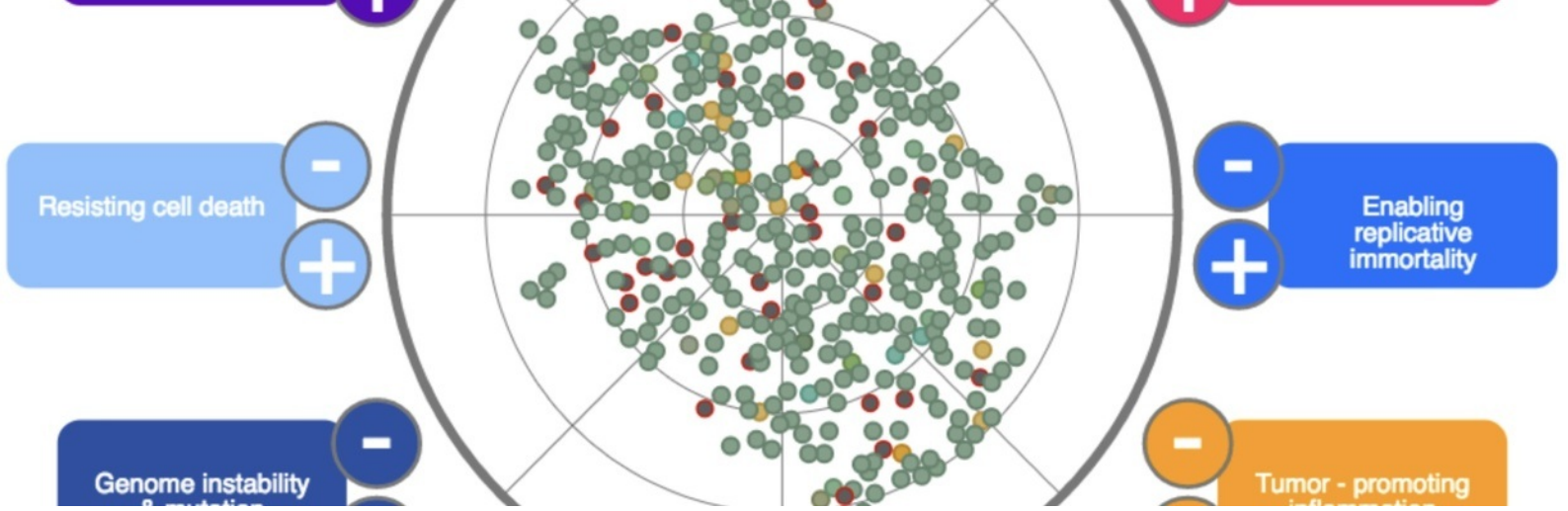


Towards Simulating Carcinogenesis: Modeling and Simulating Carcinogenesis, Hematopoietic Tissue Homeostasis and Leukemogenesis

Authors: Jenny Groten, Maximilian Georg, Oliver Worm, Christoph Borner, Roland Mertelsmann

Submitted: 6. November 2016
Published: 6. November 2016
Volume: 3
Issue: 7

Affiliation: Institut für Molekulare Medizin und Zellforschung

Keywords: Carcinogenesis, Modeling, Simulation, Tissue Homeostasis, Hematopoiesis, Leukemogenesis
DOI: 10.17160/josha.3.7.253

JOSHA
josha.org

Journal of Science, Humanities and Arts

JOSHA is a service that helps scholars, researchers, and students discover, use, and build upon a wide range of content

Institut für Molekulare Medizin und Zellforschung (1)
Medizinische Klinik 1, Universtitätsklinikum Freiburg (2)
Albert-Ludwigs-Universität Freiburg
Psiori GmbH, Freiburg, Deutschland (3)

Towards Simulating Carcinogenesis

Modeling and Simulating Carcinogenesis, Hematopoietic Tissue Homeostasis and Leukemogenesis

Jenny Groten (1)
Maximilian Georg (1)
Oliver Worm (3)
Christoph Borner (1)
Roland Mertelsmann (2)

“A doctrine of nature can only contain
so much science proper as there
is in it of applied mathematics.”
- *Immanuel Kant* (Ernest Belfort Bax 1786)

1 Abstract

The overall aim of this project was to investigate the fundamental phenotypic traits of a cancer cell to develop an “in silico” simulation model and, vice versa redefine the identified characteristics via the established simulation model. Thus, the focus lay on visualization and interactivity of the simulation.

The previously identified hallmark characteristics (Groten et al., 2016, *DOI: 10.17160/josha.3.7.252*) were described by mathematical algorithms. Subsequently, a computational simulation of carcinogenesis has been drawn up employing these mathematical algorithms. In the next step, the proposed algorithms and correlations have been tested, validated and adapted through the simulation in several repetitive phases (http://mertelsmann.psiori.com/).

In a second model, we transferred the novel insights won from the first simulation to the simulation of hematopoietic tissue homeostasis and leukemogenesis (http://hem-model.psiori.com/hema_simulation).

Our findings indicate that the ten “Hallmarks” proposed by Hanahan and Weinberg can be assigned to two different groups, “Growth/Apoptosis Balance” and “Genetic Fidelity/Immortality”, and that carcinogenesis requires just one alteration in each pathway group. Modeling Hematopoiesis revealed one missing Hallmark Capability, “Block of Differentiation”, which we propose to assign to the broader term “Stem Cell Features”.

In summary, we have developed two simulation models, which both depict previous assertions as well as provide novel unexpected, hypothesis generating and possibly underestimated insights and should be increasingly incorporated into prospective oncologic research. This approach promises to contribute to a novel type of evidence and hypothesis generation in cancer research, especially in conjunction with Machine Learning tools, which allow time-lapse experiments, independent self-learning of a system and, thus, full exploitation of computational power.

2 Table of Contents

3 Introduction

This introduction has been previously published (Groten et al., 2016) since it is pertinent to the previous and the current publication.

3.1 Preface

"Cancer is a leading cause of death, and cancer incidence is expected to increase worldwide in the coming decades. But today, cancer research is on the cusp of major breakthroughs. It is of critical […] importance that we accelerate progress towards prevention, treatment, and a cure -- to double the rate of progress in the fight against cancer -- and put ourselves on a path to achieve in just 5 years research and treatment gains that otherwise might take a decade or more." (Barack Obama, January 28, 2016)

With this statement, the President of the United States recently laid the foundations to reenter the fray against cancer (Obama 2016). Calling for a new initiative, headed by Vice-President Biden, he established the "White House Cancer Moonshot Task Force", consisting of members of various departments, to unite researchers, oncologists, patient representatives, economists, politicians and philanthropists in the envisaged revolution of the cancer research landscape (Lowy & Collins 2016). In fact, cancer is still the second-leading cause of death after cardiovascular diseases in Germany (Bundesamt 2014) as well as worldwide (WHO 2016b). Cancer mortality rates apparently have diminished during the last 25 years (Lowy & Collins 2016; IARC 2016b). However, according to the World Health Organization (WHO) the worldwide incidence of cancer is estimated to increase by about 70 % in the next two decades, which means an absolute number of annual cancer cases of up to 22 million instead of 14 million in 2012 (WHO 2016a). In the face of these alarming data, one might assume that an international commitment to cure cancer once and for all is more than overdue. However, in fact, in the long struggle against cancer, it is not the first governmental attempt to raise the cancer issue on a national, and thereby, public and more intense level. On the contrary, it was President Franklin D. Roosevelt who established the National Cancer Institute (NCI) with the help of the National Cancer Act of 1937 (National Cancer Institute

2016a). Later, on December 23, 1971, his successor President Richard Nixon signed into law the National Cancer Act of 1971 to make the “Conquest of Cancer […] a national crusade”. It should broaden the role of the National Cancer Institute (NCI), at that time a part of the National Institutes of Health (NIH), and extended its mandate to the future application of the research results to decrease the incidence, morbidity, and mortality due to cancer (National Cancer Institute 2016b). In Europe, several national and international associations with similar aims were founded. In 1933, the Union Internationale Contre le Cancer (UICC) was founded in Geneva (UICC 2014). Later, the International Agency for Research on Cancer (IARC) was established as the specialized cancer agency of the World Health Organization (WHO) (IARC 2016a). In Germany the Deutsches Krebsforschungszentrum (DKFZ) was formed in 1964 (Deutsches Krebsforschungszentrum 2016). The research objectives of the current campaigns, ranging from cancer vaccines to data sharing and the promotion of innovative and exceptional approaches, obviously reflect the urge to rethink the oncologic research landscape (Lowy & Collins 2016). In the time of big data and information sharing the focus increasingly lies in data collection, analysis, evaluation and implementation. One compelling example is the massive decoding of the human cancer genome through next-generation DNA sequencing (Hayes & Kim 2015). Paradoxically, at first sight, a major shift in patient care takes place at the same time, edging away from Evidence-Based Medicine (EBM) to a Personalized Medicine (PM) (Sugarman 2012a). In PM, patients are individually diagnosed and treated with innovative treatments, far off the beaten track, are given a place. This movement can undoubtedly be rated as a major turning point in the history of cancer research, which has been focused on detecting regularities and defining classifications for a long time. Here, the role of Bioinformatics becomes inevitable to enable big data and PM to go hand by hand allowing novel and innovative perspectives, where traditional EBM has recognized its limitations. Knowledge and data regarding the carcinogenesis process and cancer treatment

have increased dramatically and have finally reached an unmanageable complexity. Common lab experiments and clinical trial tools can no longer provide adequate opportunities to investigate carcinogenesis and cancer treatment as a whole or even its sheer endless number of subunits and possible interactions. A result is the reductionist approach leading to delusive selective insights into complex biologic processes, similar to the parable of the blind monks examining an elephant, who all fail to recognize the creature as a whole, since they only examined a small part each (Fig. 1). This approach, by far, does not satisfy cellular heterogeneity and “noise”, two fundamental characteristics of cellular and system behavior (Walker & Southgate 2009). Bioinformatics provides a solution, implementing mathematical models and information theory, which bring along massive computational power exceeding the limited capacity of the human mind, to address the complexity of data, correlations, and calculations. These so called *in silico* experiments and analyses are time-saving and, nevertheless, encompassing and lead to in-depth results. Thereby, analytical models can be distinguished from simulation models. Analytical models have experienced broad implementation during the last decade, for example, regression analyses have been used for statistical purposes, such as the prediction of affinity profiles of nucleic acid-binding protein from the protein sequence (Pelossof et al. 2015). In contrast, simulation models have hardly received attention, despite a broad and fruitful application in other disciplines, such as engineering, economics or some aspects of biology, where it has become the state-of-the-art solution for understanding and optimizing processes and complex systems (Suthaharan 2016; Sütterlin 2015). In both analytical and simulating model types, *machine learning* can be applied by means of self-teaching systems which are equipped with basic parameters, fundamental conditions and feedback mechanisms to evaluate target parameters. It shows the significant advantage of optimizing a process without an entire knowledge of the actual mechanisms, parameters, and measurements and, therefore, increases the probability of unexpected outcomes (Riedmiller et al. 2009). In comparison to

Figure 1 Blind monks examining an elephant, Hanabusa Itcho, https://en.wikipedia.org/wiki/Blind_men_and_an_elephant, 2016

analytical models, simulation models provide various benefits. Probably, the most essential aspect is the additional dimension of a simulation, *time*. This supplement allows the observation and calculation of a development over time and further enables the evaluation of data at any arbitrary point in time. Moreover, a simulation represents a tool to visualize processes and, thereby, increase comprehension of mechanisms and operations. An additional feature, which seems quite essential in scientific research, is the possibility to alter an ongoing process by adjusting any parameter at any discretionary time and directly achieve tangible results. Regarding the investigation of cell behavior and system interactions, simulations allow a "middle-out" approach, instead of common "top-down" or "bottom-up" models, focusing on the cell, as the "basic unit of life" (Walker & Southgate 2009).
A typical example of the exploitation of the aforementioned simulation features including machine learning tools, is the investigation of evolutionary processes, which are by nature determined by probability and chance and bear a great potential to evolve unpredictable effects (Mertelsmann & Georg 2016).
In sight of the widely accepted hypothesis of carcinogenesis as an evolutionary development (Almendro et al. 2013; Beerenwinkel et al. 2015; Cairns 1975; Klein 2013; Greaves & Maley 2012; Hanahan & Weinberg 2011; Merlo et al. 2006; Nowell 1976; Vogelstein et al. 2013; Willyard 2016) it seems reasonable indeed to establish the use of machine learning, simulation models in particular, in cancer research. Recent emphasis on the pivotal role of chance in the development of malignant diseases (Tomasetti & Vogelstein 2015; Luzzatto & Pandolfi 2015) even fosters the perception of simulation models as the next logical step in the investigation of carcinogenesis.

"It is the quality of our work which will please God and not the quantity."

