculate_output(parse_file('Experiment_table_6.csv', 1200));
66
67 echo 'Done!';
68
69 ?>
70 </pre>

```

figure 13: code procedure

After defining this code there is one critical notion we have to make on our approach. In this analysis, we use the total amount of cars per car type as measure for analysis. However, it is not sure what the effect is of compressing a string of [2 3 5 7] against [12 13 15 17] for example. It could just be the latter can be less compressed as it requires more digits. Therefore our outcomes might be slightly biased. A suggestion to improve this might be to work out of percentages with a fixed number of digits. This is however not done in this report as this idea originated after doing the actual analysis.

DATA ANALYSIS

This chapter discusses and analyzes the output of the model. The aim is to analyze and describe the results in such a way that the main question can be answered. First the initial output will be shown, consisting of the total amount of EV owners. This output is shown since it should visualize the desired s-curve, the traditional metrics on stability are extracted from this output. In addition it is the starting point for the kolmogorov complexity analysis. Subsequently, some analyses will be executed regarding the kolmogorov complexity, in order to answer the main question.

The sections in this chapter are using the following structure. First an exploration of the data is described, in order to get understanding of the output. Subsequently, if useful, the data will be visualized more specifically in order to identify relevant patterns for analysis. Thereafter, the identified pattern will be interpreted and explained. If useful, the appendices are used to see the pattern over multiple runs and these conclusions are included in the this chapter.

To be able to reconstruct the analysis, the process and the used code are important. However, when this would be included in the discussion of this chapter, it would be really hard to keep overview on what is happening. So, for the sake of clarity, the used codes and the way data is reworked is added in the appendices (G, H, and I). The way of reworking data regarding kolmogorov complexity is already discussed in the previous section.

INITIAL OUTPUT

The exploration of the initial output entails two different types of graphs. A single run graph, that shows the amount of EV owners over the time and a 3D graph showing the amount of EV owners over the time for all runs.

The single run graph is made from the first run and shown in figure 14. This graph shows a pretty nice s-curve that starts after tick 1000, when the EV implementation starts. Since the analysis compares the general perspective with the Prigogine perspective it is useful see how those two perspectives are reflected in this graph. Figure 14 shows two rectangles. The left rectangle shows the phase of stability according to the Prigogine perspective and is called the linear part of the s-curve, where the right rectangle shows the stability phase according to the general perspective and is called the end part of the s-curve. The color scale in figure 14, is scaled, based on the amount of EV owners.

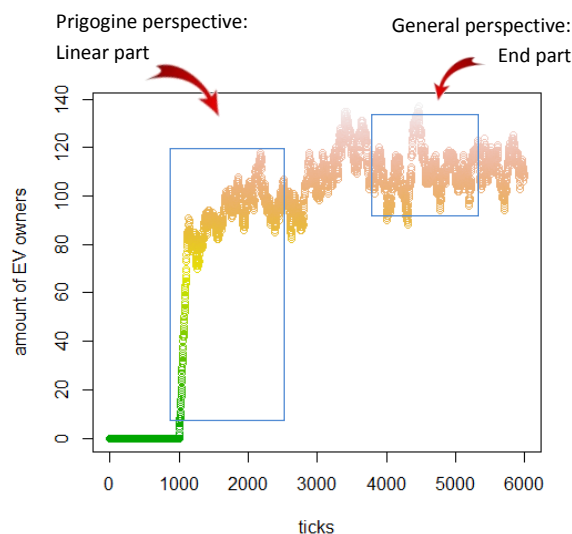


figure 14: Amount of EV owners in a sample run

The 3D graph (figures 15, 16, and 17) shows the same information of the single graphs, only for all the runs side by side. Basically it comes down to an added axis including the runnumber, which are 1800 in total. The 6000 ticks are the same, i.e. the time horizon within a run and the number of turtles that own an EV are also again shown.

A few things can be concluded from these 3D graphs. First of all, the amount of turtles that owns an EV is highly related to the runnumber. The runnumbers 0-600 have resulting in the highest total amount of EV owners, followed by the runnumbers 601-1200 and runnumbers 1201-1800. These runs differ on the variable on average links with neighbors 0-600 = average 2 links, 601 – 1200 = average 4 links, and 1201-1800 = average 6 links.

In addition, these graphs show that the s-curve is not everywhere that nice as shown in the graphs of the single run. Based on these graphs there is doubted whether the s-curve actually exists in the runs with a small total amount of EV owners, since the runs 1201- 1800 are showing almost no fluctuation at all. Therefore the maximum amount of EV owners is calculated over each tick over all those 600 runs. The result shows that the maximum amount of EV owners is 23 in these 600 runs, versus 352 within the runs 0 – 600 and 81 within the runs 601 – 1200. Based on this information is decided to exclude the runs 1200-1800 in the pattern analysis, since the desired behavior doesn't exist in these runs.

Furthermore, the behavior of all the graphs is equal around tick 1000. Right after the implementation the amount of EV owners' increases and the linear part of the s-curves starts.

To conclude this exploration, the runs 0-1200 showing behavior according the desired behavior and are therefore useful to analyze. Furthermore, the linear part of the s-curve appears to start in all runs right after the 1000 ticks. Since the focus of this analysis will be on the kolmogorov complexity, there is no need for further pattern identification in this part of the analysis.

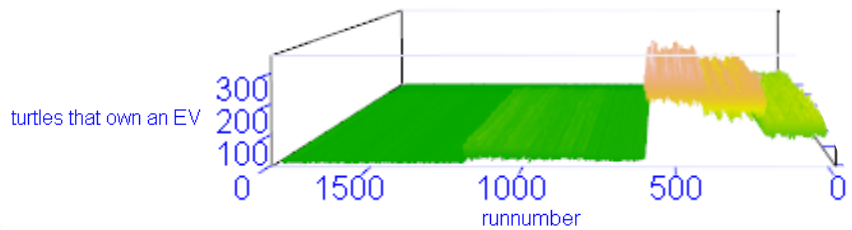


Figure 15 – front view, runnumbers vs. amount of turtles owning EV

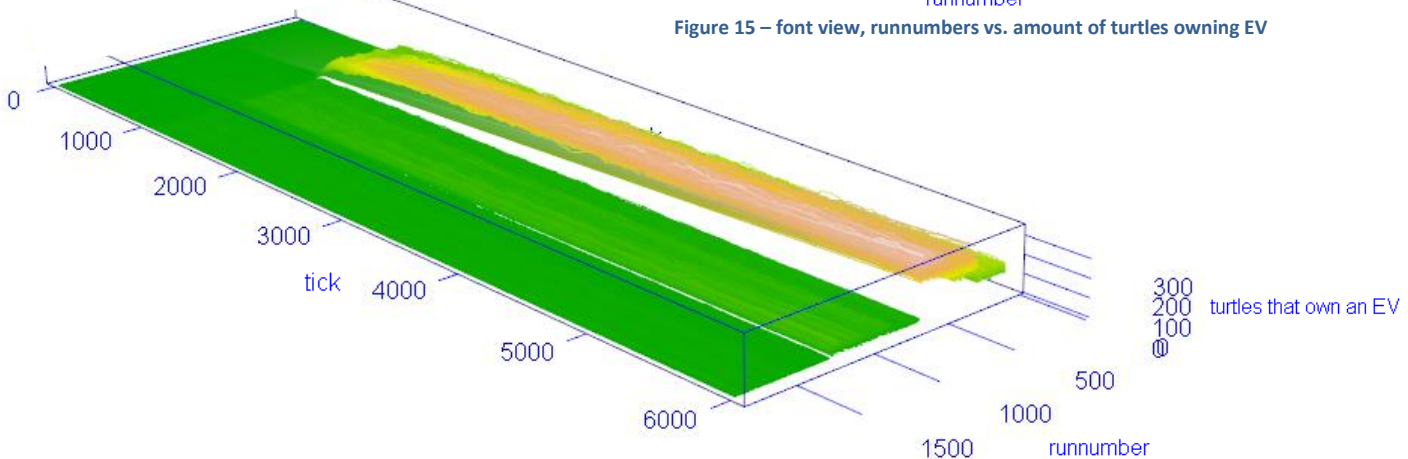


Figure 16 – helicopter view, runnumbers vs. amount of turtles owning EV vs. ticks

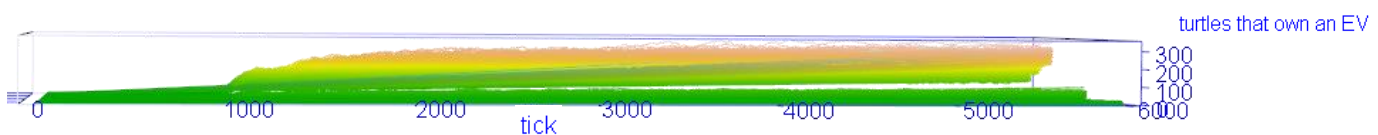


Figure 17 – side view, ticks vs. amount of turtles owning EV vs. ticks

KOLMOGOROV COMPLEXITY

As discussed in the previous section, this analysis uses kolmogorov complexity as starting point. Since the data is reworked in order to analyze it, this section will start with a small summary of the reworked data. This also entails which strings will be used in the analysis as linear part and as end part of the s-curve. Subsequently three analysis will be executed, respectively the comparison between linear and end part of compression within the EV string, the comparison between linear and end part of compression over the whole system and the influence of the chosen step- and blocksize.

SUMMARY REWORKED DATA

The data that is reworked with PHP, notepad ++ and excel is summarized in table 4 below, as discussed in the previous section. This data uses a stepsize of 25 ticks and a blocksize for each step of 50 ticks. The empty columns of compressed values contain of course the specific compressed value for each string, for each run.

number of strings per run **241**
number of replications **100**

Runset	Runnumber	startrow	end row	variabele			Compressed values		
				Neighbors	EV carlife	Initial diesel%	EV compression value	Petrol compression value	Diesel compression value
1	1-100	1	24100	2	12	29			
2	101-200	24101	48200	2	12	50			
3	201-300	48201	72300	2	15	29			
4	301-400	72301	96400	2	15	50			
5	401-500	96401	120500	2	18	29			
6	501-600	120501	144600	2	18	50			
7	601-700	144601	168700	4	12	29			
8	701-800	168701	192800	4	12	50			
9	801-900	192801	216900	4	15	29			
10	901-1000	216901	241000	4	15	50			
11	1001-1100	241001	265100	4	18	29			
12	1101-1200	265101	289200	4	18	50			
13	1201-1300	289201	313300	6	12	29			
14	1301-1400	313301	337400	6	12	50			
15	1401-1500	337401	361500	6	15	29			
16	1501-1600	361501	385600	6	15	50			
17	1601-1700	385601	409700	6	18	29			
18	1701-1800	409701	433800	6	18	50			

Table 3: identification of runsets

The previous paragraph concluded that the linear part of the s-curve starts right after the 1000 ticks. Based on this information the strings that will be used to use as reflection of the linear part of the s-curve are defined. Tick 1000-1050 is equal to string forty. Therefore the linear part will be represented by string 40 till string 60 (tick 1525). We are aware that using twenty strings is an relative arbitrary choice. However, considering the exploratory analysis, the main linear part of the diffusion curve occurs in the first 500 ticks after the EV invention.

The end part hasn't a specific starting point. Therefore, some strings that are relative late in the run are included to represent this end part. This have become string 150 – 170, which are covers the ticks 3725 to 4250. Again, these are twenty ticks. That makes that a same number of compressed values are compared.

Furthermore, it is important that only strings within runsets can be compared. The exploration of the previous section already shows that the runsets differ a lot. In addition, if the strings are compared over the borders of the runsets, different model settings are compared with each other. This would be difficult to interpret, aiming at making a statement about stability.

RESULTS

The part of the analysis chapter discusses different analyses that are executed in order to make a statement about stability using kolmogorov complexity as a metric. First of all the compression of the amount of EVs will be discussed. Subsequently, the combination is made between the three different types of car, in order to involving the aspect of system stability instead of stability from a perspective of new technology. This analysis is followed by an analysis on the impact of the block- and step size.

COMPARISON ON DEGREE OF COMPRESSION OF AMOUNT OF EVS

This first paragraph discusses the degree of compression of the amount of EVs within a block of the linear part compared to the end part. Though, first an overview of the results of the compression is depicted in the 3D graphs (figure 18, 19 and 20) below, with the aim to get understanding of the used axes in the graphs. These graphs depict for all 1800 runs (x- axis) the size of the compression (on the z-axis) for all 241 strings (y-axis) which is the transformed time axis per run.



Figure 18 – side view, compression size vs. runnumber

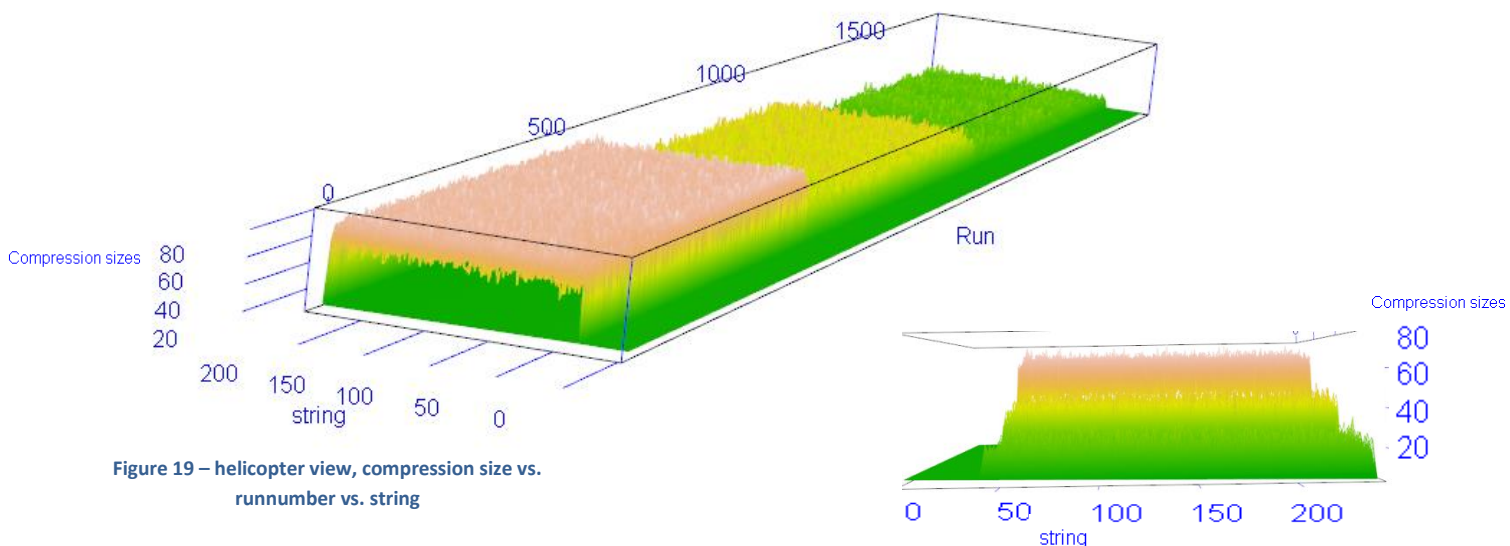


Figure 19 – helicopter view, compression size vs. runnumber vs. string

Figure 20 – front view compression size vs. string

These graphs show again a clear distinction in the runs 0-600, 601-1200 and 1201 – 1800, whereby the runs between 0 and 600 have the highest values on the size of compression. This means that these runs are less compressed than strings in run 601-1200 and therefore resulting in a higher value. In addition, the graphs show very clear that all the 1800 runs have a same level of compression until the 40th string. This is obvious since the implementation of the technology isn't started before that run, which results in same string length for each run.

Although these graphs provide some sense of the output that will be analyzed, it is difficult to conclude something about the linear and the end part of stability from these graphs. There are no clear distinctions from these graphs over the 1800 different runs between the linear part (starting at string 40) and the end part (starting at string 150).

PATTERN RECOGNITION

After the exploration of the compressed data, there is zoomed into a single runset, the first runset. As every runset, this first runset entails 100 runs that are using the same variable input. The aim of this zoom is to recognize an interesting pattern that could be matter for further analysis. The visualization used for this pattern recognition is a 3D graph, as depicted in figure 21 and 22. Figure 21 does not provide new insights, regarding the previous paragraph. However, figure 22 does. When there circles are drawn around the linear part of the s-curve and around the end part of the s-curve, it becomes clear that the linear part seems to have a smaller compression size than the end part. The lines green dotted lines are added for reference.

However, this observation could be analyzed further. In order to do so, histogram and density curves are used that compare this linear part of the string, with an end part.

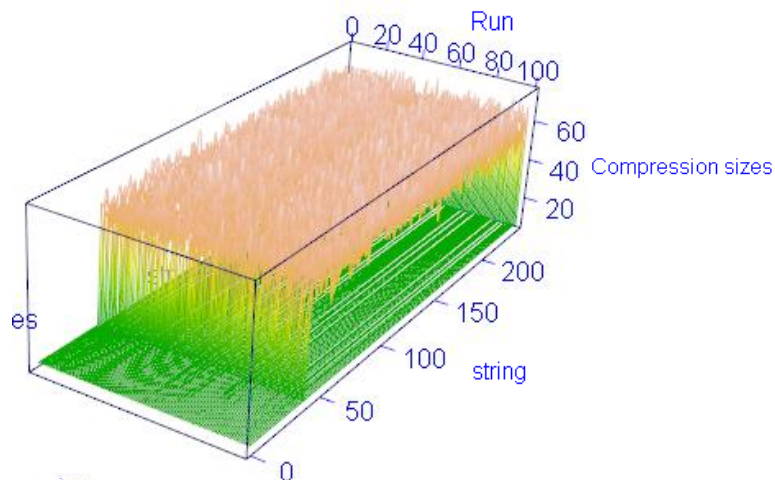


Figure 21 – helicopter view, string vs. compression size vs. run (used data is first runset, i.e. 100 repetitions)

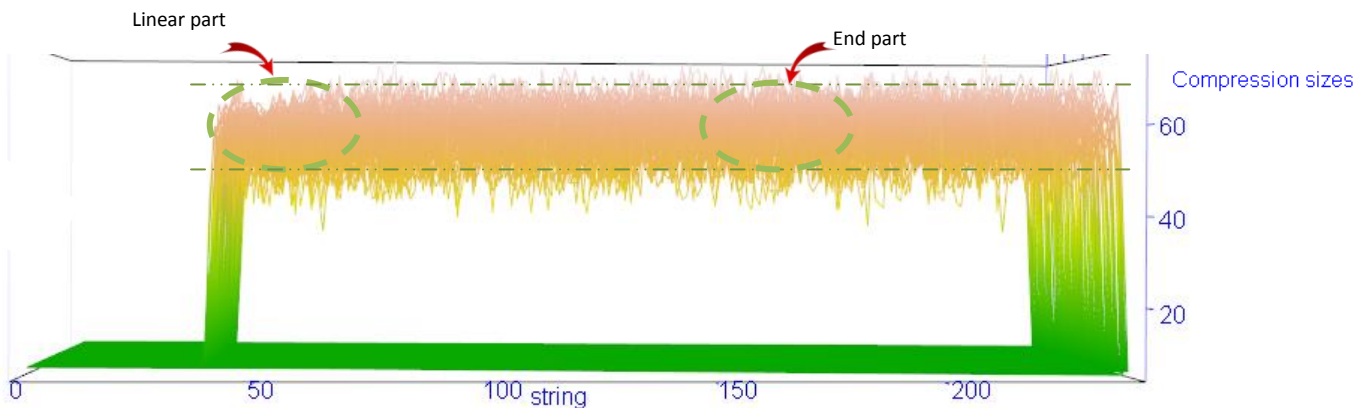


Figure 22 – side view, string vs. compression size (used data is first runset, i.e. 100 repetitions)

INTERPRETATION AND EXPLANATION

The in the previous section recognized pattern is confirmed with the density curve of figure 23. This density curve shows that the peak of the compression sizes of the linear part is at a lower size than the peak of the end part. Although the difference is small, the difference is present. Moreover, this conclusions is valid for almost all the 12 runsets, that are analyzed this analysis, see appendix C (histograms), and appendix D (density).

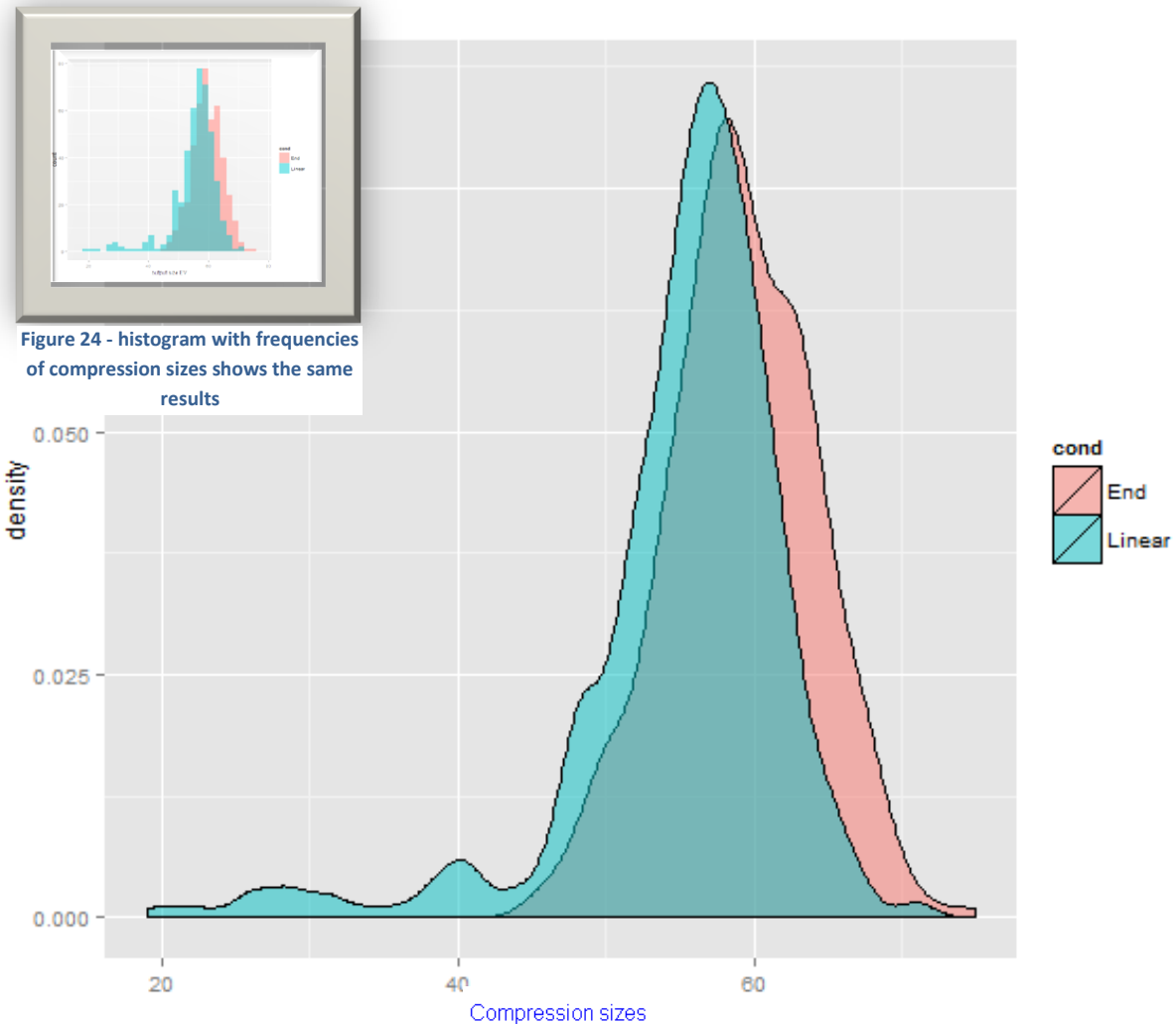


Figure 23 –density curve, showing the distribution of the compressed sizes of the linear part of the model (blue) and the end part of the model (red). (Data used is the first runset, first 20 replications).

Note that only the first twenty replications of the first runset are included in figure 23 and 24, instead of the first hundred replications. This, because the learning curve regards using R statistics wasn't finished yet and some formulating issues arises making this graph.

From this could be concluded that regarding the kolmogorov complexity the linear part of the s-curve is less complex. This strengthens the statement that the linear part of the s-curve is more stable than the end part of the s-curve. Thus, using this metrics, it can be concluded that the two perspectives could exclude each other.

OVERALL COMPRESSION SIZES INCLUDING OTHER SYSTEM ELEMENTS

This sections deals with the complexity whether only one s-curve should be analyzed or also the reflection of this s-curves on the other system elements. Gives looking at all the techniques together another conclusion on which part of the system can be called stable according to kolmogorov complexity. To include all the system elements in one analysis the original data is reworked. First, all the separated amount of cars are put into strings, subsequently they are separately compressed. The size of these compressed files for each type of technology is added to one compressed size for each string for each run.

Again, first an exploration is given by using 3D graphs. The compressions sizes shows the summed compressions of all compressed technologies. From figure 25 and 26 becomes clear that the different runsets are varying a lot, but this doesn't influence the analysis since only within runsets there is compared. Figure 27 shows that the compression sizes the first forty strings is decreasing and thereafter there is seems to be a peak. This will be included in the pattern recognition of the following paragraph.



Figure 25 – side view, compression size vs. runnumber

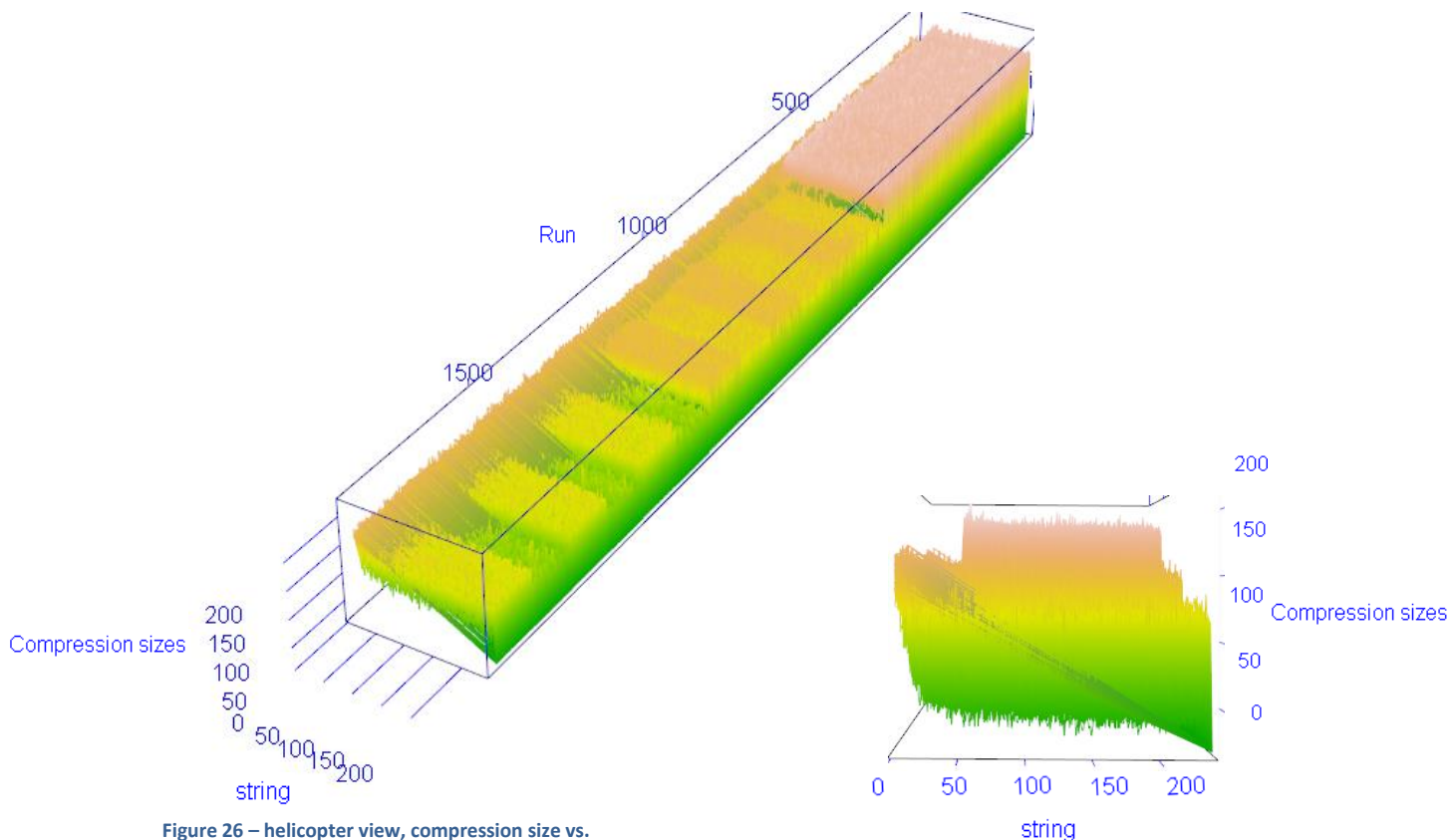


Figure 26 – helicopter view, compression size vs. runnumber vs. string

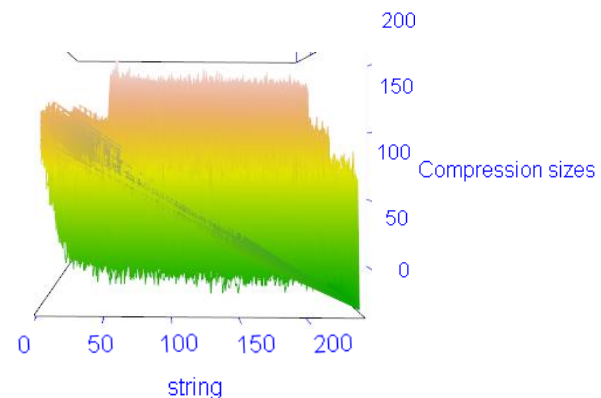


Figure 27 – front view compression size vs. string

PATTERN RECOGNITION

Figure 28 and 29 shows both summed compression sizes of the first 100 replication of the first runset. Figure 29 does not add new insights in patterns that could be recognized. Nevertheless, the helicopter view of figure 28 shows in the circles some color difference between the linear part and the end part. The linear part, as also observed in figure 27, seems to have a peak in the sizes of compression.

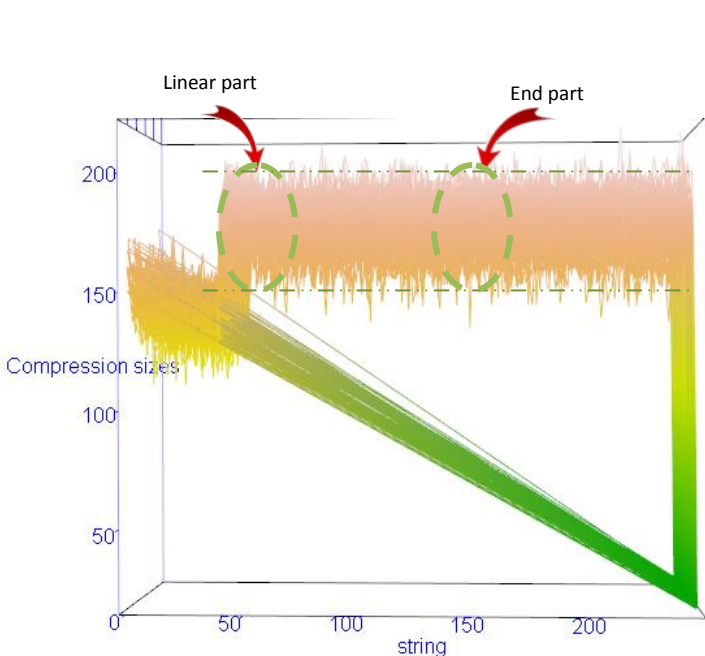


Figure 29 – side view, string vs. compression size (used data is first runset, i.e. 100 repetitions)

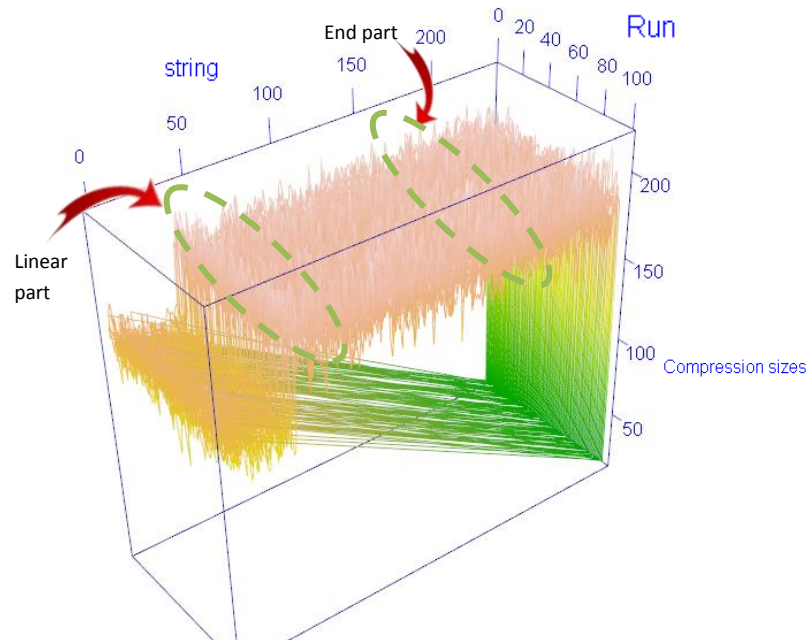


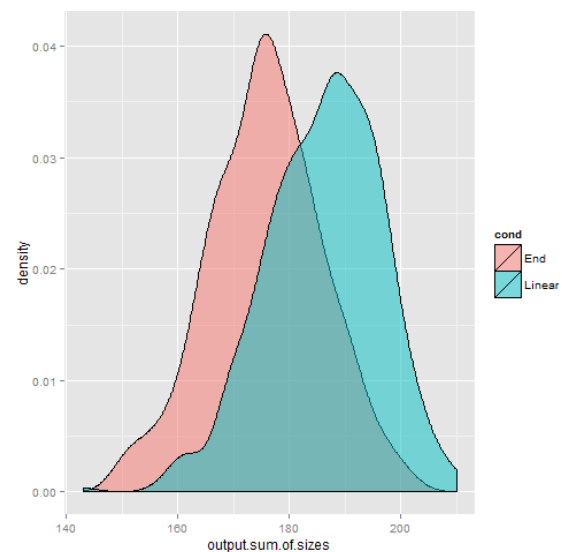
Figure 28 – helicopter view, string vs. compression size vs. run (used data is first runset, i.e. 100 repetitions)

INTERPRETATION AND EXPLANATION

The in the previous section recognized pattern is confirmed with the density curve of figure 30. This density curve shows that the peak of the compression sizes of the end part is at a lower size than the peak of the linear part. The difference is most clear in runset 6, and therefore this runset is depicted in figure 30. Moreover, this conclusions is valid for almost all the 12 runsets, that are analyzed this analysis, see appendix E (histograms), and appendix F (density).

Note that, again, only the first twenty replications of the first runset are included in figure 30, instead of the first hundred replications. This, because the learning curve regards using R statistics wasn't finished yet and some formulating issues arises making this graph.

From this could be concluded that regarding the kolmogorov complexity the end part of the s-curve is less complex. This strengthens the statement that the end part of the s-curve is more stable than the end part of the s-curve, when the whole system is taken into account. Again, the two perspectives on stability seem to exclude each other.



Compression sizes

Figure 30 –density curve, showing the distribution of the compressed sizes of the linear part of the model (blue) and the end part of the model (red). (Data used is the sixth runset (most clear graph), first 20 replications).

VARYING STEP AND BLOCK SIZE

Using the compressed size of the strings has result in some interesting conclusions. Though, the analyses that substantiate these conclusions are based on a block size of 50 ticks and a step size of 25 ticks. These numbers are fairly arbitrary and therefore it is good to analyze how a change in these numbers influences the outcome. To do so, the first 20 replications of the first runset are used.

Table x.x. shows in the middle the used settings for the analysis. Around the middle there is varied with the block size the step size of both. The upper row shows an decreasing of the block size of 40 ticks, while the bottom row shows an increasing of 40 ticks. The row in the middle stays the same. A more or less same structure is used for the step size. The right column shows an increasing step size of 15 ticks, to 40 ticks. The left columns shows a decreasing with the same range of 15 ticks. Again, the row in middle is kept the same. This table shows the variation that will be made, in order to check what the effect of a different block and step size is on the outcome of the model. Two combinations will not be included in this analysis. Those are the two grey fields in the upper row at the right. In both cases the step size is higher than the block size which means that there is data lacking in those two settings. Therefore these two settings doesn't make sense to include in the analysis.

Block 10 Step 10	<i>Block 10 Step 25</i>	<i>Block 10 Step 40</i>
Block 50 Step 10	Block 50 Step 25	Block 50 Step 40
Block 90 Step 10	Block 90 Step 25	Block 90 Step 40

Figure 31, shows the graphs of the different step sizes and block sizes in accordance to the structure shown in table 4. In this figure the 6 density graphs around the central graph should be compared with the central graph.

Table 4: Comparison structure

This comparison shows that an increasing in the block size broadened the bandwidth and sometimes even added an extra peak. In addition, a smaller step size seems to provide more aligned peaks between the end and the linear part. However, the surface appears to differ more in the advantage of the linear part, by using a smaller step size.

Overall can be concluded that a different step size and block size influences the shape of the graph. However, the differences are not to that extend and that different that the chosen level of block size and step size could be used to contradict the conclusions in the previous sections.

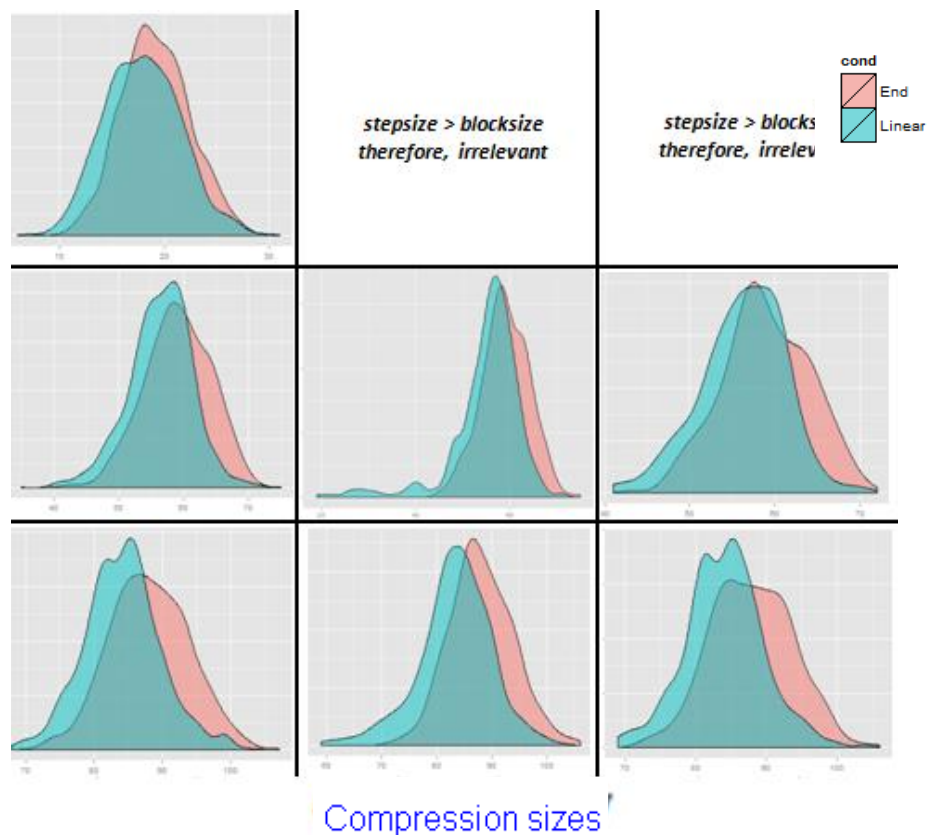


Figure 31 –Density curves with different step sizes and different block sizes as discussed in table 4.

MODEL VALIDATION

Validation of the model concerns the question: 'Did we build the right thing?'. Answering this question is hard, especially for Agent-Based Models. As our model is quite abstract, we will here focus on whether the model is useful and convincing in contributing to the main question. For that, we will discuss four methods used for validation (Dam, Nikolic, & Lukszo, 2012): Historic replay, literature validation, face validation through expert validation, and model replication.

HISTORIC REPLAY

Historic replay is interesting for models that can be compared to a real-world situation. Although it may seem like the analogy used in this model may help for this type of validation, this is actually not the case. Diffusion literature talks about diffusion in closed systems. In reality however, it is very rare that a closed system is measured, especially in a more social setting like the introduction and spreading of EVs. Hence, validating the model through historic replay seems not the way to go.

LITERATURE VALIDATION

Letting go of the analogy, the core of our developed model lies in the production of the s-curves that literature so typically describes for diffusion processes. For a limited number of social links per car owner (on average), smaller than six, the model replicated this type of behavior pretty well. Hence, this contributes to the validity of the model.

When we look to the general outcomes and recommendations of the model, this is, again, much harder to say. Although we confirm our hypothesis that was based on scholarly literature, there are also some question that arise on basis on the outcome. Furthermore, it is hard to say that this will actually contribute to the validity, as the way we approached the analysis has, at least in our knowledge, not been done before.

FACE VALIDATIONS THROUGH EXPERT CONSULTATION