- Mahatma Gandhi (Alli 2013)

While current cancer research focuses on data generation, the next major step promises to be a view from a meta-level by exploring data analysis, which, hopefully, will lead to better understanding and novel concepts of cancer prevention, diagnosis, control and therapy.

3.2 Aims and Objectives

To address the need for *in silico* simulation models mentioned above, we want to provide a visually attractive and interactive, and at the same time plausible and qualitatively valid simulation model of carcinogenesis and cancer treatment, developed from experimental data. Thereby, the overall objective is to offer a novel tool for basic, clinical and therapeutic research, as well as a teaching tool to make *in silico* research tangible and applicable to a wide audience.

In the present research, the focus lies on the collection and review of relevant data, the formation process of the developed simulations and the first qualitative validation process. In this context, the term *validation* is used to document the close resemblance of the qualitative prediction of the *in silico* simulation and *real-life* biological and clinical data extracted from the relevant literature.

To accomplish this aim, we will address the following research objectives:

1. Identify the essential “Hallmarks of Cancer”, based on literature review. The term “Hallmarks of Cancer” was adopted from Hanahan and Weinberg (Hanahan & Weinberg 2000; Hanahan & Weinberg 2011) and defines the most fundamental phenotypic characteristics of cancer cells, which are assumed to distinguish the latter from normal cells.
2. Consider literature about the history of cancer research, the hallmarks of cancer, presented by Hanahan and Weinberg, the genetic hallmarks with a focus on gene expression, the principles of evolution, entropy and chance in cancer, the basis of cancer treatment and recent research strategies to develop a new concept of phenotypic cancer hallmarks.
3. Evaluate and synthesize the elaborated hallmark capabilities.
4. Develop mathematical models and algorithms to describe the hallmark characteristics of carcinogenesis.
5. Develop a computational simulation of carcinogenesis based on the algorithms.
6. Test, validate and adapt the algorithms and correlations via repetitive simulation phases.
7. Transfer the insights from the first simulation to a simulation of hematopoiesis.
8. Simulate hematopoietic tissue homeostasis, the establishment of hematopoiesis after stem cell transplantation and leukemogenesis.
9. Reevaluate the “Hallmark Concept” in the light of the simulation results.

4 Materials and Methods

4.1 Mathematics and Programming

Mathematical models were developed using the review of pertinent literature. Algorithms and conditions described the mathematical models. These algorithms were then used to program simulations. Programming languages applied here were Python, Java, and JavaScript. For the exact codes, see the Appendix.

4.2 Validation Process

4.2.1 Balancing – Visual Validation

The first validation step, which we performed during the development of the simulation, was a tool called “Balancing” (Schell 2015). Here, “Balancing” defines a strategy normally applied in Game Design and describes a process of iterative observation, testing, and comparison to known evidence and subsequent adaptation of the settings of the simulation, until the resulting processes and developments visually resemble the processes and mechanisms depicted in pertinent literature and seem mainly plausible. This strategy is preferably implemented by several people from different perspectives (Schell 2015).

4.2.2 Regression Analysis – Statistical Validation

The second validation step was a statistical analysis of possible correlations between the different parameters by computational regression analysis. For the analysis, we applied the function for Ridge Regression (Suthaharan 2016; Pedregosa et. al.

2011). Therefore, evolutionary sections of the simulation were eliminated. For the analysis, we parameterized both pathways and defined cell characteristics, so that their correlations could be investigated.
After running a pre-loop, settings with all possible combinations of one, two and three pathways set to the maximum value were simulated for 6000 ticks, which equaled 120 days in real time. To achieve better statistical power, they were repeated six times each. Every simulation round started with six stem cells. All pathways were set to an average level. The pathways to be altered were set to the maximum level. During the simulation, the parameterized cell characteristics were measured at pre-defined points in time. The results were analyzed employing the equation of Ridge Regression (Pedregosa et. al. 2011).

5 Results

5.1 Modeling and Simulation

5.1.1 In Silico Research

5.1.1.1 General

One tool to address both the rising demands for a more encompassing investigation of the processes occurring in malignant diseases as well as for the implication of personalized medicine in oncology is the so-called *in silico* research. The term, describing the application of mathematical models, information theory, machine learning and computer simulation in biological and medical research, is added to the common expressions *in vivo* and *in vitro* and was coined to emphasize the equality of the role of bioinformatics in biomedical sciences in comparison to traditional research methods. It comprises the analysis, visualization, and prediction of natural processes by collecting, cataloging, altering and modeling data employing algorithms and computation (Nature 2016). Computational analysis thereby allows the processing of huge amounts of data as well as the performance of highly theoretical studies and

experiments under precise and selected conditions to avoid disturbance variables and to minimize the complexity of biological events (Mackey et al. 2015).
Mathematically, the benefits of information-theoretical analysis lie in the ability to map multi-parametric processes and systems, both respecting continuous and discrete variables, by the application of measures of “entropy” and “mutual information” (Blokh & Stambler 2016). In contrast, the more common reductionist approach leads to delusive selective insights into complex biologic processes, which do by far not satisfy cellular heterogeneity and “noise”, two fundamental characteristics of cellular and system level behavior (Walker & Southgate 2009).
Information-theoretical methods can be used in different modalities with different consequences and desirable outcomes. First, they can serve as a tool to develop patterns derived from empirical measures and exploit the full information hidden in gathered data. Examples are the construction of probability functions for parameter inference or the examination of correlations between different parameters, for instance, via linear regression analysis (Mackey et al. 2015; Blokh & Stambler 2016; Suthaharan 2016). Second, mathematical models can serve as “proof-of-concept tests” of logical predictions in verbal hypotheses and represent valid tests themselves to evolve further testable quantitative predictions. As verbal hypotheses, based on general assumptions, which are modified deliberately by inclusion and exclusion of certain factors, generally follow a chain of logic to draw conclusions, they provide ample scope for logical errors and oversight. In contrast, the implication of mathematical models facilitates the validation of verbal chains of logic by displaying the assumptions above mathematically. To describe a process to be tested properly, assumptions have to be made explicit. Thus, the precise nomination and characterization of critical assumptions reduce the probability of logical error.
At last, the described methods cannot only abstract complex data, but they can also provide new qualitative data to be further elaborated in empirical research. For instance, unknown or underestimated phenomena can be discovered, or assumptions hidden in a verbal hypothesis can be detected, as the synthetic system, different from a logical chain, can also show counterintuitive outcomes (Mackey et al. 2015).

5.1.1.2 Analytical versus Simulating Models

As mentioned above, so-called *in silico* experiments and analyses allow time-saving and, nevertheless, encompassing and in-depth investigations. Apart from the aforementioned mathematical models, one can distinguish analytical models from simulation models. Analytical models have widely been implemented in biologic and cancer research. For instance, the prediction of affinity profiles of nucleic acid-binding protein from the protein sequence has been achieved via statistical regression analyses (Pelossof et al. 2015). In contrast, despite a strong implication in other fields, like engineering, economics or some aspects of biology, simulation models have barely been applied in oncology. In these disciplines, it has become a common solution for understanding and optimization of processes and complex multi-parametric systems (Suthaharan 2016; Sütterlin 2015). In contrast to analytical models, simulation models provide several advantages. First, the additional dimension of a simulation, time, offers a plethora of observational and analytical possibilities, as well as the capability to interact simultaneously. It enables to observe and calculate a development over time and allows the evaluation of data at any arbitrary point in time. Besides, a simulation provides the ability of visualization and thereby increases the understanding of mechanisms and operations. Additionally, the possibility to interact allows the alteration of ongoing processes by adjusting any parameter at any arbitrary time and directly achieve visible results. In conclusion, simulations thereby provide a “middle-out” approach to the investigation of multiparametric cellular systems, instead of common “top-down” or “bottom-up” models, focusing on the cell, as the “basic unit of life” (Walker & Southgate 2009).

5.1.1.3 Machine Learning

Both analytical and simulating model types can be extended by the application of Machine Learning tools. These tools represent a group of self-teaching systems which work with predefined basic parameters, conditions and feedback mechanisms to optimize a process due to a chosen end point. It provides the benefit that processes can be investigated and optimized without knowledge of the entirety of actual mechanisms, parameters, and measurements. Also, the probability of unexpected outcomes is high due to the non-deterministic prerequisites (Riedmiller et al. 2009).

5.1.1.4 Bioinformatics in Evolutionary Biology

As Evolution is known to be a complex multi-parametric process with a large time scale, mathematical models are widely utilized as a solution to the analysis and simulation of evolutionary developments (Mackey et al. 2015).

Evolution bears various phenomena that, additionally, are connected via various correlations. To reveal the impact of each factor, as well as the related interaction with other factors, one would have to test each evolutionary parameter one at a time in a wet-lab experiment, which is simply impossible to perform in sight of the immense amount of possibilities. Further, it is impossible to reproduce realistic environmental conditions in a lab, as many phenomena exist concurrently. The investigation of real organisms, on the other hand, brings along a lack of abstraction and control in experiments performed to test hypothesized phenomena. In Silico approaches in evolutionary biology represent methods consisting of simulated organisms and populations, which are observed and tested in synthetic experiments. In these experiments evolutionary conditions of the organisms and the evolutionary process, respectively, can be characterized precisely and set perfectly one at a time to test the individual effects on the genome and the organization of both the organism and the population (Batut et al. 2013). This way simulated organisms can be observed during competition and reproduction, while phenomena like “linkage, epistasis, demographic and environmental variability and behavior” (Mackey et al. 2015) are considered. Total control is given by the possibility to apply random, hand-written or former evolved genomes and to set e.g. the mutation rate, the fitness measure or the spatial arrangement of a population. The implication of synthetic experiments provides various advantages, for instance, they can be repeated limitlessly to gain statistical power. Furthermore, one can observe as many generations as necessary and the events, both in genotype and phenotype, can be recorded simultaneously. In synthetic experiments, depictions vary from mathematical functions to graphs, sequences of nucleotides, 2D- and 3D-simulations as well as computer games.

Previous examples of implications of mathematical models in evolutionary processes have led to clarity in areas of the role of sex, the origin of new species (Mackey et al. 2015) and nucleotide sequences and their functions (Batut et al. 2013). Pelossof *et al.* performed an example of machine learning by the development of an algorithm of

“affinity regression” to predict the recognition code of nucleic-acid-binding proteins (Pelossof et al. 2015).
Evolutionary algorithms have also been applied in adversarial games with high branching factors and a non-deterministic outcome to achieve the best possible action sequence in each turn. A method called “rolling horizon evolution” was used to let the controller learn to play on its own, starting from a random population, which is enabled to evolve an offspring by uniform crossover and random mutation in the subsequent (Justesen et al. n.d.).

5.1.1.5 Bioinformatics in Oncology

As the perception of cancer as an evolutionary process is widely spread in the present research landscape (see above), the aforementioned mathematical models and information-theoretical simulations have been applied to oncology in several approaches. Advances worth mentioning are for instance the detection of meta-markers in breast cancer (Blokh et al. 2007), investigations of DNA-methylation processes (De Carvalho et al. 2012), revelation of the role of microRNA and proteins in prostate cancer (Alshalalfa et al. 2013), the examination of gene-gene- and gene-environment correlations in bladder cancer (Fan et al. 2011) and the deviation of transcriptional profiles in cancer cells (Blokh & Stambler 2016). Furthermore, oligo-parametric simulation models have been developed to investigate targeted therapy of cancer, with the possibilities of cell death and Boolean states of mutations to symbolize resistance (Komarova & Wodarz 2016). Two models that played a crucial role as models for the development of our work shall be explained in detail here.
A spatial 3-dimensional model was developed by Waclaw *et al.* in 2015 to elucidate how cell dispersal and turnover could contribute to rapid cell mixing inside a tumor (Fig. 9) (Waclaw, Bozic, Pittman, Hruban, Vogelstein, et al. 2015). They first modeled metastasis as an expansion of cancer cells, which have left their primary site, assumed to have acquired all necessary driver gene mutations in advance. Then they let cells replicate stochastically according to the number of surrounding empty spaces. They found that a cell without any neighboring cells replicates at the maximal rate of

$b = \ln(2) = 0.69^{(-1)}$ (Waclaw, Bozic, Pittman, Hruban, Vogelstein, et al. 2015),

whereas a surrounded cell does not replicate at all.
Assuming that a cell moves with the probability M to a certain place near the surface of the lesion, comparable to short-range migration due to an epithelial-to-mesenchymal transition (EMT), they observed that cells with little dispersal (M=0) build strictly spherical tumors at larger size, while cells with dispersal M>1 derive conglomerates of several small round structures. This outcome proved to be equivalent to observations made in real metastatic lesions, where round tumor structures were found to be divided into groups of non-neoplastic stromal cells and extracellular matrix. So they elucidated the firm correlation between cell dispersal and the growth rate of the tumor, as well as the probability of metastasis.

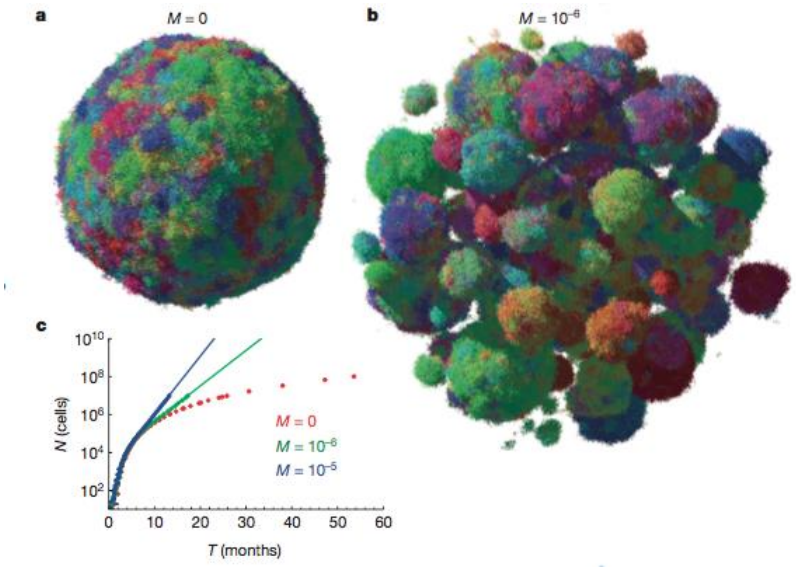


Figure 2 Short-range dispersal affects size, shape and growth rate of tumors, A spatial model predicts that dispersal and cell turnover limit intra-tumor heterogeneity, Waclaw, Bozic, Pittman et al., 2015

Mathematically, they applied the “Eden lattice model” to simulate the combination of genetics, spatial expansion, and short-range cell dispersal. To simplify the simulation, they did not model metabolism, mechanics, spatial tissue heterogeneity, different cell types or angiogenesis. A tumor was modeled by a group of non-overlapping balls or microlesions of cells. Each evolving cell was associated with a certain position and a list of genetic alterations over time, whereby it was differentiated between passenger, driver and resistance-carrying mutations. Driver mutations were modeled to increase the growth rate by deregulating cellular divisions and give an advantage to increased proliferation and decreased apoptosis. In a cell division, each daughter cell is provided with n new genetic alterations of each type, with n being different in both cells and randomly calculated from the Poisson probability distribution. The vast

majority of their results was consistent with experimental findings, which emphasized the applicability and validity of mathematical models for biological processes (Waclaw, Bozic, Pittman, Hruban, Nowak, et al. 2015).
Mertelsmann and Georg provided a distinct approach with the help of a virtual game called “Mitosis”. To address a wider audience, they modeled an interactive simulation to provide a tool to describe and actively model evolutionary processes, which can eventually lead to malignancy as well. For simplicity, they reduced the relevant cell mechanisms to ten fundamental intrinsic parameters called “Hallmarks of Evolution” and six external environmental parameters (Tab. 4).

“Hallmarks of Evolution”	Environmental Parameters
Reproduction	**Oxygen (O2)**
Regeneration	**pH**
Energy Store	**Temperature**
Absorption	**Nutrition**
Agility	**Mutation Rate**
Interaction	**Cytokines**
Attack	
Defense	
Lifetime	
Receptor Sensitivity	

Table 1 “Hallmarks of Evolution” and Environmental Parameters, Cancer: Modeling evolution and natural selection, the „Mitosis Game“, Mertelsmann & Georg, 2016

The whole simulation is based on evolutionary algorithms. The controller can adjust different tools, both altering the intrinsic and the environmental conditions, to grow a cell population and observe cell progression (Fig. 10).

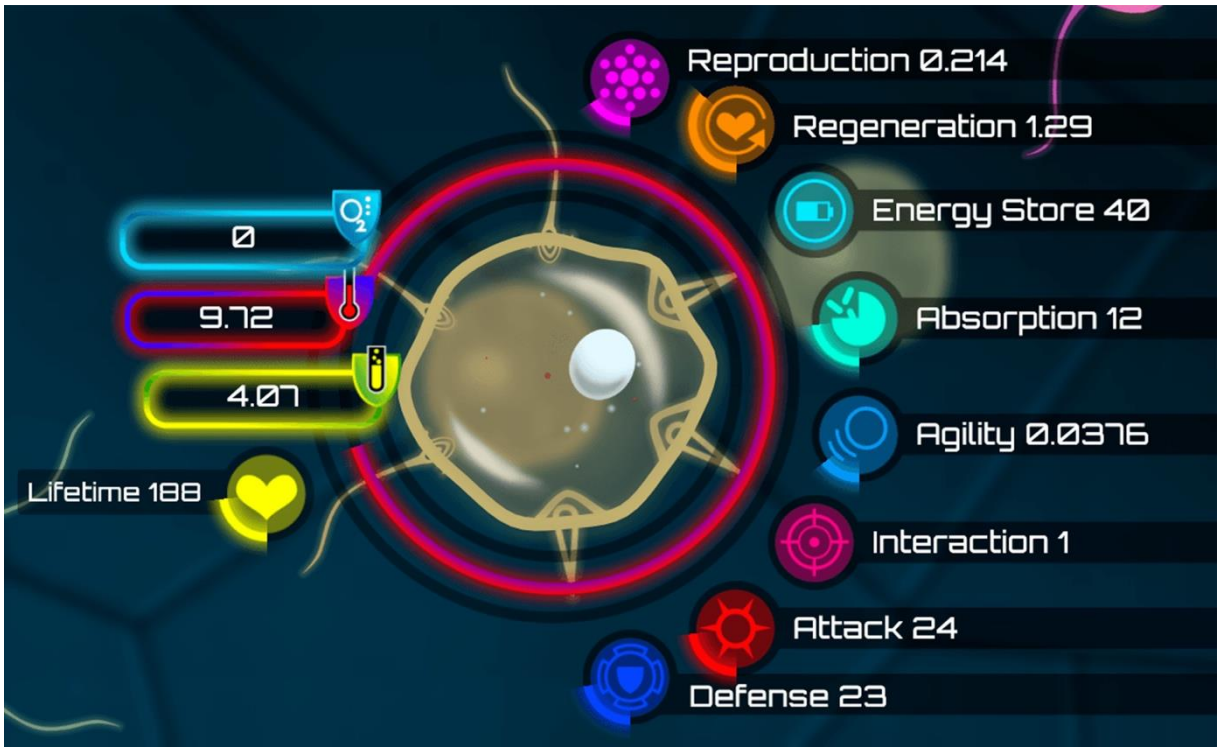


Figure 3 Surface of the Mitosis Game, Cancer: Modeling evolution and natural selection, the "Mitosis Game", Mertelsmann & Georg, 2016

The novelty as compared to previous models is the fact that the genome of a cell cannot be changed directly, but can only be influenced by alterations in the parameters, which have an impact on the probability of cell survival. This way, the simulation follows the rules of random mutation and natural selection (Mertelsmann & Georg 2016).

5.1.2 Towards Simulating Carcinogenesis

5.1.2.1 Approaches – Primary Models

The first versions of our simulation of carcinogenesis are primarily based on the model of the aforementioned "Mitosis-Game" by Mertelsmann and Georg (Mertelsmann & Georg 2016). The main aim is to ease control and handling of the former rather complex and undetermined game, which was previously supposed to address both gamers and scientists while maintaining the capability of interaction to appeal to a more scientific target audience. The new simulations are meant to serve as an educational tool as well as an approach to allow a broad range of scientists to explore the applicability of computational models in oncology, concerning both basic research and clinical trials. Therefore, the focus of the recent models lies in the

improvement of comprehensibility and clarity by reduction of complexity and increase of transparency of the simulated processes.

5.1.2.2 Simulating Cancer Treatment Response

The first approach consists of a simulation of tumor growth, treatment response, and resistance. On the surface of the simulation, there is a petri dish containing continuously growing cells, representing a malignant cell population. Cell characteristics and behavior can be influenced by ten different targeted therapies. Their application can be controlled by adjusting the related "+" and "-" buttons placed around the petri dish, with "+" increasing the allocated therapeutic dosage and "-" decreasing the assigned therapeutic dosage. Each targeted therapy is accompanied by a description of the attacked cellular pathway, the so-called "Hallmark", frequently altered in a malignant cell (Fig. 11) (Hanahan & Weinberg 2000; Hanahan & Weinberg 2011). In the beginning, the dosage of each drug is set at an average concentration to represent a steady state of malignant growth with all pathways partially mutated in equal parts.

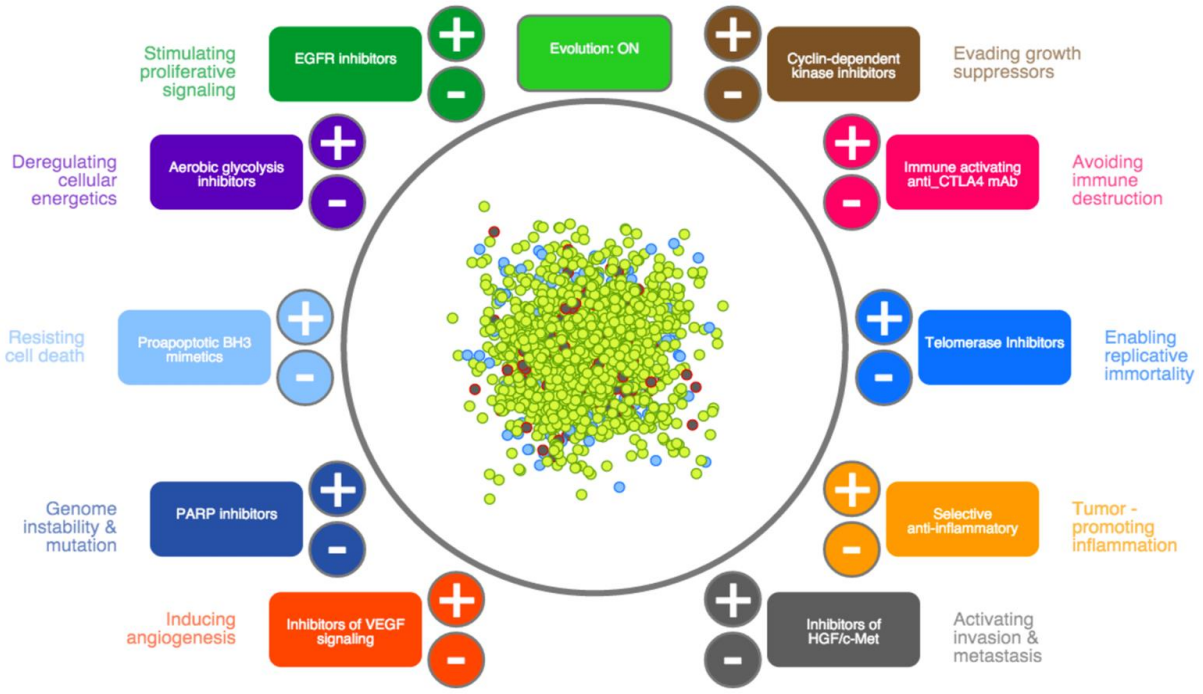


Figure 4 Surface of the Simulation of Cancer Therapy, Georg, Groten, Mertelsmann et al., 2016

The concrete therapies considered in this simulation are listed in *Table 5*:

Targeted Therapy	Cellular Pathway
Cyclin-Dependent Kinase Inhibitor	Evading Growth Suppressors
Immune-Activating anti_CTLA4 mAB	Avoiding Immune Destruction
Telomerase Inhibitor	Enabling Replicative Immortality
Selective Anti-Inflammatory Derivative	Tumor-Promoting Inflammation
HGF/c-Met Inhibitor	Activating Invasion and Metastasis
VEGF Inhibitor	Inducing Angiogenesis
PARP Inhibitor	Genome Instability and Mutation
Pro-Apoptotic BH3 Mimetic	Resisting Cell Death
Aerobic Glycolysis Inhibitor	Reprogramming Cellular Energetics
EGFR Inhibitor	Stimulating Proliferative Signaling

Table 2 Targeted Therapies and related Cellular Pathways, Georg, Groten, Mertelsmann et al., 2016, Hanahan & Weinberg, 2011

For further evidence about each registered Cellular Pathway, see paragraph “The Hallmarks of Cancer” (Hanahan & Weinberg 2000; Hanahan & Weinberg 2011). Moreover, randomly acquired resistance to distinct therapeutic drugs can be activated via an “Evolution”-Button above the petri dish, choosing between the states “on” and “off”. This way, cellular behavior and treatment response can be observed in the case of randomly acquired resistance.