Expert validation is the most commonly used type of validation in ABM. Although there is little room for an extensive ('formal') expert validation in the context of this report, as well as that the subject is so abstract that it significantly reduces the amount of available experts, still some expert validation can be done. Joolie Kasmire is the direct client of this report and also initiated this research. As such, Joolie has a bright and clear idea on the concept of stability in the diffusion literature.

During the development of the model, there have been numerous consultations and discussion on 1) what the model should be doing and 2) whether model outcomes resulted. The model outcomes used in the analysis were classified as 'appears reasonable'. The most important limitations of the model outcomes can most likely be traced back to the variation in adopter categories in the model, an issue already mentioned in the verification. Hence, we consider this statement to contribute to the validity of the model for its purpose, too.

MODEL REPLICATION

Finally, a strong, but intensive form of validation is is model replication: creating a second ABM model with a different decomposition, or a model that uses a different modeling technique, to compare the outcomes (Dam, Nikolic, & Lukszo, 2012). Because of time constraints, this form of validation is unfortunately not possible here. However, it is highly recommended to perform these tests to validate our conclusions in a further research.

MODEL USE

In this section, we will present the outcomes, i.e. conclusions, of the analysis in relation to knowledge gaps identified in the beginning. Secondly, we will identify a number of new questions that arose during this modeling cycle. As in our modeling process, the long term stakeholder engagement (with the model) is less relevant, we will not describe this here. What we will do, as third and final step in this section, is step out of the modeling cycle to discuss (our learning curve during) this modeling cycle and model limitations (and the course).

CONCLUSIONS

The hypothesis defined during the problem identification was:

‘The general perspective and the Prigogine perspective exclude each other in identification of stability.’

To test the hypothesis, we build an agent-based model using the analogy of a social network and monitored the introduction of a new technology. In evaluating the effect of the introduction on the stability perspectives, a kind of ‘metametric’ was developed. As ‘metametric’, we choose to look at the extent to which the outcomes of standard metrics in innovation studies could be compressed, as literature seems to link the concepts of stability, complexity and predictability. Furthermore, we analyzed the results for the introduction itself, and for the effect on the overall system.

The results of the analysis are surprising. Considering the introduction of a technology, the phase of introduction seems to be less complex (i.e. more stable) than the equilibrium phase following. However, when we look from a system level, and incorporate all technologies, the phase of introduction seems to be more complex (i.e. less stable) than the equilibrium phase following. In any way, the general perspective and the Prigogine perspective seem to exclude each other in the identification of stability. Hence, we confirm the hypothesis.

Critical note that has to be made is that we just considered a single alternative type of measuring stability, using the standard metrics of innovation studies. One might think of other possible ‘metametrics’ using the standard metrics, that would falsify the hypothesis.

RECOMMENDATIONS

As was already stated, the analysis results were somewhat surprising. In fact, one could argue that stability is just a matter of perspective. One could look at a system from different viewpoints, thereby ‘experiencing’ stability according to different stability perspectives. This might then imply that there is some kind of spectrum between the general and the Prigogine perspective of stability.

In diffusion literature, stability often seems to be considered as a matter of ‘constancy’. As we saw, in many other fields stability is also connected to terms of robustness and resilience. Therefore, three subjects for further research can be identified:

- The potential existence of a spectrum between the general perspective and the Prigogine perspective on stability
- The relation of the two perspectives on stability in respect to the concepts of robustness and resilience
- The relation of stability from looking on the ‘part of the system’ level vs. looking from a ‘full system’ level.

DISCUSSION ON MODEL LIMITATIONS

The model is made for a really specific purpose. One should be aware of that when using the model. This entails that the model cannot be used for other purposes than what it is designed for.

The model is made to create a diffusion among agents in order to analyze different perspectives of stability on the resulting s-curve. This means that the model is not a predictor on how a technology diffusion will happen in reality. The analogy of cars doesn't add any value to it. Therefore the model cannot be used as argument in the field of policies regarding electric vehicles. In addition, although the model is validated and verified, one should be aware that the model is made for a course with the purpose to experience all the modeling steps. Therefore, the model has not to be perfect, since the focus of the modeling approach was more on a system level than on a detailed level.

DISCUSSION ON THE MODELLING CYCLE EXPERIENCE

Starting the SPM9555 course, neither of us did have a clear idea on what agent-based modeling was, what it is (best) used for, and what are its key characteristics/advantages and disadvantages. Neither did any of us have any experience with programming in any language. Hence, following this course, gave us quite the challenge.

Step 1 of the ABM modeling cycle, problem formulation and actor identification, gave us quite a clear idea on the issue at hand, on a real abstract philosophical topic. In this cycle, we were greatly supported by Joolie in explaining the basic concepts and expectations.

During the system identification and decomposition (step 2), and concept formalization (step 3), we noticed our lack of knowledge on the ABM concepts. For us, not knowing what ABM exactly is, it felt a bit like making a causal relationship diagram for building a discrete-event model. Hence, we quickly went to the course document of SPM4530 which gave us 1) quite some extra work, but 2) the needed extra insight. Based on that knowledge, we would also make it through step 4: modal formalization.

As step 5, software implementation, meant our first ever programming experience, this again was quite a challenge, even to the NetLogo language proved to be relatively friendly. Fulfilling this step required quite a bit of wandering through the NetLogo dictionary, google through NetLogo communities and ask help from some more experienced NetLogo users inside TPM.

As the verification went smoothly, except for the weird normal-distribution issue, we started thinking on how to use and analyse the model results to contribute for evidence on the main hypothesis. The analysis part also posed the biggest challenge in the overall modeling cycle.

Barely having processed our first programming experience, in NetLogo, the data analysis required much more new learning. The ideas we had using Kolmogorov complexity initially seemed possible to do partially in Java, but would also require a significant extent of scripting in a Linux environment. However, when deepening our ideas on this analysis, we luckily found out all of the analysis was actually doable in PHP in a relatively easy code. However, to implement it, we still had to learn understand and use the PHP language. With this, the data analysis just started. As the initial output file contained over nearly 11 million lines of data, and the compressed data 443.000, we needed R to create some meaningful output. This meant the third programming language to be learned within a matter of weeks, let alone the mission of pattern recognition over these amounts of data.

Overall, we feel we have experienced some of the steepest learning curves one could get within the TPM faculty. However, seeing the full possibilities and strengths of ABM, we slightly doubt whether this course is in the right position within the SEPAM MSG program, as formally no background knowledge of any kind is required to take the course. But let us conclude with that achieving a learning curve this steep feels good.

REFERENCES

- Berwick, D. (2003). Disseminating Innovations in Health Care. *Journal of the American Medical Association*, 289(15), 1969-1975.
- Boulding, K. (1978). *Stable Peace*. University of Texas Press.
- Burke, S., & Schumann, T. (1928). Diffusion Flames. *Industrial & Engineering Chemistry*, 20(10), 998-1004.
- CBS. (2012). *Statline: Verkeersprestaties personenauto's: kilometers, brandstof en grondgebied*. Opgeroepen op December 2012, van <http://statline.cbs.nl/StatWeb/publication/?VW=T&DM=SLNL&PA=80428NED&LA=NL>
- Chen, P. (2004). A biological perspective of macro dynamics and the division of labor: persistent cycles, disruptive technology and the trade-off between stability and complexity. In K. Dopfer, *The evolutionary foundations of economics*. Cambridge: Cambridge University Press.
- Clausen, V. (1998). Money Demand and Monetary Policy in Europe. *Review of World Economics*, 134, 712-740.
- Dam, K. v., Nikolic, I., & Lukszo, Z. (2012). *Agent-based modelling of socio-technical systems*. Springer.
- Glandsdorff, P., & Prigogine, I. (1972). *Thermodynamic Theory of Structure, Stability and Fluctuations*. Wiley.
- Grofman, B., & Uhlauer, C. (1985). Metapreferences and the Reasons for Stability in Social Choice: Thoughts on Broadening and Clarifying the Debate. *Theory and Decision*, 19, 31-50.
- Hansson, S., & Helgesson, G. (2003). What is stability? *Synthese*, 136, 219-235.
- Kasmire, J. (2011). *Joolie's Guide to Setting up an Experimental in NetLogo, for the complete numpty*. Opgeroepen op January 2013, van wiki.tudelft.nl/bin/view/Education/SPM955xABMofCAS/JooliesExperimentalSetupStepByStep
- Mahajan, V., Muller, E., & Srivastava, R. (1990). Determination of Adopter Categories by Using Innovation Diffusion Models. *Journal of Marketing Research*, 27(1), 37-50.
- McCoy, E., & Shrader-Frechette, K. (1992). Community Ecology, Scale, and the Instability of the Stability Concept. *Philosophy of Science Association*, 184-199.
- Rogers, E. (1995). *Diffusions of Innovations* (5th ed.). New York: Free Press.
- Smith, A., Stirling, A., & Berkhout, F. (2005). The governance of sustainable socio-technical transitions. *Research policy*, 34(10), 1491-1510.
- Weisstein, E. (N.D.). *Pearson's Skewness Coefficients*. Opgeroepen op January 14, 2013, van <http://mathworld.wolfram.com/PearsonsSkewnessCoefficients.html>

APPENDICES

APPENDIX A: FULL MODEL CODE USED FOR MODEL VERIFICATION