5.1.2.2.1 Logical Background and Programming

The whole simulation is written in JavaScript and executable in every common web browser. Easel.js is applied as an additional library.

At the beginning of the simulation, there is one single cell, which subsequently divides into two, one new daughter cell and one renewed parent cell. Each evolving cell in the simulation exhibits six different characteristics: Color, Position, Velocity, Direction Vector, Hayflick-Limit and a Boolean State (YES/NO-State) of Immune Attack. For each newly built cell, these characteristics are tested and calculated in a chronological order depending on the concentration of therapeutic drugs, which will be further elucidated below. First, the features themselves, their biological relevance and their calculation will be explained.

Color

Each viable cell is colored green.

Further color coding is used for the state “Imminent Apoptosis” (light blue) and the state of “Immune Attack” (dark gray/red contour), which will be depicted below.

Position

The simulation interface is defined via a two-dimensional coordinate system. The fist cell always starts at the center of the visible part of the coordinate system. The Cell Position of each evolving cell is calculated taking into account the former position of the parent cell. The new daughter cell will be placed at a spot around the parent cell, at a random angle in the distance of the diameter of a cell between the centers of the cells. In the current simulation, this happens without considering the availability of space. As a result, cells frequently pile up on top of one another, whereby the younger cell is placed on top of the former cells.

Velocity

The speed of a cell is defined as pixel per tick. It ranges from 0.1 to 1 and depends on the value of the HGF/c-Met Inhibitor which decreases the capacity to invade and metastasize. At the starting point, the speed of a cell is 0.5 pixel/tick. After each tick, the speed of a cell decreases exponentially, till it runs towards zero.

Direction Vector

The Direction Vector is a vector two, which determines the direction of cell movement. It is allocated randomly to every evolving cell and remains unchanged for the whole lifetime of the cell.

Hayflick-Limit

The Hayflick-Limit defines the number of possible divisions of a cell. It depends on the length of the chromosomal telomeres, which decreases in a standard cell with every cell division. To fasten and ease the processes observed in the simulation the default Hayflick-Limit is determined much smaller than in reality. In the simulation, the default Hayflick-Limit of a normal stem cell is 5 in contrast to the realistic number of 72, 50 to 70 respectively (Shay & Wright 2000). If a cell shows a Hayflick-Limit of 1 or

less at the time of testing, it is marked with the color light blue and will die in the next tick in accordance to apoptosis.

Immune Attack

Another state, a cell can show, is the state of being recognized and attacked by the immune system. This state is a Boolean state. Proposing that once a cell has been detected as a target cell, it will be eliminated by the immune system, a cell in this irreversible state is coded with the color dark gray with a red contour and will be dead after ten ticks. The probability of Immune Attack depends on two targeted therapies. First, the concentration of “Immune-activating anti_CTLA4 mAB” determines the interval and thereby the likelihood of testing. The test interval ranges between 90 and ten ticks. Second, the concentration of “PARP inhibitors” further alters the probability of testing with a range of 1 to 10 %, presuming that the probability of detection by the immune system rises with the likelihood of mutation.

The Targeted Therapies

Each therapeutic drug is provided a particular mode of operation, defined by intervals and probabilities, strictly proportionally to the size of the related “drug-button”.
The specific ways of operation are listed in *Table 6*:

Targeted Therapy	Mode of Operation	Values (you may assume that the values between highest and lowest dosage are roughly interpolated)
Cyclin-Dependent Kinase Inhibitor	Calculated probability (p) of proliferation	Lowest dosage: p=1 Highest dosage: p=0,25
Immune-Activating anti_CTLA4 mAB	Calculates interval duration within which the probability of attack and death by the immune system is calculated	Lowest dosage: duration = 90 ticks Highest dosage: duration= 10 ticks

Telomerase Inhibitor	Manipulates Hayflick limit (the simulations default value is 5 instead of 50-70, due to limited processing power of current version)	Lowest dosage: limit = 15 divisions Highest dosage: limit = 1 division
Selective Anti-Inflammatory Derivative	Slightly decreases probability of proliferation	Lowest dosage: reduction = 0,04 Highest dosage: reduction = 0,06
HGF/c-Met Inhibitor	Determines Cell Speed (v)	Lowest dosage: v = 1 pixel/tick Highest dosage: v = 0,1 pixel/tick
VEGF Inhibitor	Not in use	
PARP Inhibitor	Calculates probability of attack and death by immune system	Lowest dosage: p = 0,01 Highest dosage: p = 0,1
Pro-Apoptotic BH3 Mimetic	Not in use	
Aerobic Glycolysis Inhibitor	Not in use	
EGFR Inhibitor	Calculates duration of the interval in which probability of proliferation is calculated	Lowest dosage: duration = 10 ticks Highest dosage: duration = 90 ticks

Table 3 Algorithms of Targeted Therapies, Georg, 2016

5.1.2.2.2 The Simulation Process

Every simulation round has a defined starting point, both temporal and local. The time unit of the simulation is the “tick”, which defines one update loop. 50 ticks in the simulation equal one day in real time. The duration of one “tick” thereby depends on

the power of the processor, which should usually result in about 50 ticks per second. One update loop consists of chronologically determined assessment and subsequent consequences. These conditional links underlie distinct test mechanisms, algorithms, mainly if-then-instructions, and probability ranges.

So, at the beginning of the simulation, when the program starts, a first cell evolves containing the characteristics mentioned above, calculated by the initial settings.

In each following tick, the below-mentioned assessment process is performed.

1) If the “Evolution”- Button is set “on”, a countdown, starting at “50”, is decremented at each tick. If it becomes “0”, a random drug is set at the lowest dosage or probability, symbolizing randomly acquired resistance to one of the targeted therapies.

   During the period a drug button is muted, the user cannot adjust it manually. At the end of the time of 50 ticks, the mute is removed, and another random drug is set to the lowest dosage or probability.

2) Each cell is tested individually concerning divisibility.

   First, only cells not evolved in this tick can go through cell division.

   Second, the event of a division depends on the current EGFR-Inhibitor related Interval. Only if the predetermined Interval is expired, so that the value is “0”.

   The probability that a division actually occurs is further determined by the dosage of the Cyclin-Dependent-Kinase-Inhibitor, which is presumed to have an impact on growth control. The probability ranges between 0.5 and 1. So the higher the Inhibitor-dosage, the less probable is a cell division. If the Inhibitor-dosage is set at the lowest, cell division is always successful. A fourth parameter which influences the probability of cell division is the Selective-Anti-Inflammatory Derivative.

   The higher the Inhibitor dosage, the less probable is the occurrence of inflammation, the less probable is a cell division. The underlying hypothesis postulates that tumor-promoting inflammation supports tumor growth by increasing the cell division rate (Coussens & Werb 2002).

   Also, cells can only divide, if the maximum of living cells does not exceed 1800 at the time of testing.

   This maximum is chosen to both simulate limited space and resources, as well as keep the simulation clear and manageable since the power of a conventional computer is limited.

3) If new cells evolve by the division of mother cells, they receive the characteristics described above. Thereby, Speed is only attributed to the newly arisen cells, whereas the rest of the qualities is given to all apparent cells at the time of testing.
4) The probability of the cell to be attacked by the immune system is calculated. It depends on the Interval determined by the Immune-Activating anti_CTLA4 mAB and the probability alteration according to the concentration of the PARP Inhibitor. For the exact probability calculation see paragraph “Immune Attack”.
5) The Hayflick-Limit of the cell is tested. If it is 1, the cell is turned light blue and will be eliminated at the next division.
6) In the last step, the probability of cell death is calculated. A cell dies, if it either placed outside of the petri dish, if its Hayflick-Limit counts less than 1, or if the ticks determined via “Immune Attack” reach 0 ticks.

In the background of the simulation, a list of all living cells is created simultaneously. In this list, each cell is considered a particular variable or object, which has a given position, determined by its consecutive appearance, and is saved with its individual characteristics.

5.1.2.3 Simulating Carcinogenesis based on the “Hallmarks of Cancer”

Since the main aim of a simulation of carcinogenesis and cancer treatment was both visualization and interactivity, the first approach depicted above was still far too complex and opaque for visual validation. As a consequence, the treatment option was eliminated, and instead, cancer growth was simulated taking into account the impact of the different “Hallmarks” of cancer (Hanahan & Weinberg 2000; Hanahan & Weinberg 2011). These “Hallmarks” define the phenotypic characteristics found to be shared by the vast majority of malignant tumors, each resulting from an altered cellular pathway. This way, the user can influence cell behavior and tumor progress instantly by adjusting so-called “Hallmark”-Buttons (instead of “Targeted Therapy”-Buttons) via “+” and “-”, analogously to the handling of the first model focusing on therapeutic interventions. “+” stands for a greater probability of alteration of the related pathway, whereas “-” decreases the likelihood of change.
The interface, as well as the central logical mechanisms of the new simulation, are the same as in the first one, described in detail above (Fig. 12).

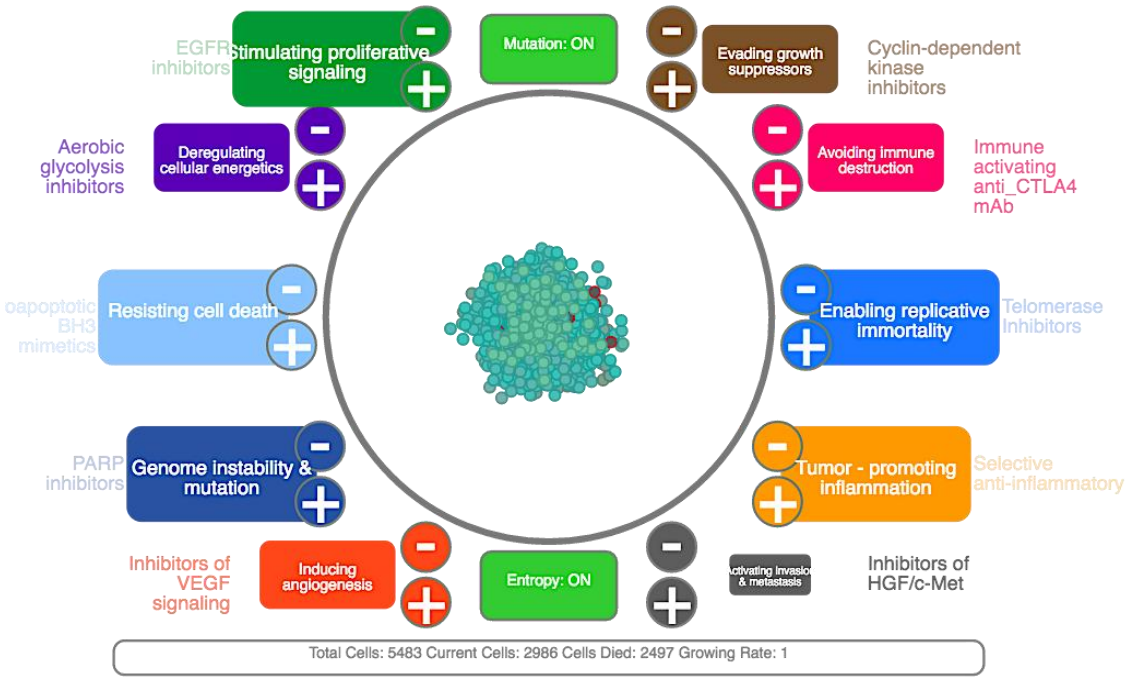


Figure 5 Surface of the Simulation of Carcinogenesis I, Georg, Groten, Mertelsmann et al., 2016, Hanahan & Weinberg 2011

Due to higher transparency and closer resemblance to wet-lab experiments, the previously developed pseudo-3D-surface of the simulation was altered to a 2D-visualization, which allows the visibility of every single cell and its behavior (Fig. 13).

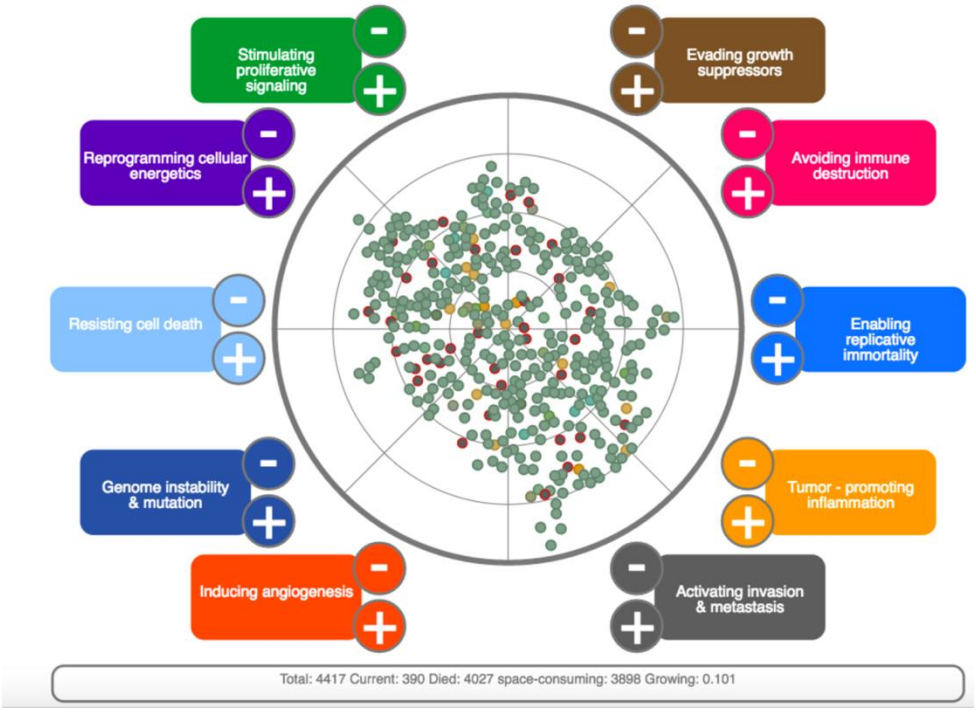


Figure 6 Surface of the Simulation of Carcinogenesis II, Georg, Groten, Mertelsmann et al., 2016

The full simulation model is provided via http://mertelsmann.psiori.com/. In this current version, spectral colors are programmed but do not show high contrast. The process of cell rise and division stays the same. Each evolving cell is equipped with the six characteristics as mentioned above. But, instead of the “Drug”-Buttons applied in the first simulation, “Hallmark”-Buttons influence these features and the resulting cell behavior and tumor growth. For further detail, see *Table 7*.

“Hallmarks of Cancer” (Hanahan & Weinberg 2000; Hanahan & Weinberg 2011)
Evading Growth Suppressors
Avoiding Immune Destruction
Enabling Replicative Immortality
Tumor-Promoting Inflammation
Activating Invasion and Metastasis
Inducing Angiogenesis
Genome Instability and Mutation
Resisting Cell Death
Reprogramming Cellular Energetics
Stimulating Proliferative Signaling

Table 4 “Hallmarks of Cancer”, Hanahan & Weinberg 2000, Hanahan & Weinberg 2011)

The former called “Evolution”-Button, is renamed “Mutation”-Button to achieve a more accurate depiction of the adjusted process, but works in the same way. Moreover, an additional “Entropy”-Button, which can be switched “on” and “off”, allows the simulation of the development of a progressively malignant tumor with increasing amount of mutated pathways over time. Maximally three pathways can be mutated at once. If a fourth pathway is mutated, the earlier one is ignored. In the last version (Figure 12) this feature was abandoned because it limited the modes of action concerning probability and chance in carcinogenesis.

5.1.2.3.1 Logical Background and Programming

The whole simulation is written in JavaScript and executable in every common web browser. Easel.js is applied as an additional library for visualization purposes.
At the beginning of the simulation, there is an arbitrary number of start cells, adjustable in code (currently set at 6 to prevent the system from early death because of the death of the first and only cell), which subsequently divides into two, one new daughter cell and one renewed parent cell. Each evolving cell in the simulation is equipped with six different characteristics: Color, Position, Velocity, Direction Vector, Hayflick-Limit and a Boolean-State of Immune Attack. For each newly built cell, these components are tested and calculated in a certain chronological order depending on the intensity of alteration of the cellular pathways. Below, the particular characteristics are described and the differences compared to the first simulation are elucidated.

Color

Each pathway is represented by a spectral color value, which is calculated as the weighted sum of the color values (RGB, vector 3 with values from 0 to 255) of all pathways, depending on their percentage of mutation (a state between 1 and 5). This calculated value builds the primary color of a cell. This calculation is inspired by a work of Weber et al. (Weber et al. 2011).
Further color coding is used for the state “Imminent Apoptosis” (light blue) and the state of “Immune Attack” (dark gray/red contour).

Position

The simulation interface is defined via a two-dimensional coordinate system. The fist cells are placed at approximately equal distances from the center point of the visible part of the coordinate system. The Cell Position of each evolving cell is calculated taking into account the former position of the parent cell. The new daughter cell will be placed at a spot around the parent cell, at a random angle in the distance of the diameter of a cell.

Velocity

The speed of a cell is defined as pixel per tick. It ranges from 0.1 to 1 and depends on the value of “Activating Invasion and Metastasis”, which increases the capacity to invade and metastasize. At the starting point, the speed of a cell is 0.5 pixel/tick.

After one tick, the speed of a cell decreases exponentially over time, till it runs towards zero.

Direction Vector

The Direction Vector is a two-coordinate vector, which determines the direction of cell movement. It is allocated randomly to every evolving cell and remains unchanged for the whole lifetime of a cell.

Hayflick-Limit

The Hayflick-Limit defines the number of possible divisions of a cell. It depends on the length of the chromosomal telomeres, which decreases in a standard cell with every cell division. In the simulation, the default Hayflick-Limit of a normal stem cell is 72 as an approximation of the realistic number between 50 and 70 (Shay & Wright 2000). Each cell evolving from cell division is assigned the Hayflick-Limit of its predecessor cell minus 1. If a cell shows a Hayflick-Limit of 1 or less at the time of testing, it is marked with the color light blue and will die at a Hayflick-Limit of 0 or less, in accordance to apoptosis.

Immune Attack

Another state, a cell can show, is the state of being recognized and attacked by the immune system. This state is a Boolean-state. Proposing that once a cell has been detected as a target cell, it will be eliminated by the immune system, a cell in this constant state is coded with the color dark gray with a red contour and will be dead after 40 ticks. The probability of Immune Attack depends on two “Hallmarks” (Hanahan & Weinberg 2000; Hanahan & Weinberg 2011). First, the intensity of “Avoiding Immune Destruction” determines the interval and thereby the probability of testing. The test interval ranges between 90 and ten ticks. Second, the intensity of “Genome Instability and Mutation” further alters the likelihood of testing with a range of 1 to 10 %, presuming that the probability of detection by the immune system rises with the number of mutation.

The “Hallmarks of Cancer” (Hanahan & Weinberg 2000; Hanahan & Weinberg 2011)

“Hallmarks of Cancer”	Mode of Operation	Values (you may assume that the values between highest and lowest dosage are roughly interpolated)
Evading Growth Suppressors	Calculated probability (p) of proliferation	Lowest dosage: p=0,25 Highest dosage: p=1
Avoiding Immune Destruction	Calculates interval duration within which the probability of attack and death by the immune system is calculated	Lowest dosage: duration = 10 ticks Highest dosage: duration= 90 ticks
Enabling Replicative Immortality	Manipulates decrease of the Hayflick-Limit per cell division (the simulations default value is 72)	Lowest dosage: decrease = 0 Highest dosage: decrease = 1,5
Tumor-Promoting	Slightly decreases	Lowest dosage: reduction

Inflammation	probability of proliferation	= 0,06 Highest dosage: reduction = 0,04
Activating Invasion and Metastasis	Determines Cell Speed (v)	Lowest dosage: v = 0,1 pixel/tick Highest dosage: v = 1 pixel/tick
Inducing Angiogenesis	Calculates probability of death resistance*	Lowest dosage=0 Highest dosage=1
Genome Instability and Mutation	Calculates probability of attack and death by immune system	Lowest dosage: p = 0,1 Highest dosage: p = 0,01
Resisting Cell Death	Calculates probability of death resistance*	Lowest dosage=0 Highest dosage=1
Reprogramming Cellular Energetics	Calculates probability of death resistance*	Lowest dosage=0 Highest dosage=1
Stimulating Proliferative Signaling	Calculates duration of the interval in which probability of proliferation is calculated	Lowest dosage: duration = 90 ticks Highest dosage: duration = 10 ticks

*Table 5 Algorithms of the "Hallmarks of Cancer", Georg, 2016, Hanahan & Weinberg, 2011. *mean value of all three values is calculated.*

An overview over the modes of action and the related interactions is given by the table in *Figure 14*:

Pathway - Feature Relations

	Evading growth suppressors	Avoiding immune destruction	Enabling replicative immortality	Tumor-promoting inflammation	Activating invasion & metastasis	Inducing angiogenesis	Genome instability & mutation	Resisting cell death	Reprogramming cellular energetics	Stimulating proliferative signaling	Comments static values can only be changed in code
position											Predefined by mother cell, new cells are always placed next to mother cell
initial speed					1						
current speed											Initial speed decreases over time (static factor)
speed reduction											static factor (see current speed)
color	2	2	2	2	2	2	2	2	2	2	
reproduction interval										3	
reproduction probability	4										
hayflick limit											Initial cells have a hayflick limit of 72
hayflick modifier			5								
immune attack interval		6									
immune attack probability							7				
immune attack duration											static value (100 ticks, equivalent to 2 days)
death resistance probability						8		8	8		

1. The expression of „Activating invasion & metastasis" raises the initial speed of a cell after proliferation to enforce wider spread thus thinner population.
2. Cell color is calculated through weighted average of gene expression and the related colors.
3. The expression of „Stimulating proliferative signaling" shortens the interval between probability checks for proliferation.
4. The expression of „Evading growth suppressors" increases the probability of proliferation.
5. The expression of „Enabling replicative immortality" decreases the value thats subtracted from a cell's current hayflick after proliferation. On maximum expression hayflick limit doesn't change.
6. The expression of „Avoiding immune destruction" extends the interval between probability checks for immune attacks.
7. The expression of „Genome instability & mutation" increases the probability for immune attacks, thus it is the only pathway whose expression has a clearly negative influence on the tumor's survivability.
8. The average expression of „Inducting angiogenesis", „Resisting cell death" and „Reprogramming cellular energetics" increases the probability of death resistance. By now all pathways are treated equally important.

Figure 7 Overview over the modes of action and interactions of pathways and cell characteristics, Georg, 2016

5.1.2.3.2 The Simulation Process

The Simulation Process is only slightly altered as well, compared to the first model. Every simulation round has a defined starting point, both temporal and local. The time unit of the simulation is the “tick”, which defines one update loop. 50 ticks in the simulation equal one day in real time. The duration of one “tick” thereby depends on the power of the processor, which should usually result in a number of about 50 ticks per second. One update loop consists of chronologically determined assessment and subsequent consequences. These conditional links underlie distinct test mechanisms, algorithms, mainly if-then-instructions, and probability ranges.
So, at the beginning of the simulation, when the program starts, a first cell evolves containing the characteristics mentioned above, calculated by the initial settings.
In each following tick, the below-mentioned assessment process is performed.

1) The current size of the pathway buttons is measured, and their proportion of the entirety of pathway buttons is evaluated. As a result, the core color of the evolving cell (or cells) is calculated by the proportional summation of the distinct color values (Weber et al. 2011).
2) If the “Mutation”- Button is set “on”, a countdown, starting at “50”, is decremented at each tick. If it becomes “0”, a random pathway is set at the

highest dosage or probability, symbolizing randomly acquired mutations of one of the cellular pathways.

During the period a pathway button is set to the maximum, the user cannot adjust it manually.

At the end of the time of 50 ticks, the automatically set adjustment is removed, and another random pathway is fixed at the highest dosage or probability.

In case that the “Entropy”-Button is switched on, pathways, which have once been altered during the current round of the simulation, which have been saved in a background list, are considered permanently altered. Maximally three pathways can be mutated at once. If a fourth pathway is mutated, the earlier one is ignored. This way, the entirety of occurred alteration is accumulated over time.

In the latest version of the simulation, this option was abandoned due to the insufficient validity of the mode of action. It shall be reintroduced at a later point in time with an adequate algorithm.

3) Each cell is tested individually concerning divisibility.

First, only cells not evolved in this tick can go through cell division.

Second, the event of a division depends on the Interval determined by the current state of the pathway “Stimulating Proliferative Signaling”. Only if the predetermined Interval is expired, so that the value is “0”. The probability that a division occurs, is further determined by the dosage of the ability of a cell to evade “Growth Suppressors”, which is presumed to have an impact on growth control. The probability ranges between 0.5 and 1. So the higher the Probability of Evasion from Growth Suppressors, the more probable is a cell division. A fourth parameter which influences the probability of cell division is the “Tumor-Promoting Inflammation”.

The higher the Probability of Inflammation, the more probable is a cell division. The underlying hypothesis postulates that tumor-promoting inflammation supports tumor growth by increasing the cell division rate (Hanahan & Weinberg 2011).

If new cells evolve by the division of mother cells, they are equipped with the characteristics described above. Thereby, qualities are given to all new cells at the time of testing according to the currently set parameters, except for the Hayflick-Limit and the Position, which is predefined by the mother cell.