```

globals [
  buyers-EV
  buyers-Petrol
  buyers-Diesel
]

turtles-own
[
  Electric-Car?      ;; if true, the turtle has an Electric-Vehicle
  Petrol-Car?       ;; if true, the turtle has an Petrol-Car
  Diesel-Car?       ;; if true, the turtle has an Diesel-Car
  car-type          ;; Holds string of text with car type

  Innovator?        ;; if true, turtle falls in innovator category
  Early-adopter?    ;; if true, turtle falls in early adopter category
  Early-majority?   ;; if true, turtle falls in early majority category
  Late-majority?    ;; if true, turtle falls in late majority category
  Laggards?         ;; if true, turtle falls in laggards category
  adopter-category  ;; Holds string of text with category of adopter

  car-age           ;; number of ticks since the car was bought
  car-lifetime      ;; number of ticks the car 'survives'

  attitude-to-innovation  ;; Determines the innovation category & change threshold of turtle
  change-threshold        ;; The willingness of an agent to change its technology

  ev-list                ;; Holds data on the amount of neighbors with an electric car for a specified period of ticks
  petrol-list            ;; Holds data on the amount of neighbors with a petrol car for a specified period of ticks
  diesel-list            ;; Holds data on the amount of neighbors with a diesel car for a specified period of ticks

  car-history-list      ;; Holds data on the history of car types of each turtle
]

;;;;;;;;;;;;;;
;;; Setup Procedures ;;;
;;;;;;;;;;;;;;

to setup
  clear-all
  ask patches [set pcolor white]
  setup-nodes
  setup-initial-car
  setup-network
  setup-innovation-preferences
  setup-lists
  set buyers-EV 0
  set buyers-Petrol 0
  set buyers-Diesel 0
  reset-ticks
end

to setup-nodes
  set-default-shape turtles "circle"
  crt number-of-car-owners[
    setxy (random-xcor * 0.90) (random-ycor * 0.90) ;; don't put any nodes real close to the edges
  ]
end

to setup-initial-car
  ask turtles [if Log-car-owner-stats? [set ev-list (list 0) set petrol-list (list 0) set diesel-list (list 0) set car-type "initial setup"]]
  ask turtles [become-PETROLowner]
  ask n-of ((number-of-car-owners * initial-diesel-percentage) / 100) turtles with [Petrol-Car?] [become-DIESELowner]
  ;; it might be possible that a laggard start with an insuperior technology (i.e. diesel), while being surrounded by other technologies

  ask turtles [
    if initial-car-age = 0 [set car-age 0]
    if initial-car-age = "random" [set car-age random car-lifetime]
  ]
]

```

```

;; this sets the turtles car properties
end

to setup-network
let num-links (average-links-per-car-owner * number-of-car-owners) / 2      ;; divide by 2 as each link connects two nodes
ask turtles [if count link-neighbors <= 0
  [create-link-with min-one-of (other turtles with [not link-neighbor? myself]) [distance myself]]] ;; make sure every node has at least 1 link
while [count links <= num-links ]
  [ask one-of turtles
    [ let choice (min-one-of (other turtles with [not link-neighbor? myself])
      [distance myself])
      if choice != nobody [ create-link-with choice ]          ;; add connect links if node is unconnected
    ]
  ]

ask links [ set color grey ]

repeat 5
  [layout-spring turtles links 0.5 (world-width / (sqrt number-of-car-owners)) 1] ;; make network look bit nicer
end

to setup-innovation-preferences
ask turtles [
  set attitude-to-innovation random-normal 0.5 0.16      ;; normal distribution N(0.5;0.16), this follows theory
  set change-threshold (round (attitude-to-innovation * count link-neighbors)) ;; threshold is now set to number of neighbors
  if change-threshold < 0 [set change-threshold 0]
  if change-threshold > count link-neighbors [set change-threshold count link-neighbors]

  ;; make sure no turtle is categorized yet
  set Innovator? false
  set Early-adopter? false
  set Early-majority? false
  set Late-majority? false
  set Laggards? false

  ;; arrange categorization of turtles
  if attitude-to-innovation >= 0.84 [set Laggards? true set adopter-category "laggard"]
  if attitude-to-innovation < 0.84 [set Late-majority? true set adopter-category "late majority"]
  if attitude-to-innovation < 0.50 [set Early-majority? true set late-majority? false set adopter-category "early majority"]
  if attitude-to-innovation < 0.16 [set Early-adopter? true set early-majority? false set adopter-category "early adopter"]
  if attitude-to-innovation < 0.025 [set Innovator? true set early-adopter? false set adopter-category "innovator"]

]
end

to setup-lists
ask turtles [
  ;; record initial environment of turtle + own car type
  set ev-list (list 0)
  ifelse Petrol-Car? [set petrol-list (list (1 + count link-neighbors with [Petrol-Car?]))] [set petrol-list (list count link-neighbors with [Petrol-Car?])]
  ifelse Diesel-Car? [set diesel-list (list (1 + count link-neighbors with [Diesel-Car?]))] [set diesel-list (list count link-neighbors with [Diesel-Car?])]

  ;; record initial car type of turtle
  ifelse Petrol-Car? [set car-history-list (list "PETROL")]
  [ifelse Diesel-Car? [set car-history-list (list "DIESEL")]
    [show "ERROR in car history list"]
  ]
]
end

.....
;;; Go Procedures ;;;
.....

to go
set buyers-EV 0

```

```

set buyers-Petrol 0
set buyers-Diesel 0
ask turtles [set car-age car-age + 1]
ask turtles [if Log-car-owner-stats? [print (word self "my car is now " car-age " ticks, while the lifetime of my car is " round car-lifetime "
(rounded) ticks" )]]
ask turtles [update-memory]
ask turtles with [car-age >= car-lifetime] [buy-car]
tick
end

to update-memory      ;; 10 ticks equal 1 year, this section does not cope with shortening 'car-owner-memory' during run time
adequately
  ifelse Electric-Car? [set ev-list lput (1 + count link-neighbors with [Electric-Car?]) ev-list] [set ev-list lput count link-neighbors with [Electric-
Car?] ev-list]
  ifelse Petrol-Car? [set petrol-list lput (1 + count link-neighbors with [Petrol-Car?]) petrol-list] [set petrol-list lput count link-neighbors with
[Petrol-Car?] petrol-list]
  ifelse Diesel-Car? [set diesel-list lput (1 + count link-neighbors with [Diesel-Car?]) diesel-list] [set diesel-list lput count link-neighbors with
[Diesel-Car?] diesel-list]

if Innovator? [
  ;; limit list for innovators
  if length ev-list > (innovator-memory * 10) [set ev-list remove-item 0 ev-list]
  if length petrol-list > (innovator-memory * 10) [set petrol-list remove-item 0 petrol-list]
  if length diesel-list > (innovator-memory * 10) [set diesel-list remove-item 0 diesel-list]
]

if Early-adopter? [
  ;; limit list for early-adopters
  if length ev-list > (early-adopter-memory * 10) [set ev-list remove-item 0 ev-list]
  if length petrol-list > (early-adopter-memory * 10) [set petrol-list remove-item 0 petrol-list]
  if length diesel-list > (early-adopter-memory * 10) [set diesel-list remove-item 0 diesel-list]
]

if Early-majority? [
  ;; limit list for early-majority
  if length ev-list > (early-majority-memory * 10) [set ev-list remove-item 0 ev-list]
  if length petrol-list > (early-majority-memory * 10) [set petrol-list remove-item 0 petrol-list]
  if length diesel-list > (early-majority-memory * 10) [set diesel-list remove-item 0 diesel-list]
]

if Late-majority? [
  ;; limit list for late-majority
  if length ev-list > (late-majority-memory * 10) [set ev-list remove-item 0 ev-list]
  if length petrol-list > (late-majority-memory * 10) [set petrol-list remove-item 0 petrol-list]
  if length diesel-list > (late-majority-memory * 10) [set diesel-list remove-item 0 diesel-list]
]

if Laggards? [
  ;; limit list for laggards
  if length ev-list > (laggard-memory * 10) [set ev-list remove-item 0 ev-list]
  if length petrol-list > (laggard-memory * 10) [set petrol-list remove-item 0 petrol-list]
  if length diesel-list > (laggard-memory * 10) [set diesel-list remove-item 0 diesel-list]
]
end

to buy-car
  ifelse ticks < (years-to-ev-invention * 10) [
    ;; procedure in case ev is not evented yet
    ifelse Diesel-Car? [
      ifelse mean petrol-list >= change-threshold [become-PETROLowner] [become-DIESELowner]
    ] [ifelse Petrol-Car? [
      ifelse mean diesel-list >= change-threshold [become-DIESELowner] [become-PETROLowner]
    ]
    ]
    [show "DECISION ERROR IN BUYING CAR BEFORE EV INVENTION"]
  ]
  [
    ;; procedure in case ev is invented
    ifelse Electric-Car? [

```

```

if (mean diesel-list >= change-threshold) and (mean petrol-list >= change-threshold) [
  if Innovator? or early-adopter? [
    if mean diesel-list > mean petrol-list [become-PETROLowner]
    if mean diesel-list < mean petrol-list [become-DIESELowner]
    if mean diesel-list = mean petrol-list [ifelse random-float 1 <= 0.50 [become-DIESELowner] [become-PETROLowner]]]
  if early-majority? or late-majority? [
    ifelse random-float 1 <= 0.50 [become-DIESELowner] [become-PETROLowner]]
  if laggards? [
    if mean diesel-list < mean petrol-list [become-PETROLowner]
    if mean diesel-list > mean petrol-list [become-DIESELowner]
    if mean diesel-list = mean petrol-list [ifelse random-float 1 <= 0.50 [become-DIESELowner] [become-PETROLowner]]]]
  if (mean diesel-list >= change-threshold) and (mean petrol-list < change-threshold)[become-DIESELowner]
  if (mean diesel-list < change-threshold) and (mean petrol-list >= change-threshold)[become-PETROLowner]
  if (mean diesel-list < change-threshold) and (mean petrol-list < change-threshold)[become-EVowner]]