4) In the new 2D simulation, an improvement technique is applied to avoid cell stacking and only to permit cell placement where there is free space available. This technique consists of a collision process, in which it is tested at each tick if the distance between the centers of two cells has a minimal value of 1.2 x r with r = radius of a cell. This way every cell is compared to every other cell existing in the current process (listed in the table of all cells). If two cells do not fulfill this criterion, the compared cell dies, so that the new cell survives. This way, cells cannot pile up anymore, only a small percentage of overlap is permitted. So, by a selective mechanism, in areas with more available space, i.e. at the edge of the tumor, more cells can grow.
5) The probability of the cell to be attacked by the immune system is calculated. It depends on the Interval determined by the ability to avoid Immune Destruction and the probability alteration according to the likelihood of "Genome Instability and Mutation". For the exact probability calculation see paragraph "Immune Attack".
6) The Hayflick-Limit of the cell is tested. If it is 1, the cell is turned light blue and will die after the next division.
7) In the last step, the probability of cell death is calculated. A cell is eliminated, if it either placed outside of the petri dish, if its Hayflick-Limit counts less than 1, or if the ticks determined via "Immune Attack" reach 0 ticks.

In the background of the simulation, a list of all living cells is created simultaneously. In this list, each cell is considered a particular object, which has a given position, determined by its consecutive appearance, and is saved with its specific characteristics. As a novel feature to improve transparency for a more scientific use, a "lab report bar" is added to the surface of the simulation, which shows the current numbers of total cells, current cells, dead cells and the growth rate.

5.1.2.3.3 The Validation Process

To implement the suggested simulation in oncology, a validation of the depicted processes is inevitable. The Validation Process was approached in two phases.

1) Balancing

The first step proceeded during the development of the simulation via a tool called "Balancing" (Schell 2015). "Balancing" describes a strategy frequently applied in

Game Design and represents a process of iterative observation, testing and comparison to literal evidence and subsequent adjustment of the settings of the simulation, until the resulting processes and developments visually approach the processes and mechanisms depicted in pertinent literature. This strategy is preferably pursued by several people from different perspectives (Schell 2015). In our case, the literal primary basis was the review above by Hanahan and Weinberg (Hanahan & Weinberg 2011), which was complemented by the present literature review on carcinogenesis. The iterative “Balancing” was performed by a team of experts in the fields of Game Design, Information Theory, Cognitive Science, and Medicine/Oncology.

2) Regression Analysis

The second phase contained a statistical analysis of individual correlations by computational regression analyses. More precisely, the function for Ridge Regression was applied (Suthaharan 2016; Pedregosa et. al. 2011).
Since the evolutionary sections of the simulation turned out to be insufficient for this type of analysis, they were eliminated here.
For the analysis, both the pathways and certain cell characteristics were parameterized, so that their correlations could be investigated.
After running a pre-loop, every possible combination of one, two and three pathways was simulated for 6000 ticks, which equals 120 days in real time. For better statistical power, they were repeated six times each. Thereby, the standard deviations as fractions of the means were 1.23% for the results of “currentCellMax”, 4.08% for “allCells”, 5.61% and “ImmuneAttacked”. Detailed standard deviation values are provided in *Table 13* in the Appendix. Every simulation round started with six stem cells. All pathways were set to an average level, the pathways to be altered were set to the maximum level. During the simulation, the parameterized cell characteristics were measured at pre-defined points in time. Using this approach, 175 possibilities plus one standard constellation were tested, six times each so that 1050 experiments were performed. All tests taken together correspond to 365 years in real time.
The results were analyzed utilizing the equation of Ridge Regression (Pedregosa et. al. 2011). Out of the immense amount of generated data, we focused on the results concerning the following aspects, considering these to be the most robust parameters to validate the correctness of the simulation of tumor growth and

proliferation: the maximum number of current cells, the maximum number of all cells over time, and the maximum number of cells killed by the immune system. All of these showed both valid results as well as unexpected correlations and outcomes. The term “unexpected” here means not directly determined by the code.
For manageability purposes, abbreviations were introduced for the pathways described above. *Table 9* gives an overview over these abbreviations.

Abbreviation	Pathway
signaling	Sustaining Proliferative Signaling
energetics	Reprogramming Cellular Energetics
deathresistance	Resisting Cell Death
instability	Genome Instability and Mutation
angiogenesis	Inducing Angiogenesis
metastasis	Activating Invasion and Metastasis
inflammation	Tumor-Promoting Inflammation
immortality	Enabling Replicative Immortality
immune	Avoiding Immune Destruction
growth	Evading Growth Suppressors

Table 6 Abbreviations of Cellular Pathways, Georg & Lau, 2016

First, we investigated the results of all performed experiments sorted by the maximum number of simultaneously existing cells at any point of the simulation round (currentCellsMax). The results can be seen in *Figure 15*.

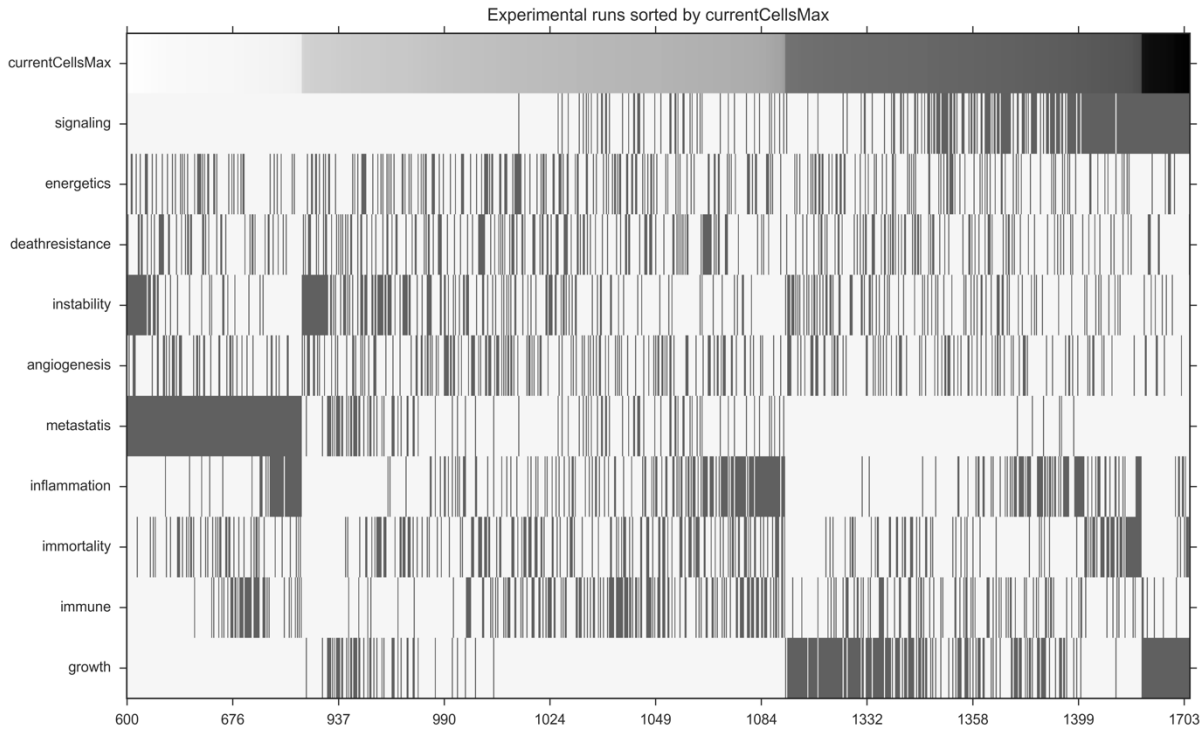


Figure 8 Experimental runs sorted by currentCellsMax, Georg & Lau, 2016

The maximum number of current cells for each possible combination, and each repeat is plotted on the x-axis. Thereby, every bar represents a single experiment, whereas the color indicates the activation status with white standing for an average activation level and gray standing for a maximum of activation, for each pathway. These pathways are plotted on the y-axis. This way, the entirety of all run experiments is sorted by the maximum number of current cells, ranging from 600 cells on the left to 1703 cells on the right side. One has to mention that the x-axis is non-linear here, as one unit equals one experiment.

From this diagram, one can already recognize, that the two pathways associated the most with a high amount of cells, are “signaling”, standing for “Sustaining Proliferative Signaling”, and “growth”, representing “Evading Growth Suppressors”.

This rather visual evaluation can be further interpreted mathematically via linear regression. The results can be extracted from *Figure 16*.

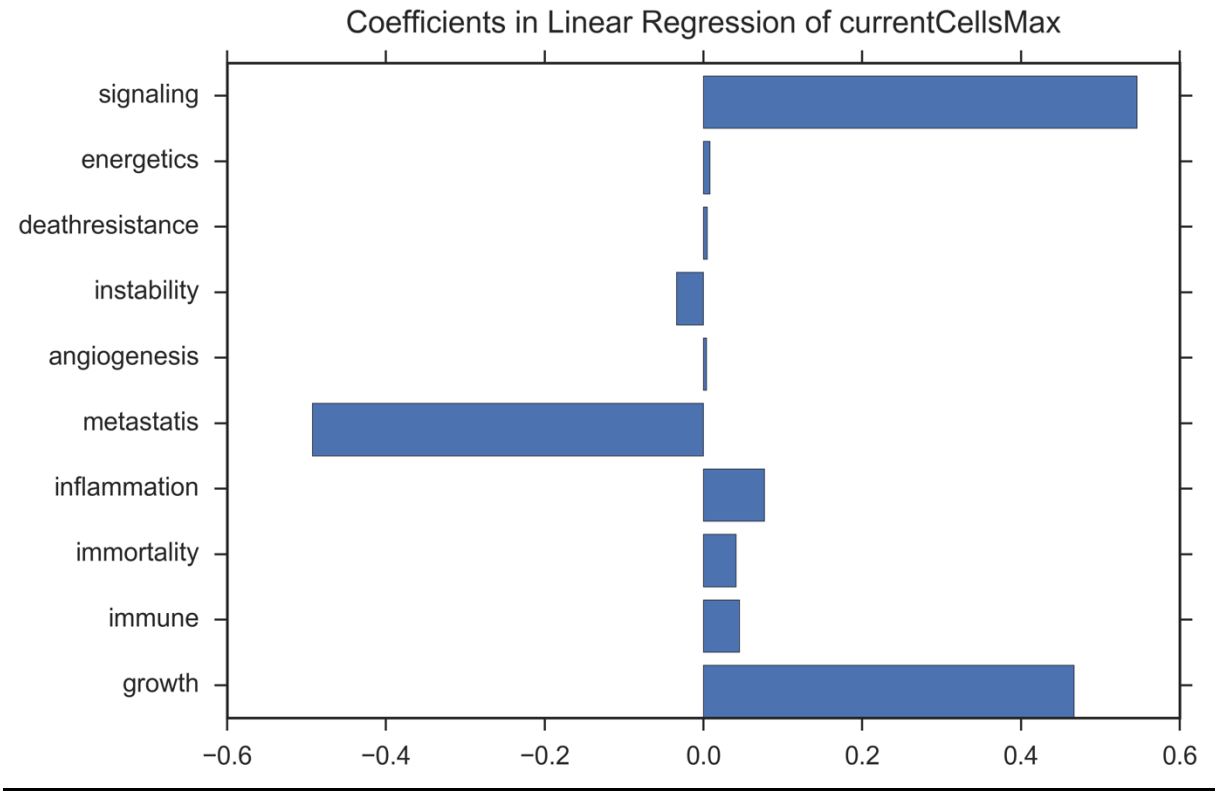


Figure 9 Coefficients in Linear Regression of currentCellsMax, Georg & Lau, 2016

The results from linear regression present the predictive coefficients for the maximum amount of current cells for each pathway. In other words, it displays the correlation between the activation, or the expression respectively, of each pathway and the number of current cells at the end of the simulation. So, a positive coefficient indicates that the activation of the related pathway increases the probability of a high number of cells in a population.

While we do not claim correct quantities, the qualitative outcome appears to be in concordance with expected results based on current clinical and wet-lab literature. According to the correlation coefficients above, “Sustaining Proliferative Signaling” and “Evading Growth Suppressors” show by far the most significant positive correlation. This finding corresponds to the assertion of Hanahan and Weinberg, who consider “Sustaining Proliferative Signaling” to be “arguably the most fundamental trait of cancer cells” (Hanahan & Weinberg 2011). The capacity of “Evading Growth Suppressors”, on the other hand, is closely linked to mutations in tumor suppressor genes. The fact that mutations in genes like TP53 and APC are highly prevalent in human cancers has been shown in many studies and indicates the importance of the

cancerous trait to evade growth suppression by alterations of the responsible genes (Vogelstein et al. 2013; Vogelstein & Kinzler 2015b).

A rather surprising and unexpected result is the large negative coefficient associated with the characteristic of “metastasis” standing for “Invasion and Metastasis”. At first sight, it appears counterintuitive that the ability to invade and disseminate, which is notoriously affiliated with high-grade malignancy and aggressive growth, has an adverse effect on the maximum number of current cancer cells and related tumor size. However, since the number of cells in our simulation corresponds to the number of cells in the primary lesion, with the total number of cancer cells in the whole organism ignored, this result appreciates in value.

Even though, metastasis might be widely associated with aggressive tumor growth, it seems plausible at the same time, that a high amount of disseminating cells, which leave the primary lesion and do not continue to proliferate in the latter, cause either a steady state of tumor growth or even a lack of proliferating cells and thereby a decrease in tumor cell number in the primary site. This hypothesis could, for example, explain the rare, but recognized, case of Cancer of Unknown Primary (CUP). A CUP is defined as metastatic cancer, without any visible primary lesion. Sometimes, only minute rests of such a primary lesion can be identified. This process might become clearer against the background of the phenomenon depicted above. It might be possible, that at some stages in tumor progression, dissemination and metastasis present a disadvantage to the primary lesion concerning cell proliferation of the latter. This assertion can be emphasized by the considerations of Vogelstein, mentioned in the paragraph “4.3.2 Cancer is an evolutionary disease”. According to him, dissemination and metastasis can occur at any time in the development of cancer, even in premalignant phases, and it is not yet understood, if additional genetic mutations are required for the potential of metastasis (Vogelstein et al. 2013).

The rest of the regression results do not show significantly positive or negative coefficients, which indicates that their particular impact on the cell population size can be neglected. Nevertheless, in certain combinations, these coefficients can show secondary importance. The importance of the interrelationship of other parameters can be extracted from an additional analysis of the data above, split into four ranges of currentCellsMax, oriented to the distribution of the number of experiments, which resulted in the different cell amounts (Fig. 17).

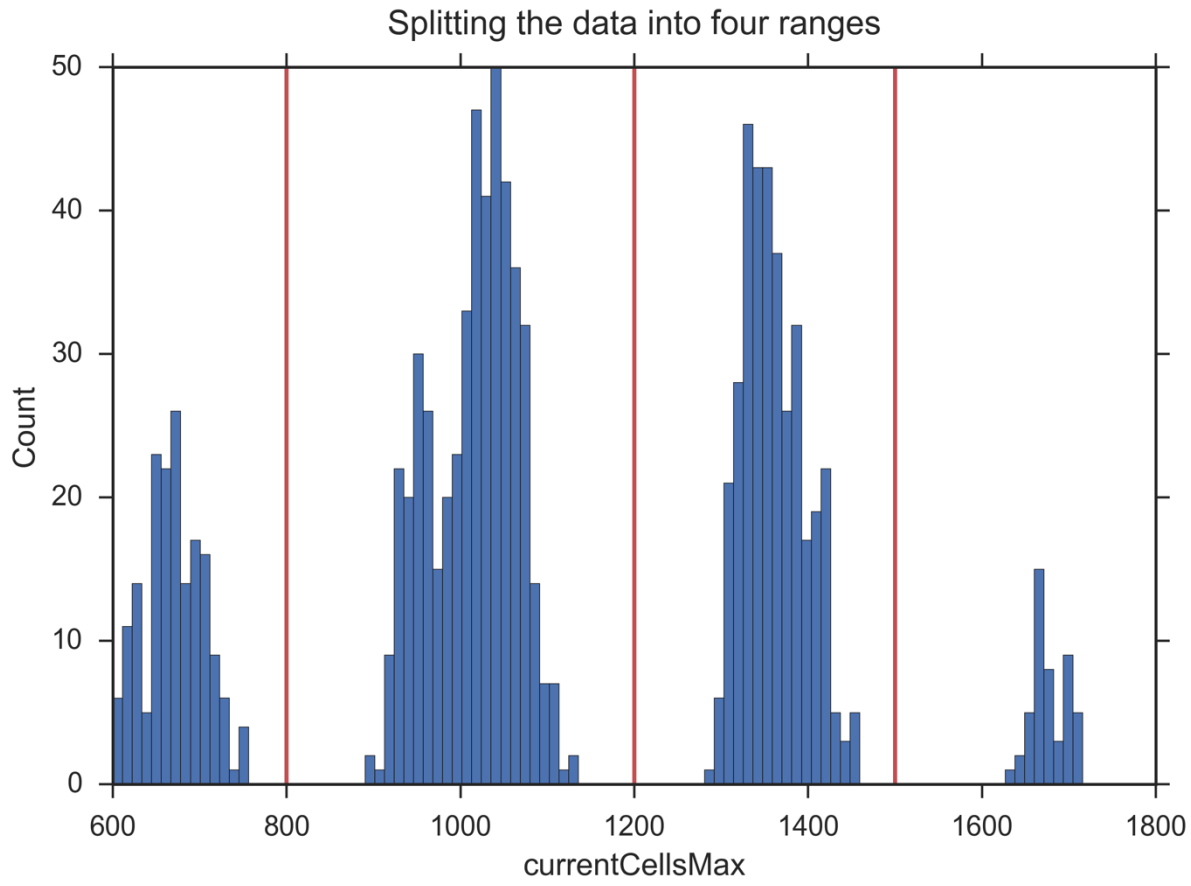


Figure 10 Splitting the data into four ranges, currentCellsMax, Geoerg & Lau, 2016

In the data depicted above, one can distinguish four ranges: ≤800 cells, 800-1200 cells, 1200 to 1500 cells and ≥1500 cells.

Via classification of the coefficients resulting from regression analysis in the different fields one can notice the impact of the three most significant pathways, both with positive and negative effect on cell population size.

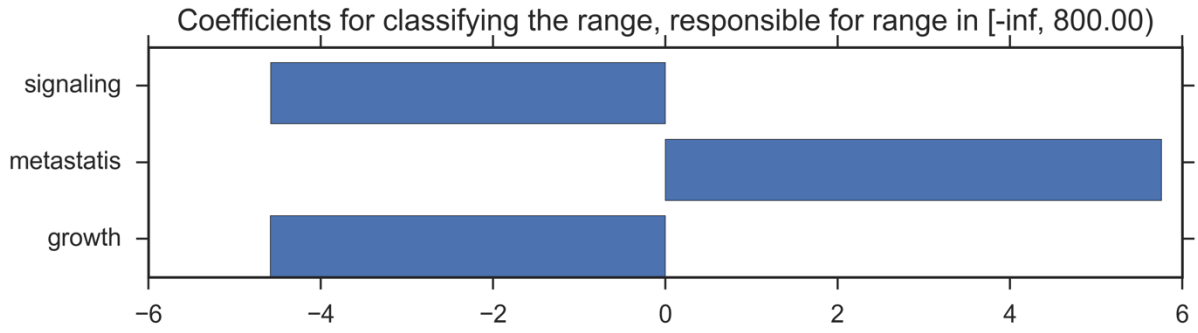


Figure 11 Coefficients for classifying the range, responsible for range in [-inf, 800.00), Georg & Lau, 2016

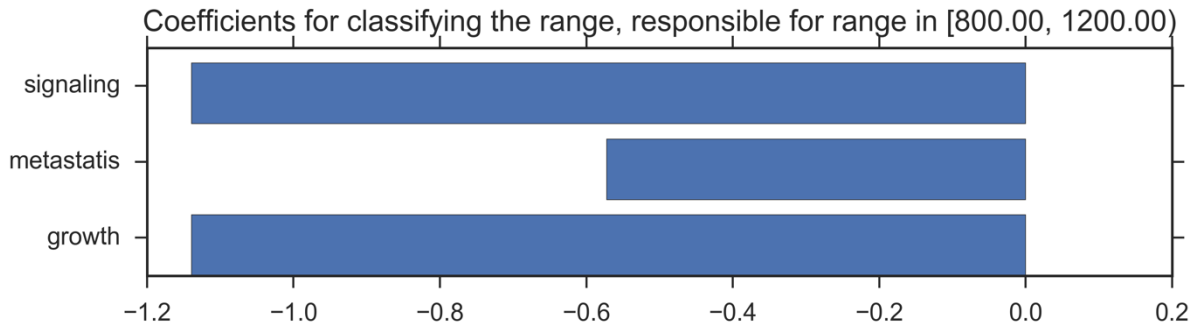


Figure 12 Coefficients for classifying the range, responsible for range in [800.00, 1200.00), Georg & Lau, 2016

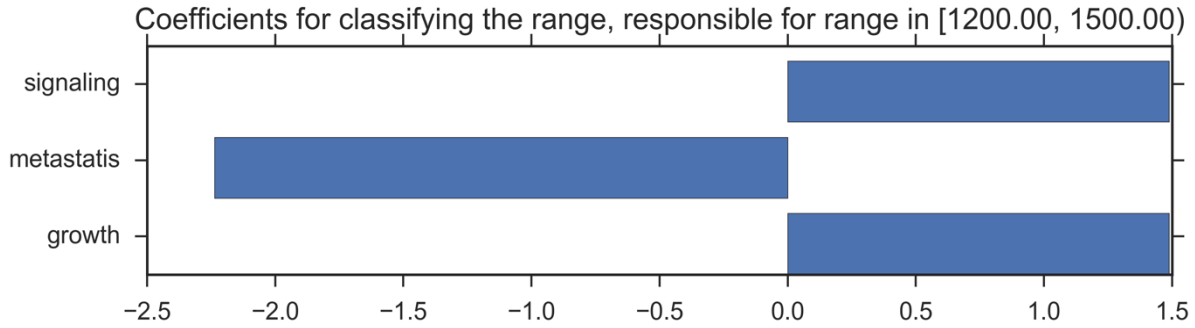


Figure 13 Coefficients for classifying the range, responsible for range in [1200.00, 1500.00), Georg & Lau, 2016

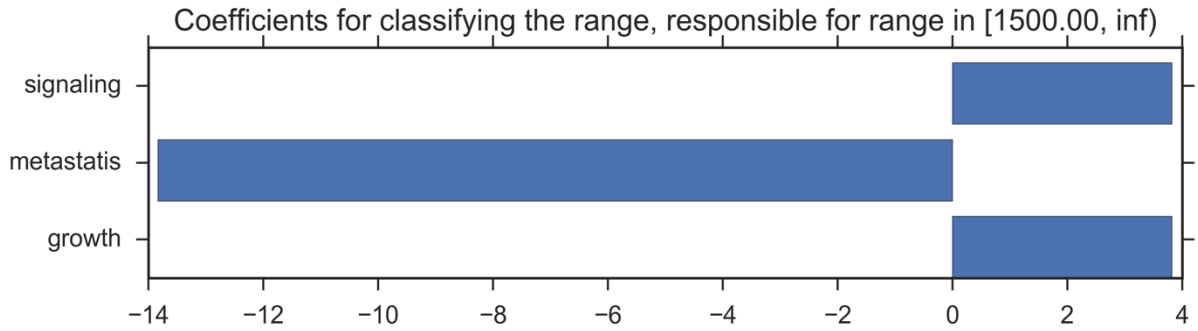


Figure 14 Coefficients for classifying the range, responsible for range in [1500.00, inf), Georg & Lau, 2016

While “signaling” and “growth” decrease the probability of small population size (≤ 800 cells), metastasis is found to have a positive impact on a small population size (Fig. 18). This distribution changes with increasing population sizes. To achieve population sizes from 800 to 1200 cells, all three pathways have to be switched off, as their coefficients are highly negative (Fig. 19). While the impact on the population size stays negative in bigger cell counts (≥1200 cells) for “metastasis”, the influence

of “signaling” and “growth” is highly positive in these population sizes, which underlines the findings elucidated above (Fig. 20).

Linear regression of the fourth range, ≥1500 cells, shows additional results concerning the importance of co-pathways in between the group of high cell counts (Fig. 21).

As *Figure 22* indicates, in between the group of high cell counts, “inflammation” and “immortality” have a positive effect on increased population sizes. This means, that in addition to the factors “signaling” and “growth”, extracted from the main analysis, “inflammation” and “immortality” raise the probability of high cell counts in the range of ≥1500 cells.