[ifelse Diesel-Car? [
  if (mean ev-list >= change-threshold) and (mean petrol-list >= change-threshold) [
    if Innovator? or early-adopter? [
      if mean ev-list > mean petrol-list [become-PETROLowner]
      if mean ev-list < mean petrol-list [become-EVowner]
      if mean ev-list = mean petrol-list [ifelse random-float 1 <= 0.50 [become-EVowner] [become-PETROLowner]]]
    if early-majority? or late-majority? [
      ifelse random-float 1 <= 0.50 [become-EVowner] [become-PETROLowner]]
    if laggards? [
      if mean ev-list < mean petrol-list [become-PETROLowner]
      if mean ev-list > mean petrol-list [become-EVowner]
      if mean ev-list = mean petrol-list [ifelse random-float 1 <= 0.50 [become-EVowner] [become-PETROLowner]]]]
    if (mean ev-list >= change-threshold) and (mean petrol-list < change-threshold)[become-EVowner]
    if (mean ev-list < change-threshold) and (mean petrol-list >= change-threshold)[become-PETROLowner]
    if (mean ev-list < change-threshold) and (mean petrol-list < change-threshold)[become-DIESELowner]]
  ]

[ifelse Petrol-Car? [
  if (mean ev-list >= change-threshold) and (mean diesel-list >= change-threshold) [
    if Innovator? or early-adopter? [
      if mean ev-list > mean diesel-list [become-DIESELowner]
      if mean ev-list < mean diesel-list [become-EVowner]
      if mean ev-list = mean diesel-list [ifelse random-float 1 <= 0.50 [become-EVowner] [become-DIESELowner]]]
    if early-majority? or late-majority? [
      ifelse random-float 1 <= 0.50 [become-EVowner] [become-DIESELowner]]
    if laggards? [
      if mean ev-list < mean diesel-list [become-DIESELowner]
      if mean ev-list > mean diesel-list [become-EVowner]
      if mean ev-list = mean diesel-list [ifelse random-float 1 <= 0.50 [become-EVowner] [become-DIESELowner]]]]
    if (mean ev-list >= change-threshold) and (mean diesel-list < change-threshold)[become-EVowner]
    if (mean ev-list < change-threshold) and (mean diesel-list >= change-threshold)[become-DIESELowner]
    if (mean ev-list < change-threshold) and (mean diesel-list < change-threshold)[become-PETROLowner]]
  ]
]
[show "DECISION ERROR IN BUYING CAR AFTER EV INVENTION"]
]
]

if Electric-Car? [set car-history-list lput "EV" car-history-list]
if Diesel-Car? [set car-history-list lput "DIESEL" car-history-list]
if Petrol-Car? [set car-history-list lput "PETROL" car-history-list]
end

; ; ; Car owner Procedures ; ; ;

to become-EVowner
  if Log-car-owner-stats? [print (word self "I buy EV, as my change threshold is " change-threshold " and my memory values are " mean ev-list " " mean petrol-list " " mean diesel-list " for ev, petrol and diesel respectively")]
  if Log-car-owner-stats? [print (word self "By the way, " adopter-category " is my adopter category and I had a " car-type "car")]
  if Log-car-owner-stats? [set car-type "EV"]
  set Electric-Car? true
  set Petrol-Car? false
  set Diesel-Car? false
  set color lime
  set size 2
  set buyers-EV buyers-EV + 1

```

```

set car-age 0
set car-lifetime random-normal (average-ev-car-life * 10) 20 ;;10 ticks equal 1 year --> car-lifetime now follows N(average car life, 2)
year distribution
end

to become-PETROLowner
if Log-car-owner-stats? [print (word self "I buy Petrol, as my change threshold is " change-threshold " and my memory values are " mean
ev-list " " mean petrol-list " " mean diesel-list " for ev, petrol and diesel respectively)]
if Log-car-owner-stats? [print (word self "By the way, " adopter-category " is my adopter category and I had a " car-type "car")]
if Log-car-owner-stats? [set car-type "PETROL"]
set Electric-Car? false
set Petrol-Car? true
set Diesel-Car? false
set color red
set size 2
set buyers-Petrol buyers-Petrol + 1

set car-age 0
set car-lifetime random-normal (average-petrol-car-life * 10) 20 ;;10 ticks equal 1 year --> car-lifetime now follows N(average car life, 2)
year distribution
end

to become-DIESELowner
if Log-car-owner-stats? [print (word self "I buy Diesel, as my change threshold is " change-threshold " and my memory values are " mean
ev-list " " mean petrol-list " " mean diesel-list " for ev, petrol and diesel respectively)]
if Log-car-owner-stats? [print (word self "By the way, " adopter-category " is my adopter category and I had a " car-type "car")]
if Log-car-owner-stats? [set car-type "DIESEL"]
set Electric-Car? false
set Petrol-Car? false
set Diesel-Car? true
set color sky
set size 2
set buyers-Diesel buyers-Diesel + 1

set car-age 0
set car-lifetime random-normal (average-diesel-car-life * 10) 20 ;;10 ticks equal 1 year --> car-lifetime now follows N(average car life, 2)
year distribution
end

.....
.....; Single car owner verification .....
.....

to setup-single-car-owner-verification
;; setup environment
clear-all
ask patches [set pcolor white]

;; setup turtle
set-default-shape turtles "circle"
crt 1 [setxy (random-xcor * 0.01) (random-ycor * 0.01) ]

ask turtles [if Log-car-owner-stats? [set ev-list (list 0) set petrol-list (list 0) set diesel-list (list 0) set car-type "initial setup"]]
ask turtles [become-PETROLowner]
ask turtles [
if initial-car-age = 0 [set car-age 0]
if initial-car-age = "random" [set car-age random car-lifetime]
if fixed-initial-car-age [set car-age fixed-initial-car-age-value]
]

setup-innovation-preferences

if fix-change-threshold [ask turtles [set change-threshold fixed-change-threshold-value]]

setup-lists
reset-ticks

```

end

to go-single-car-owner-verification

```

;; set car age
ifelse only-buying-procedure? [ask turtles [set car-age car-lifetime]] [ask turtles [set car-age car-age + 1]]
;; report car age and car lifetime
if only-buying-procedure? = false [ask turtles [if Log-car-owner-stats? [print (word self "my car is now " car-age " ticks, while the lifetime of
my car is " round car-lifetime " (rounded) ticks"))]]]

;; update memory
ask turtles [

  ifelse Electric-Car?
    [ifelse strong-ev-influence [set ev-list lput fixed-influence-value ev-list] [set ev-list lput (1 + count link-neighbors with [Electric-Car?]) ev-
list]]
    [ifelse strong-ev-influence [set ev-list lput fixed-influence-value ev-list] [set ev-list lput count link-neighbors with [Electric-Car?] ev-list]]
  ifelse Petrol-Car?
    [ifelse strong-petrol-influence [set petrol-list lput fixed-influence-value petrol-list] [set petrol-list lput (1 + count link-neighbors with
[Petrol-Car?]) petrol-list]]
    [ifelse strong-petrol-influence [set petrol-list lput fixed-influence-value petrol-list] [set petrol-list lput count link-neighbors with [Petrol-
Car?] petrol-list]]
  ifelse Diesel-Car?
    [ifelse strong-diesel-influence [set diesel-list lput fixed-influence-value diesel-list] [set diesel-list lput (1 + count link-neighbors with
[Diesel-Car?]) diesel-list]]
    [ifelse strong-diesel-influence [set diesel-list lput fixed-influence-value diesel-list] [set diesel-list lput count link-neighbors with [Diesel-
Car?] diesel-list]]

  if Innovator? [
    ;; limit list for innovators
    if length ev-list > (innovator-memory * 10) [set ev-list remove-item 0 ev-list]
    if length petrol-list > (innovator-memory * 10) [set petrol-list remove-item 0 petrol-list]
    if length diesel-list > (innovator-memory * 10) [set diesel-list remove-item 0 diesel-list]
  ]

  if Early-adopter? [
    ;; limit list for early-adopters
    if length ev-list > (early-adopter-memory * 10) [set ev-list remove-item 0 ev-list]
    if length petrol-list > (early-adopter-memory * 10) [set petrol-list remove-item 0 petrol-list]
    if length diesel-list > (early-adopter-memory * 10) [set diesel-list remove-item 0 diesel-list]
  ]

  if Early-majority? [
    ;; limit list for early-majority
    if length ev-list > (early-majority-memory * 10) [set ev-list remove-item 0 ev-list]
    if length petrol-list > (early-majority-memory * 10) [set petrol-list remove-item 0 petrol-list]
    if length diesel-list > (early-majority-memory * 10) [set diesel-list remove-item 0 diesel-list]
  ]

  if Late-majority? [
    ;; limit list for late-majority
    if length ev-list > (late-majority-memory * 10) [set ev-list remove-item 0 ev-list]
    if length petrol-list > (late-majority-memory * 10) [set petrol-list remove-item 0 petrol-list]
    if length diesel-list > (late-majority-memory * 10) [set diesel-list remove-item 0 diesel-list]
  ]

  if Laggards? [
    ;; limit list for laggards
    if length ev-list > (laggard-memory * 10) [set ev-list remove-item 0 ev-list]
    if length petrol-list > (laggard-memory * 10) [set petrol-list remove-item 0 petrol-list]
    if length diesel-list > (laggard-memory * 10) [set diesel-list remove-item 0 diesel-list]
  ]
]

;;turtles buy car if needed
ask turtles with [car-age >= car-lifetime] [buy-car]
tick
end

```

```

.....
..... Minimal model interaction testing .....
.....

```

```

.....

to setup-minimal-model-verification
  ;; setup environment
  clear-all
  ask patches [set pcolor white]

  ;;setup turtles
  set-default-shape turtles "circle"
  crt 2 [setxy (random-xcor * 0.90) (random-ycor * 0.90) ]

  ask turtles [if Log-car-owner-stats? [set ev-list (list 0) set petrol-list (list 0) set diesel-list (list 0) set car-type "initial setup"]]
  ask turtles [become-PETROLowner]
  ask one-of turtles [become-DIESELowner]
  ask turtles [
    if initial-car-age = 0 [set car-age 0]
    if initial-car-age = "random" [set car-age random car-lifetime]
    if fixed-initial-car-age [set car-age fixed-initial-car-age-value]
  ]

  ;;setup link between turtle
  let num-links 1
  while [count links < num-links ]
    [ask one-of turtles
      [ let choice (min-one-of (other turtles with [not link-neighbor? myself])
        [distance myself])
        if choice != nobody [ create-link-with choice ]           ;; add connect links if node is unconnected
      ]
    ]

  setup-innovation-preferences

  if fix-change-threshold [ask turtles [set change-threshold fixed-change-threshold-value]]

  setup-lists
  reset-ticks
end

to go-minimal-model-verification
  ;;determine car age
  ifelse only-buying-procedure? [ask turtles [set car-age car-lifetime]] [ask turtles [set car-age car-age + 1]]
  ;; report agent car age and lifetime
  if only-buying-procedure? = false [ask turtles [if Log-car-owner-stats? [print (word self "my car is now " car-age " ticks, while the lifetime of
my car is " round car-lifetime " (rounded) ticks")]]]

  ;; update memory
  ask turtles [

    ifelse Electric-Car?
      [ifelse strong-ev-influence [set ev-list lput fixed-influence-value ev-list] [set ev-list lput (1 + count link-neighbors with [Electric-Car?]) ev-
list]]
      [ifelse strong-ev-influence [set ev-list lput fixed-influence-value ev-list] [set ev-list lput count link-neighbors with [Electric-Car?] ev-list]]
    ifelse Petrol-Car?
      [ifelse strong-petrol-influence [set petrol-list lput fixed-influence-value petrol-list] [set petrol-list lput (1 + count link-neighbors with
[Petrol-Car?]) petrol-list]]
      [ifelse strong-petrol-influence [set petrol-list lput fixed-influence-value petrol-list] [set petrol-list lput count link-neighbors with [Petrol-
Car?] petrol-list]]
    ifelse Diesel-Car?
      [ifelse strong-diesel-influence [set diesel-list lput fixed-influence-value diesel-list] [set diesel-list lput (1 + count link-neighbors with
[Diesel-Car?]) diesel-list]]
      [ifelse strong-diesel-influence [set diesel-list lput fixed-influence-value diesel-list] [set diesel-list lput count link-neighbors with [Diesel-
Car?] diesel-list]]

    if Innovator? [
      ;; limit list for innovators
      if length ev-list > (innovator-memory * 10) [set ev-list remove-item 0 ev-list]
      if length petrol-list > (innovator-memory * 10) [set petrol-list remove-item 0 petrol-list]
      if length diesel-list > (innovator-memory * 10) [set diesel-list remove-item 0 diesel-list]
    ]

    if Early-adopter? [

```



```
;; limit list for early-adopters
if length ev-list > (early-adopter-memory * 10) [set ev-list remove-item 0 ev-list]
if length petrol-list > (early-adopter-memory * 10) [set petrol-list remove-item 0 petrol-list]
if length diesel-list > (early-adopter-memory * 10) [set diesel-list remove-item 0 diesel-list]
]

if Early-majority? [
;; limit list for early-majority
if length ev-list > (early-majority-memory * 10) [set ev-list remove-item 0 ev-list]
if length petrol-list > (early-majority-memory * 10) [set petrol-list remove-item 0 petrol-list]
if length diesel-list > (early-majority-memory * 10) [set diesel-list remove-item 0 diesel-list]
]

if Late-majority? [
;; limit list for late-majority
if length ev-list > (late-majority-memory * 10) [set ev-list remove-item 0 ev-list]
if length petrol-list > (late-majority-memory * 10) [set petrol-list remove-item 0 petrol-list]
if length diesel-list > (late-majority-memory * 10) [set diesel-list remove-item 0 diesel-list]
]

if Laggards? [
;; limit list for laggards
if length ev-list > (laggard-memory * 10) [set ev-list remove-item 0 ev-list]
if length petrol-list > (laggard-memory * 10) [set petrol-list remove-item 0 petrol-list]
if length diesel-list > (laggard-memory * 10) [set diesel-list remove-item 0 diesel-list]
]
]
;; turtles buy car if needed
ask turtles with [car-age >= car-lifetime] [buy-car]
tick
end
```

APPENDIX B: REWRITTEN CODE FOR RECORDING/TRACKING AGENT BEHAVIOR

The following line of code is implemented between line 57 and 58 of the code description in the section on software implementation:

```
ask turtles [if Log-car-owner-stats? [set ev-list (list 0) set petrol-list (list 0) set diesel-list (list 0) set car-type "initial setup"]]
```

Furthermore, code lines 103 to 107 get a slight addition in respect to the model description given in the lecture on software implementation. The original code:

```
if attitude-to-innovation >= 0.84 [set Laggards? true]  
if attitude-to-innovation < 0.84 [set Late-majority? true]  
if attitude-to-innovation < 0.50 [set Early-majority? true set late-majority? false]  
if attitude-to-innovation < 0.16 [set Early-adopter? true set early-majority? false]  
if attitude-to-innovation < 0.025 [set Innovator? true set early-adopter? false]
```

The 'new' code:

```
if attitude-to-innovation >= 0.84 [set Laggards? true set adopter-category "laggard"]  
if attitude-to-innovation < 0.84 [set Late-majority? true set adopter-category "late majority"]  
if attitude-to-innovation < 0.50 [set Early-majority? true set late-majority? false set adopter-category "early majority"]  
if attitude-to-innovation < 0.16 [set Early-adopter? true set early-majority? false set adopter-category "early adopter"]  
if attitude-to-innovation < 0.025 [set Innovator? true set early-adopter? false set adopter-category "innovator"]
```

APPENDIX C: COMPARISON ON DEGREE ON COMPRESSION OF # EV (HISTOGRAMS)

This appendix shows the histogram of the sizes of the compressed strings, whereby the amount of turtles that own the innovated technology over the time are compressed. The graphs compares the level of compression for the linear part with the end part of the s-curve, for all the twelve relevant runsets.

The linear part of the s-curve is depicted with a blue histogram, where the end part of the s-curve is depicted with a red histogram. The histogram shows the frequency of sizes of compressed strings. This means that the more the histogram is to the left, the more this part of the s-curve could be compressed.

This appendix shows for almost every runset that the blue histogram is more to the left then the red histogram. Therefore, the linear part of the s-curve is more compressed than the end part of the s-curve. Therefore, the linear part can better be compress than the end part of the s-curve.

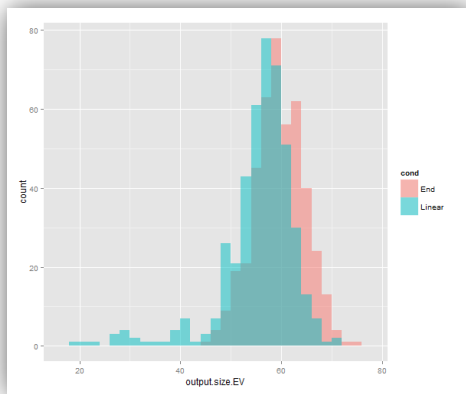


Figure C.1 - Runset 1

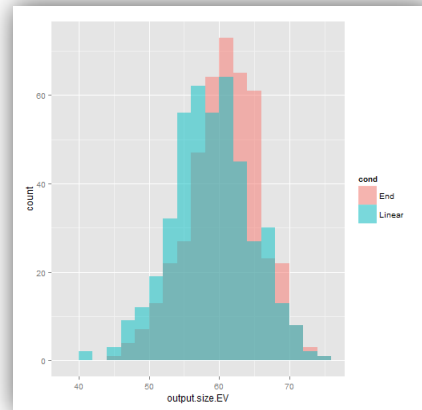


Figure C.4 - Runset 4

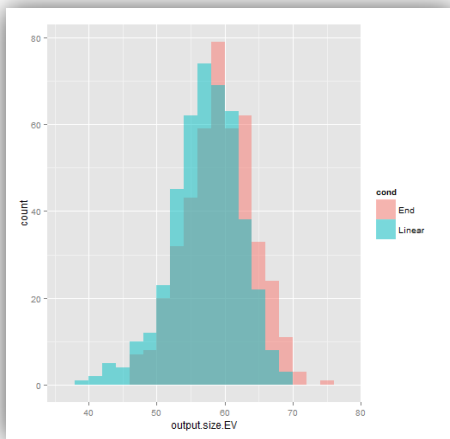


Figure C.2 - Runset 2

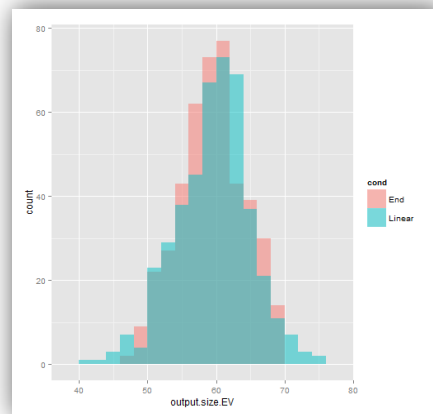


Figure C.5 - Runset 5

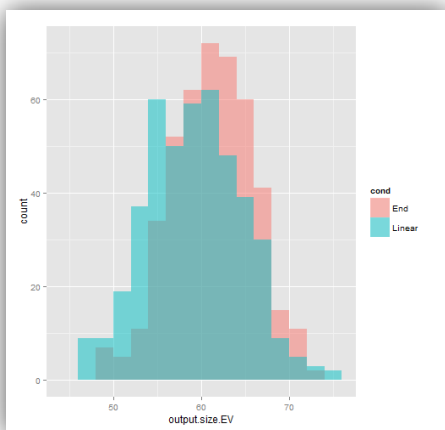


Figure C.3- Runset 3

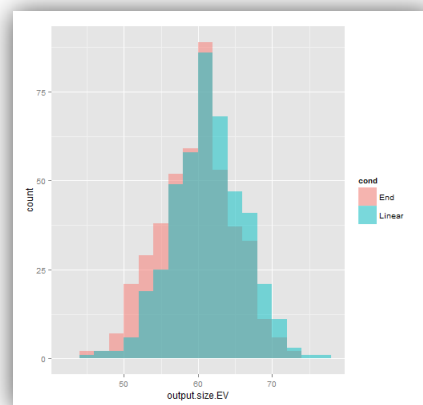


Figure C.6 - Runset 6

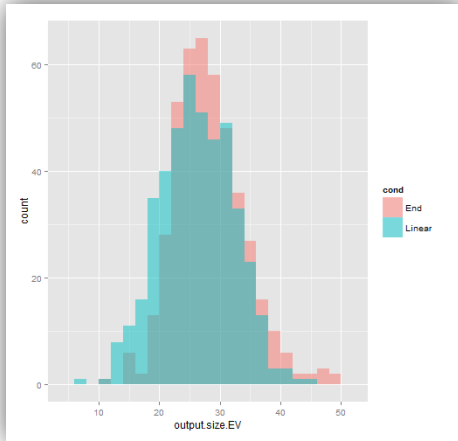


Figure C.7 - Runset 7

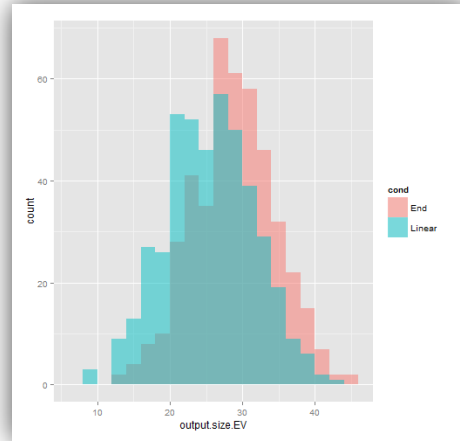


Figure C.10 - Runset 10

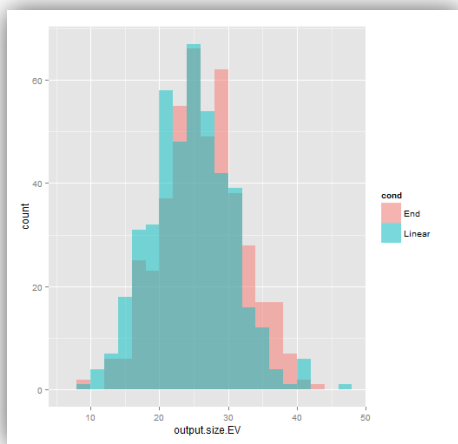


Figure C.8 - Runset 8

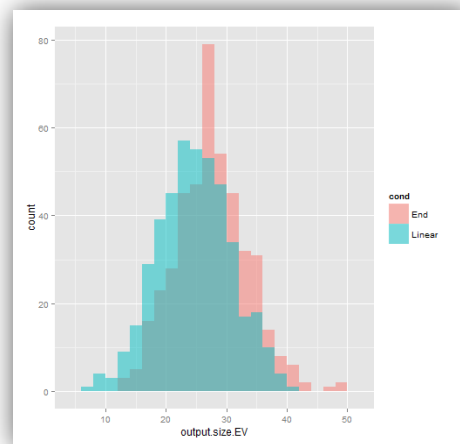


Figure C.11 - Runset 11

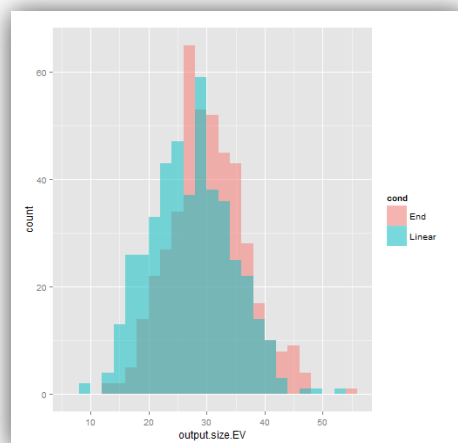


Figure C.9 - Runset 9

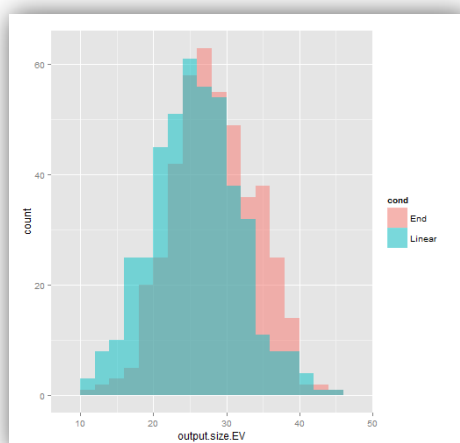


Figure C.12 - Runset 12

APPENDIX D: COMPARISON ON DEGREE ON COMPRESSION OF # EV (DENSITY CURVE)

This appendix shows the density curves of the sizes of the compressed strings, whereby the amount of turtles that own the innovated technology over the time are compressed. The graphs compares the level of compression for the linear part with the end part of the s-curve, for all the twelve relevant runsets.

The linear part of the s-curve is depicted with a blue curve, where the end part of the s-curve is depicted with a red curve. This means that the more the curve is to the left, the more this part of the s-curve could be compressed.

This appendix shows for almost every runset that the blue histogram is more to the left then the red histogram. Therefore, the linear part of the s-curve is more compressed than the end part of the s-curve. Therefore, the linear part can better be compress than the end part of the s-curve

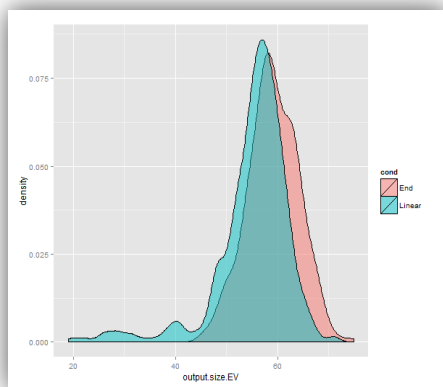


Figure D.1 - Runset 1

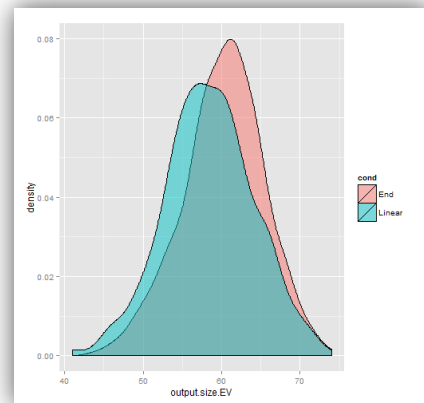


Figure D.4 - Runset 4

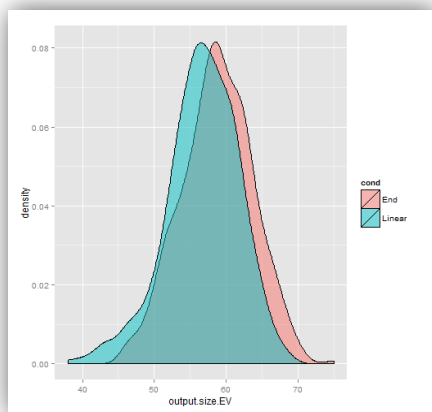


Figure D.2 - Runset 2

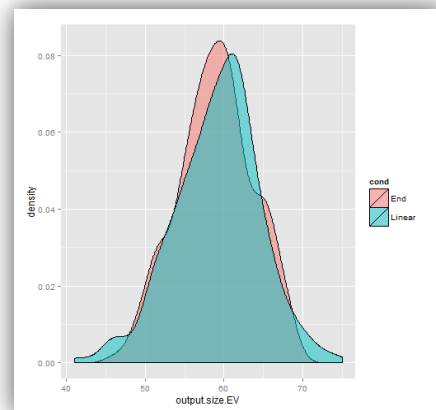


Figure D.5 - Runset 5

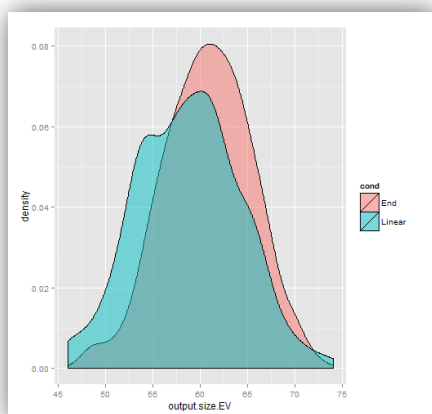


Figure D.3- Runset 3

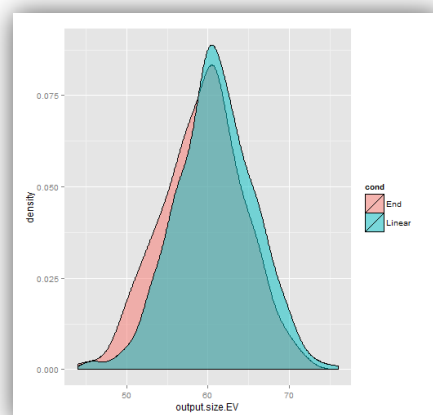


Figure D.6 - Runset 6

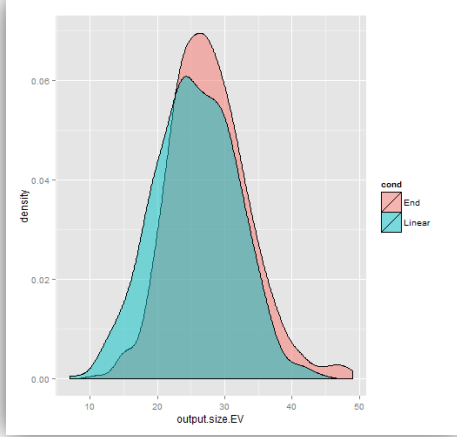


Figure D.7 - Runset 7

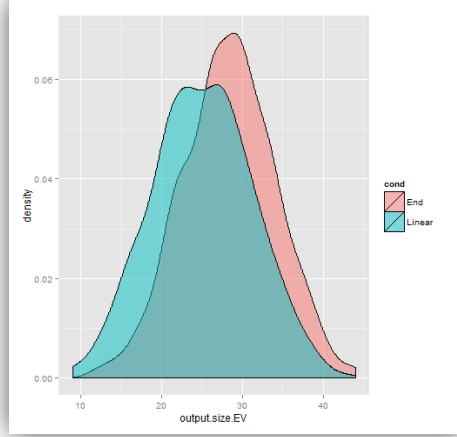


Figure D.10 - Runset 10

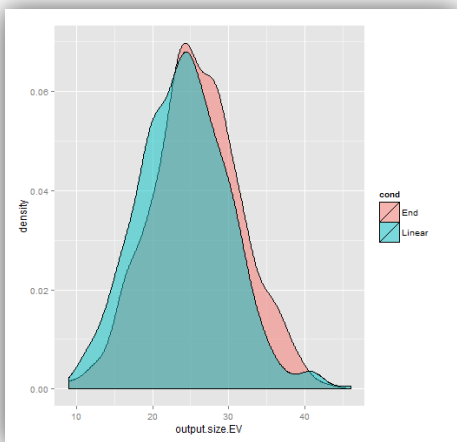


Figure D.8 - Runset 8

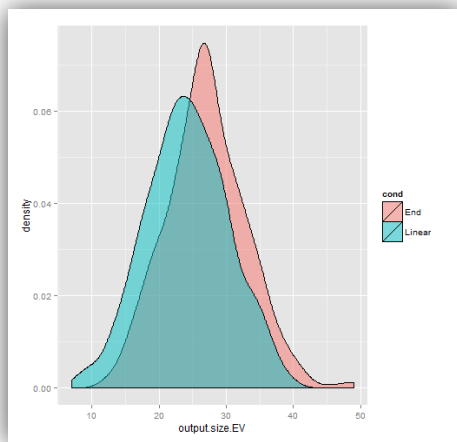


Figure D.11 - Runset 11

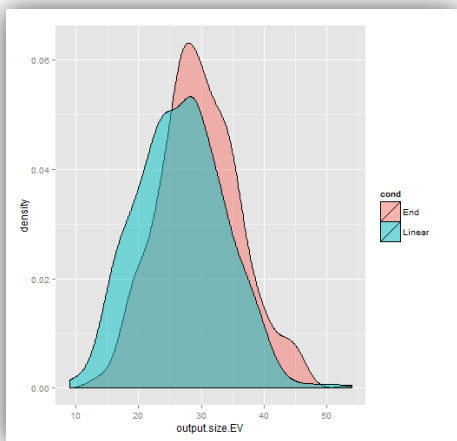


Figure D.9 - Runset 9

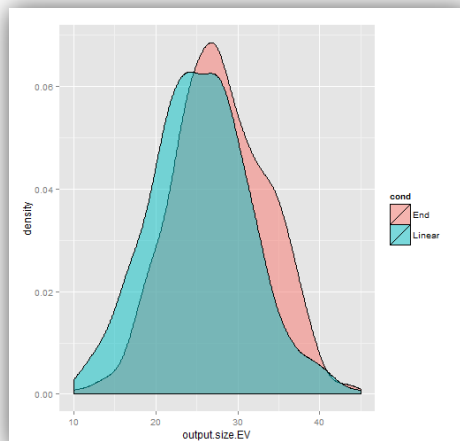


Figure D.12 - Runset 12

APPENDIX E: COMPARISON ON OVERALL COMPRESSION SIZES (HISTOGRAMS)

This appendix shows the histograms of the sizes of the compressed strings, whereby all three technologies are compressed and the compressed values are add into a single compressed value that represents the compression of the whole system. The graphs compare the linear part with the end part of the s-curve, for all the twelve relevant runsets. The linear part of the s-curve is depicted with a blue histogram, where the end part of the s-curve is depicted with a red histogram. The histogram shows the frequency of sizes of compressed strings. This means that the more the histogram is to the left, the more the string was been compressed.

This appendix shows for almost every runset the red histogram curve is more to the left, therefore more compressed than the red histogram. Therefore, when the whole system is taken into account, there can be concluded that the end part of the s-curve can better be compress than the linear part of the s-curve.

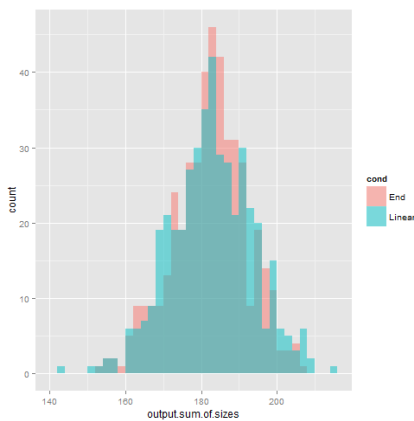


Figure E.1 - Runset 1

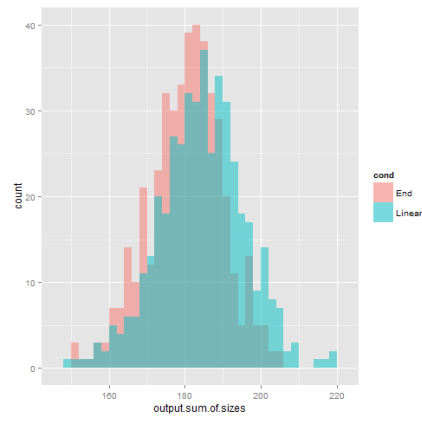


Figure E.4 - Runset 4

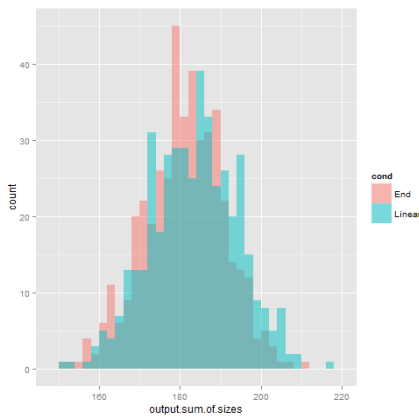


Figure E.2 - Runset 2

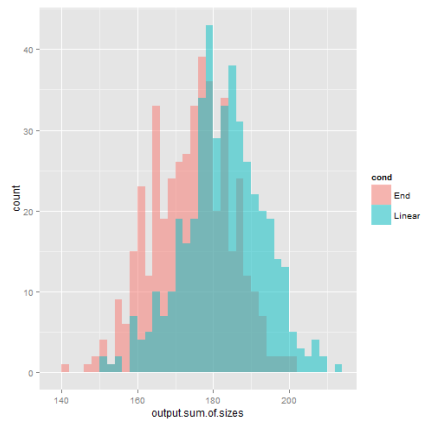


Figure E.5 - Runset 5

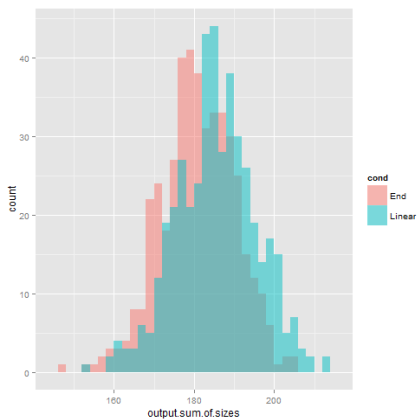


Figure E.3- Runset 3

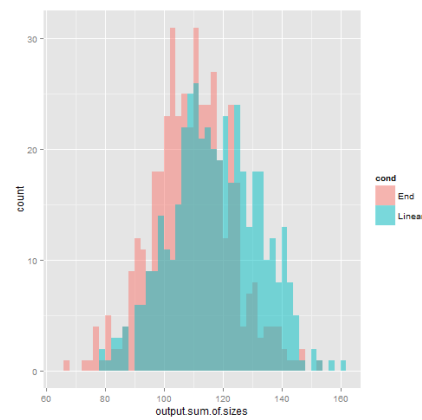


Figure E.6 - Runset 6

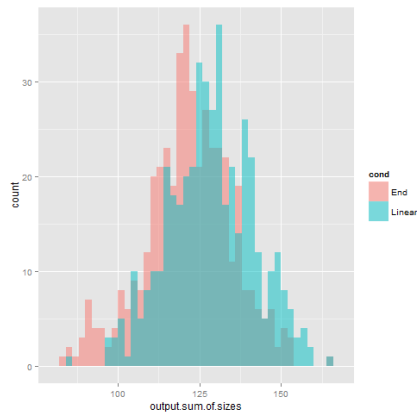


Figure E.7 - Runset 7

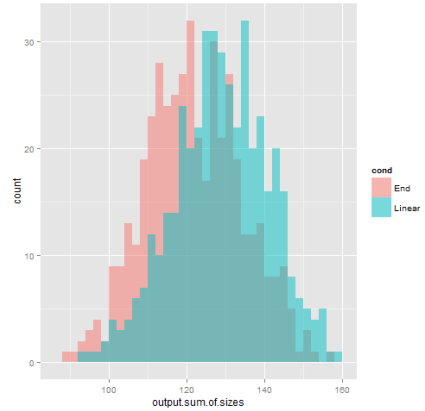


Figure E.10 - Runset 10

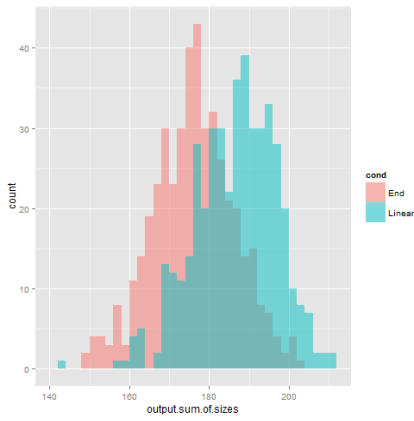


Figure E.8 - Runset 8

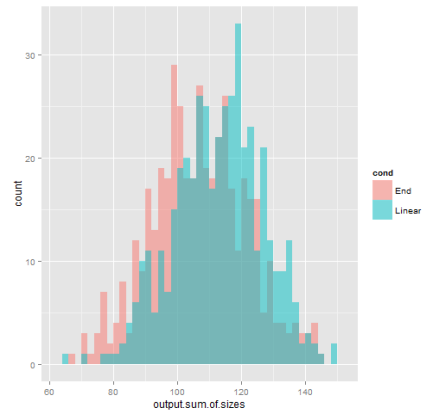


Figure E.11 - Runset 11

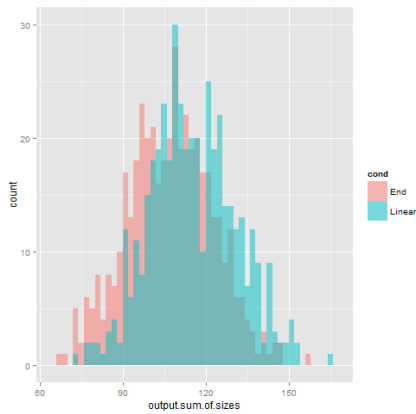


Figure E.9- Runset 9

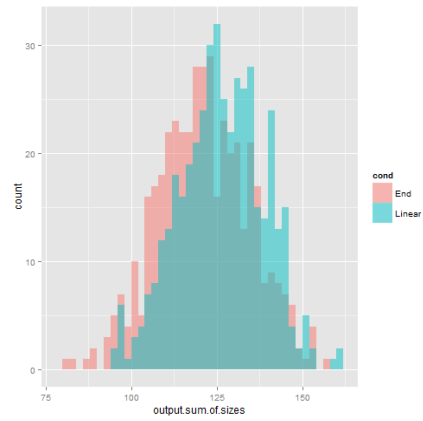


Figure E.12 - Runset 12

APPENDIX F: COMPARISON ON OVERALL COMPRESSION SIZES (DENSITY CURVES)

This appendix shows the density curves of the sizes of the compressed strings, whereby all three technologies are compressed and the compressed values are add into a single compressed value that represents the compression of the whole system. The graphs compare the linear part with the end part of the s-curve, for all the twelve relevant runsets. The linear part of the s-curve is depicted with a blue density curve, where the end part of the s-curve is depicted with a red density curve. This means that the more the density curve is to the left, the more the string was been compressed.

This appendix shows for almost every runset the red density curve is more to the left, therefore more compressed than the red density curve. Therefore, when the whole system is taken into account, there can be concluded that the end part of the s-curve can better be compress than the linear part of the s-curve.

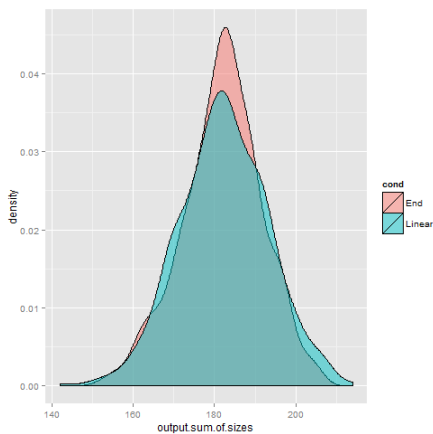


Figure F.1 - Runset 1

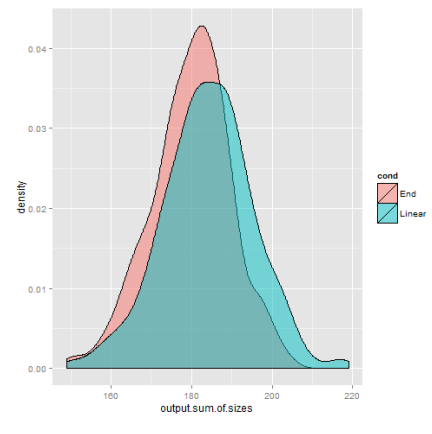


Figure F.4 - Runset 4

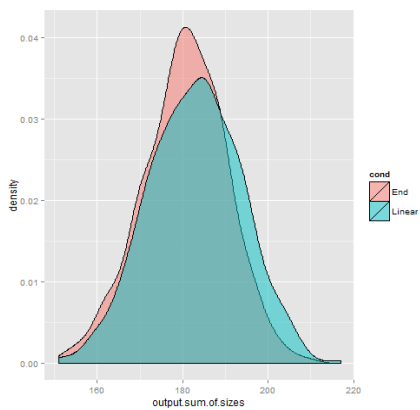


Figure F.2 - Runset 2

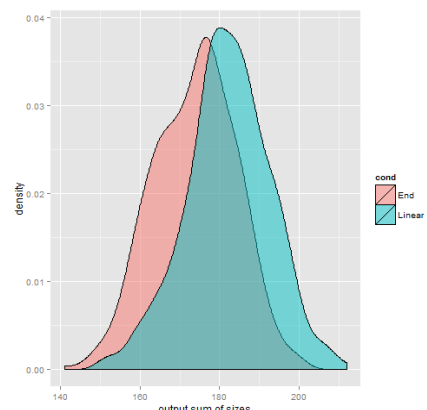


Figure F.5 - Runset 5

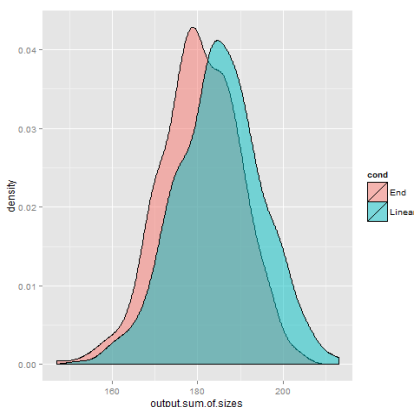


Figure F.3 - Runset 3

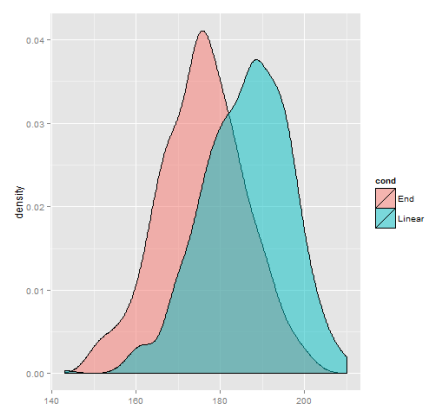


Figure F.6 - Runset 6

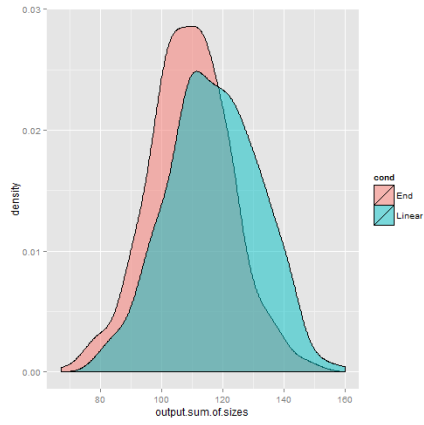


Figure F.7 - Runset 7

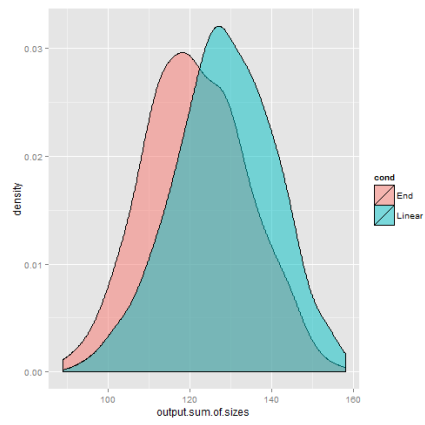


Figure F.10 - Runset 10

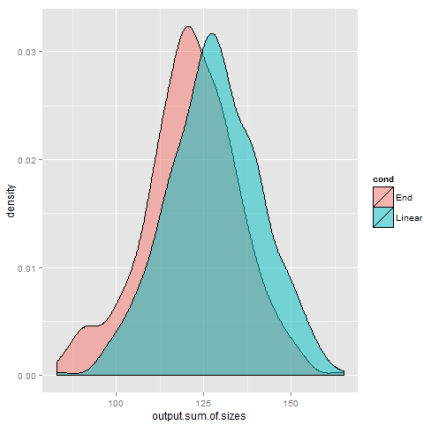


Figure F.8 - Runset 8

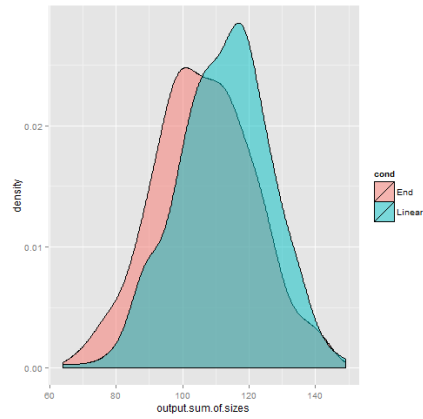


Figure F.11 - Runset 11

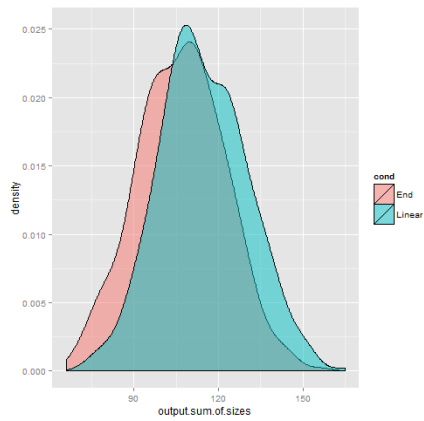


Figure F.9 - Runset 9

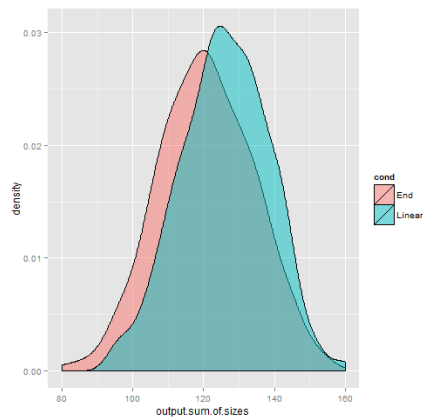


Figure F.12 - Runset 12

APPENDIX G: REWORK DATA FROM INITIAL NETLOGO OUTPUT

Before R could be used to analyze the data, the data is reworked. Reworking the data for this part of the analysis is executed with Notepad ++ and SPSS. This included the following steps:

- Notepad ++ is used to remove the upper 6 rows and subsequently is SPSS used to rework the data.
- subsequently the irrelevant columns are removed. The columns that stays in the document are: runnumber, averagelinkspercarowner, averageevcarlife, initialdieselpercentage and step (renamed to tick), countturtleswithElectricCar (renamed to turtlesEV), countturtleswithPetrolCar (renamed to turtlesPetrol), countturtleswithDieselCar (renamed to turtlesDiesel), buyersEV, buyersPetrol, buyersDiesel
- The cases are sorted on runnumber
- The runnumber is in the case of 4 and 6 averagelinkspercarowner added with respectively 600 or 1200. This is because the three files could not all use the runnumbers 1 – 600. With this transformation, the runnumbers are from 1- 1800 without double runnumbers.
- This file is saved as .csv document exp+average neighbors variable (exp2,exp4 and exp6)

APPENDIX H: R CODE FOR THE ANALYSIS OF THE INITIAL OUTPUT

The following libraries are loaded in R: library(ggplot2), library(akima), library(rgl), library(MASS), library(lattice), library(Defaults), library(foreach), library(PerformanceAnalytics)

The following settings are used:

- setwd("c:/users/bert/desktop/abm output"),
- options(stringsAsFactors=FALSE)

Initial output

read and name the datafile

```
exp2 = read.csv("exp2.csv", "header"=TRUE, "sep"=",", "row.names"=NULL)
exp4 = read.csv("exp4.csv", "header"=TRUE, "sep"=",", "row.names"=NULL)
exp6 = read.csv("exp6.csv", "header"=TRUE, "sep"=",", "row.names"=NULL)
final <- rbind(exp2,exp4,exp6)
```

_MAKING A GRAPH TO SHOW S CURVES____(SINGLE RUN)_____

define the axes

```
amountEV <- final$turtlesEV[1:6000] # the [a:b] define which runset is used. without [a:b] all runs are
  included, 1 run is 6000 ticks therefore 6000 rows
runnumber <- final$runnumber[1:6000] # the [a:b] define which runset is used. without [a:b] all runs are
  included, 1 run is 6000 ticks therefore 6000 rows
tick <- final$tick[1:6000] # the [a:b] define which runset is used. without [a:b] all runs are included, 1 run is
  6000 ticks therefore 6000 rows
```

define color range

```
amountEVlim <- range(amountEV)
amountEVlen <- amountEVlim[2] - amountEVlim [1] + 1 # define the amount of colors
colorlut <- terrain.colors(amountEVlen) # define the gradient type over defined range
col <- colorlut[ amountEV-amountEVlim[1]+1] # color assigning to heights
```

make plot of the first run (single run)

```
plot(x=tick, y=amountEV, col=col, xlab="ticks", ylab="amount of EV owners") ## graph 1
plot(x=tick, y=amountEV, col=col, type="l", xlab="ticks", ylab="amount of EV owners") ## graph 2
```

_MAKING A GRAPH TO SHOW S CURVES____(MULTIPLE RUN)_____

define the axes

```
amountEV <- final$turtlesEV # the [a:b] define which runset is used. without [a:b] all runs are included, 1
  run is 6000 ticks therefore 6000 rows
runnumber <- final$runnumber # the [a:b] define which runset is used. without [a:b] all runs are included, 1
  run is 6000 ticks therefore 6000 rows
tick <- final$tick # the [a:b] define which runset is used. without [a:b] all runs are included, 1 run is 6000
  ticks therefore 6000 rows
```

define color range

```
amountEVlim <- range(amountEV)
amountEVlen <- amountEVlim[2] - amountEVlim [1] + 1 # define the amount of colors
colorlut <- terrain.colors(amountEVlen) # define the gradient type over defined range
col <- colorlut[ amountEV-amountEVlim[1]+1] # color assigning to heights
```

```
## make plot of the first run (multiple run)
  rgl.open()
  open3d()
  lines3d(x=tick, y=amountEV, z=runnumber, color=col)
  axes3d(edges = "bbox", labels = TRUE,
         tick = TRUE, nticks = 5,
         box=TRUE, color=100)
  title3d(main="", sub=NULL,
         xlab="tick", ylab="turtles that own an EV",
         zlab="runnumber",
         color=100)

## Calculate maximum amount of EV owners in the three different groups
  max(final$turtlesEV[1:3600600]) #(3600600 = 10801800/3)
  max(final$turtlesEV[3600601:7201200])
  max(final$turtlesEV[7201201:10801800])
```

APPENDIX I: R CODE FOR THE ANALYSIS OF KOLMOGOROV COMPLEXITY

The following libraries are loaded in R: library(ggplot2), library(akima), library(rgl), library(MASS), library(lattice), library(Defaults), library(foreach), library(PerformanceAnalytics)

The following settings are used:

- setwd("c:/users/bert/desktop/abm output"),
- options(stringsAsFactors=FALSE)

```
### Kolmogorov Complexity
```

```
## read and name the datafile
```

```
output = read.csv("final_output.csv", "header"=TRUE, "sep"=";", "row.names"=NULL)
```

```
## add a column that calculates sum of compressed sizes over all car types
```

```
output["sum.of.sizes"] <- NA
```

```
output$sum.of.sizes <- output$size.EV + output$size.Petrol + output$size.Diesel
```

```
## _ MAKING A 3D GRAPH
```

```
## define x,y,z axes
```

```
run <- output$run[1:24100] # the [a:b] define which runset is used, see data summary in report. without [a:b] all runs are included
```

```
string <- output$string1[1:24100] # see comment run
```

```
compression <- output$size.EV[1:24100] # see comment run
```

```
## define color range
```

```
compressionlim <- range(compression)
```

```
compressionlen <- compressionlim[2] - compressionlim [1] + 1 # define the amount of colors
```

```
colorlut <- terrain.colors(compressionlen) # define the gradient type over defined range
```

```
col <- colorlut[ compression-compressionlim[1]+1] # color assigning to heights
```

```
## open output screen and create 3d graph
```

```
rgl.open()
```

```
open3d()
```

```
lines3d(x=run,y=string,z=compression, color=col)
```

```
axes3d(edges = "bbox", labels = TRUE,
```

```
tick = TRUE, nticks = 5,
```

```
box=TRUE, color=100)
```

```
title3d(main="", sub=NULL,
```

```
xlab="Run", ylab="string",
```

```
zlab="Compression sizes",
```

```
color=100)
```

```
rgl.quit() ## to quit the output of RGL, be aware that library(rgl) is also closed and should be loaded again.
```

```
## _ MAKING 2D PLOTS
```

```
replication <- 1:100 ## start with 1 run and 100 replications
```

```
numberofstrings <- max(output$string1)
```

```
## which rows should be used in calculating file size in linear part of s curve
```

```
first_tick_linear <- 40+replication*numberofstrings ## eventually added a start rownumber to calculate for for different runset
```

```
last_tick_linear <- 60+replication*numberofstrings ## eventually added a start rownumber to calculate for for different runset
```

```

range_linear <- c(c(first_tick_linear [1]):c(last_tick_linear [1]),
  c(first_tick_linear [2]):c(last_tick_linear [2]),
  c(first_tick_linear [3]):c(last_tick_linear [3]),
  c(first_tick_linear [4]):c(last_tick_linear [4]),
  c(first_tick_linear [5]):c(last_tick_linear [5]),
  c(first_tick_linear [6]):c(last_tick_linear [6]),
  c(first_tick_linear [7]):c(last_tick_linear [7]),
  c(first_tick_linear [8]):c(last_tick_linear [8]),
  c(first_tick_linear [9]):c(last_tick_linear [9]),
  c(first_tick_linear [10]):c(last_tick_linear [10]),
  c(first_tick_linear [11]):c(last_tick_linear [11]),
  c(first_tick_linear [12]):c(last_tick_linear [12]),
  c(first_tick_linear [13]):c(last_tick_linear [13]),
  c(first_tick_linear [14]):c(last_tick_linear [14]),
  c(first_tick_linear [15]):c(last_tick_linear [15]),
  c(first_tick_linear [16]):c(last_tick_linear [16]),
  c(first_tick_linear [17]):c(last_tick_linear [17]),
  c(first_tick_linear [18]):c(last_tick_linear [18]),
  c(first_tick_linear [19]):c(last_tick_linear [19]),
  c(first_tick_linear [20]):c(last_tick_linear [20]),
  c(first_tick_linear [21]):c(last_tick_linear [21])
)

```

which rows should be used in calculating file size in end part of s curve

```

first_tick_end <- 150+replication*numberofstrings ## eventually added a start rownumber to calculate for
for different runset

```

```

last_tick_end <- 170+replication*numberofstrings ## eventually added a start rownumber to calculate for
for different runset

```

```

range_end <- c(c(first_tick_end [1]):c(last_tick_end [1]),
  c(first_tick_end [2]):c(last_tick_end [2]),
  c(first_tick_end [3]):c(last_tick_end [3]),
  c(first_tick_end [4]):c(last_tick_end [4]),
  c(first_tick_end [5]):c(last_tick_end [5]),
  c(first_tick_end [6]):c(last_tick_end [6]),
  c(first_tick_end [7]):c(last_tick_end [7]),
  c(first_tick_end [8]):c(last_tick_end [8]),
  c(first_tick_end [9]):c(last_tick_end [9]),
  c(first_tick_end [10]):c(last_tick_end [10]),
  c(first_tick_end [11]):c(last_tick_end [11]),
  c(first_tick_end [12]):c(last_tick_end [12]),
  c(first_tick_end [13]):c(last_tick_end [13]),
  c(first_tick_end [14]):c(last_tick_end [14]),
  c(first_tick_end [15]):c(last_tick_end [15]),
  c(first_tick_end [16]):c(last_tick_end [16]),
  c(first_tick_end [17]):c(last_tick_end [17]),
  c(first_tick_end [18]):c(last_tick_end [18]),
  c(first_tick_end [19]):c(last_tick_end [19]),
  c(first_tick_end [20]):c(last_tick_end [20]),
  c(first_tick_end [21]):c(last_tick_end [21])
)

```

GRAPH SIZE EV _____

creating dataframe with output size

```

data_EVsize <- data.frame(cond=output$string1, output$size.EV)

```

```
## selecting the data (based on identified rows, defined a range)
  frame_linear <- data_EVsize[range_linear,]
  frame_end <- data_EVsize[range_end,]

## labeling data for comparison in graphs
  frame_linear <- transform(frame_linear, cond = "Linear")
  frame_end <- transform(frame_end, cond = "End")

## combine the two dataframes (by adding rows)
frame_linear_end <- rbind(frame_linear,frame_end)

## create plots
  plot <- ggplot(frame_linear_end, aes(x=output.size.EV, fill=cond))
  plot + geom_histogram(binwidth=2, position= "identity", alpha=0.5) ## histogram
  plot + geom_density(alpha=.5) ## density curve

## GRAPH SUM OF SIZES _____ WORKS IDENTICALLY AS ONLY SIZE EV _____
  data_sumsizes <- data.frame(cond=output$string1, output$sum.of.sizes)
  sum_frame_linear <- data_sumsizes[range_linear]
  sum_frame_end <- data_sumsizes[range_end,]
  sum_frame_linear <- transform(sum_frame_linear, cond = "Linear")
  sum_frame_end <- transform(sum_frame_end, cond = "End")
  sum_frame_linear_end <- rbind(sum_frame_linear,sum_frame_end)
  sum_plot <- ggplot(sum_frame_linear_end, aes(x=output.sum.of.sizes, fill=cond))
  sum_plot + geom_histogram(binwidth=2, position= "identity", alpha=0.5) ## histogram
  sum_plot + geom_density(alpha=.5) ## density curve
```