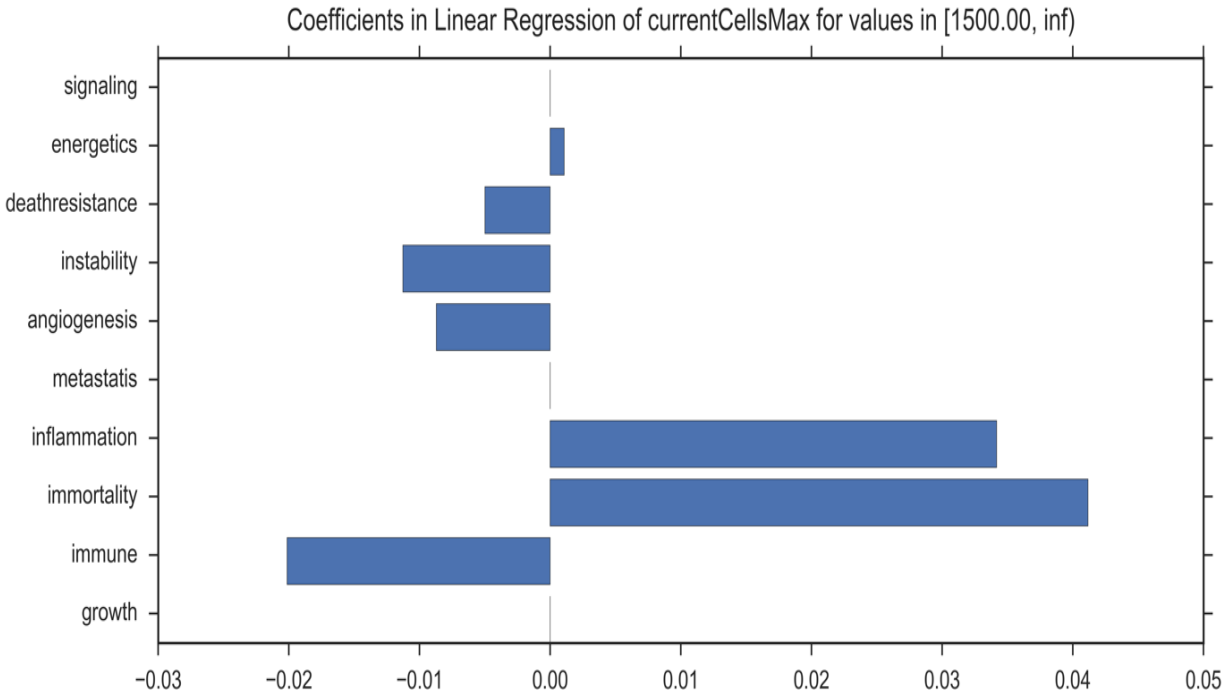


Figure 15 Coefficients in Linear Regression of currentCellsMax for values in [1500.00, inf), Georg & Lau, 2016

As a consequence, the question of intermediate steps and the formation process of the results elucidated above, need to be addressed.

Even though the current analysis does not provide direct evidence of the course of the simulation, data can be extracted indirectly by evaluating the maximum number of total cells (“allCells”) in each experiment and the number of cells killed by the immune system (“immuneAttacked”).

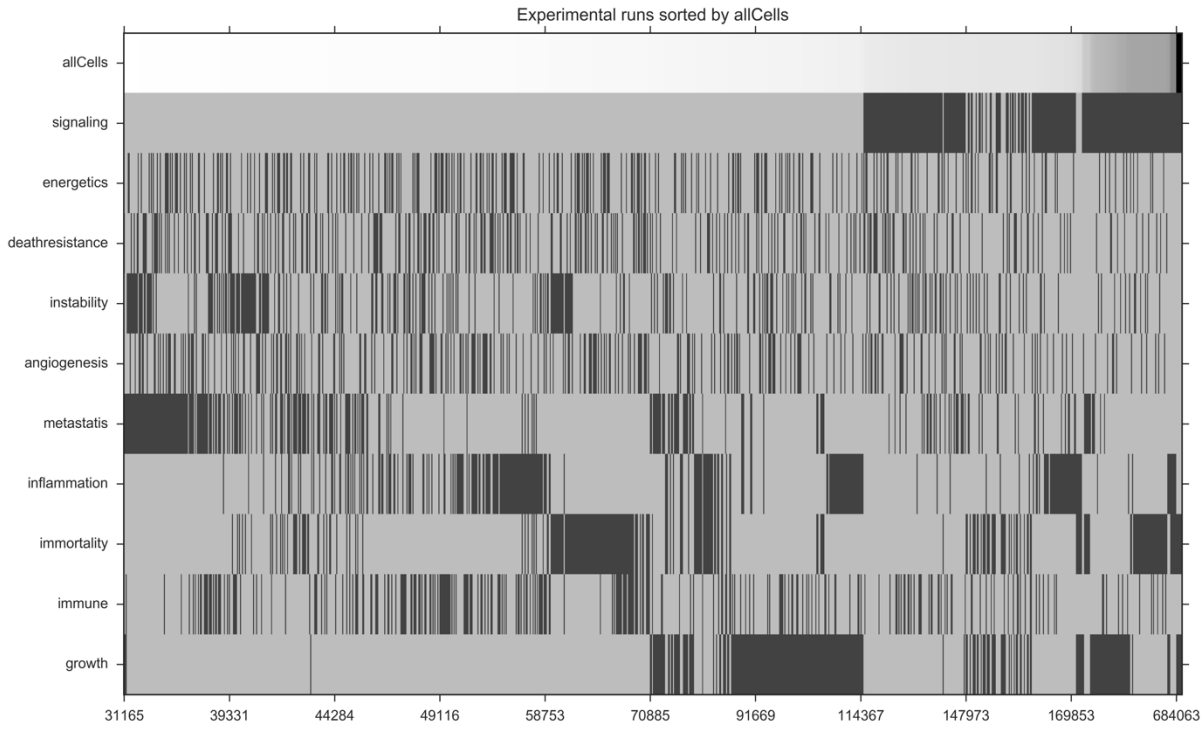


Figure 16 Experimental runs sorted by allCells, Georg & Lau, 2016

In *Figure 23*, all experimental runs are ordered due to the maximum number of all cells, which have ever evolved during one experiment.
Thereby, the x-axis is non-linear, since one unit equals one experiment. Still, it shows the number of cells ranging from 31165 cells to 684063 cells.
Comparing these figures to the results of the maximum number of current cells, it becomes apparent, that the number of ever existing cells in one experiment is about 600-fold higher than the number of cells existing at once. These findings indicate, that proportionally to the number of evolving cells, independently from the actual number, cells die at the same time, or leave the primary tumor and metastasize, respectively.

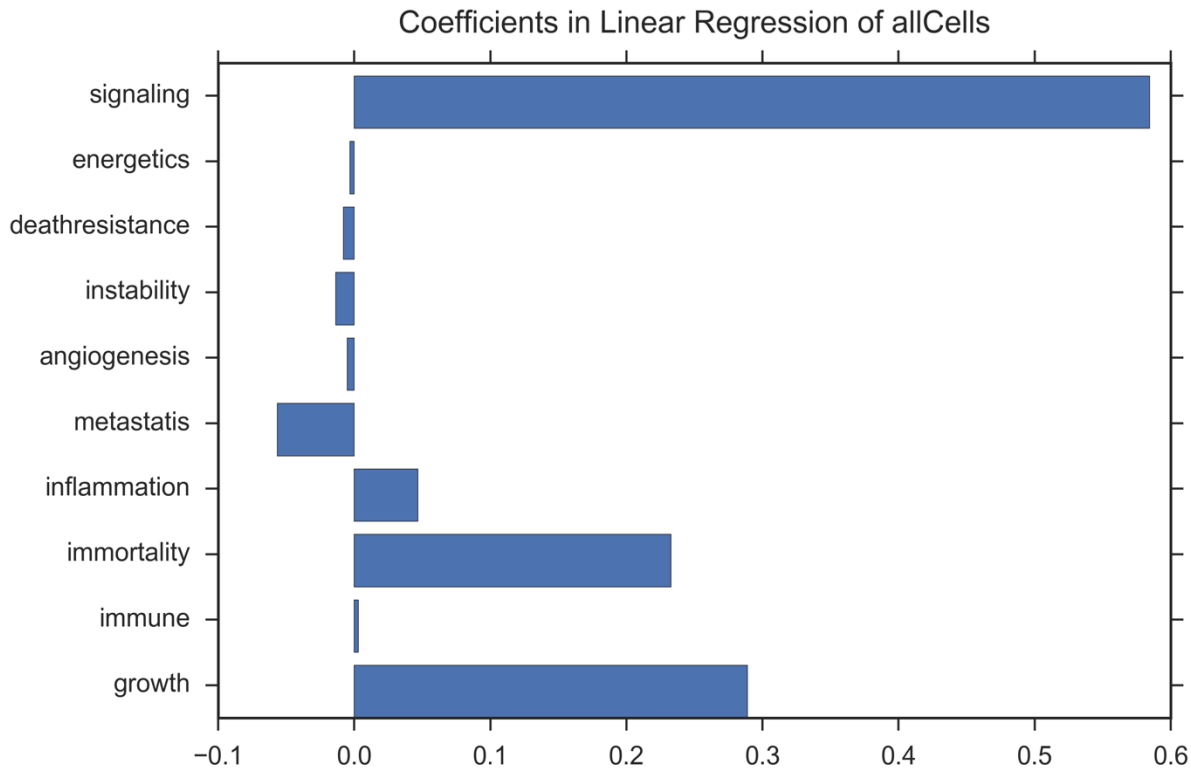


Figure 17 Coefficients in Linear Regression of allCells, Georg & Lau, 2016

Linear Regression of the experimental runs concerning “allCells” shows results similar to the results in “currentCellsMax”. However, what is remarkable here, is the fact that apart from “signaling” and “growth”, which are also highly positively correlated to a high number of current cells, “inflammation” and “immortality” show positive correlations, which corresponds to the findings after range division (Fig. 24).

One possible way of a cell to die is to be attacked by the immune system.
The number of cells, which are attacked by the immune system in each experiment is represented by the value “immuneAttacked”.

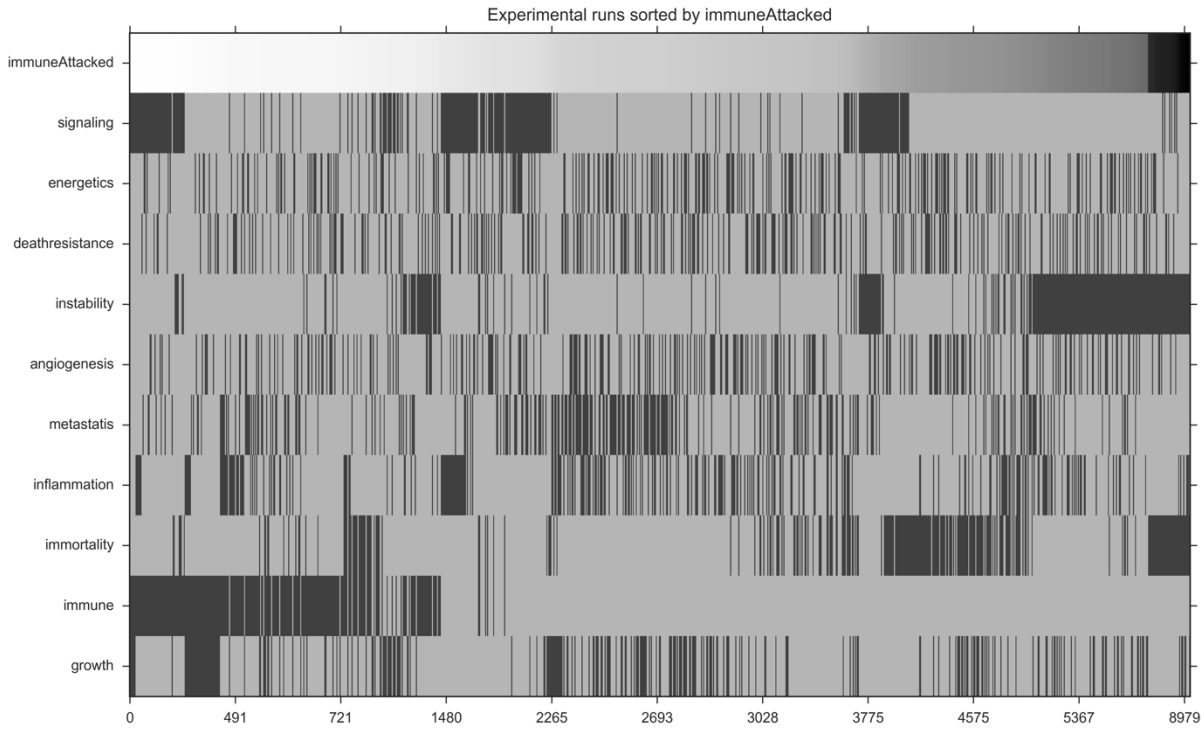


Figure 18 Experimental runs sorted by immuneAttacked, Georg & Lau, 2016

Figure 25 above shows all experimental runs sorted by the number of cells, which have been attacked by the immune system. The x-axis is non-linear here because one unit equals one experiment. Nevertheless, the x-axis shows the number of cells attacked by the immune system ranging from 0 cells to 8979 cells.

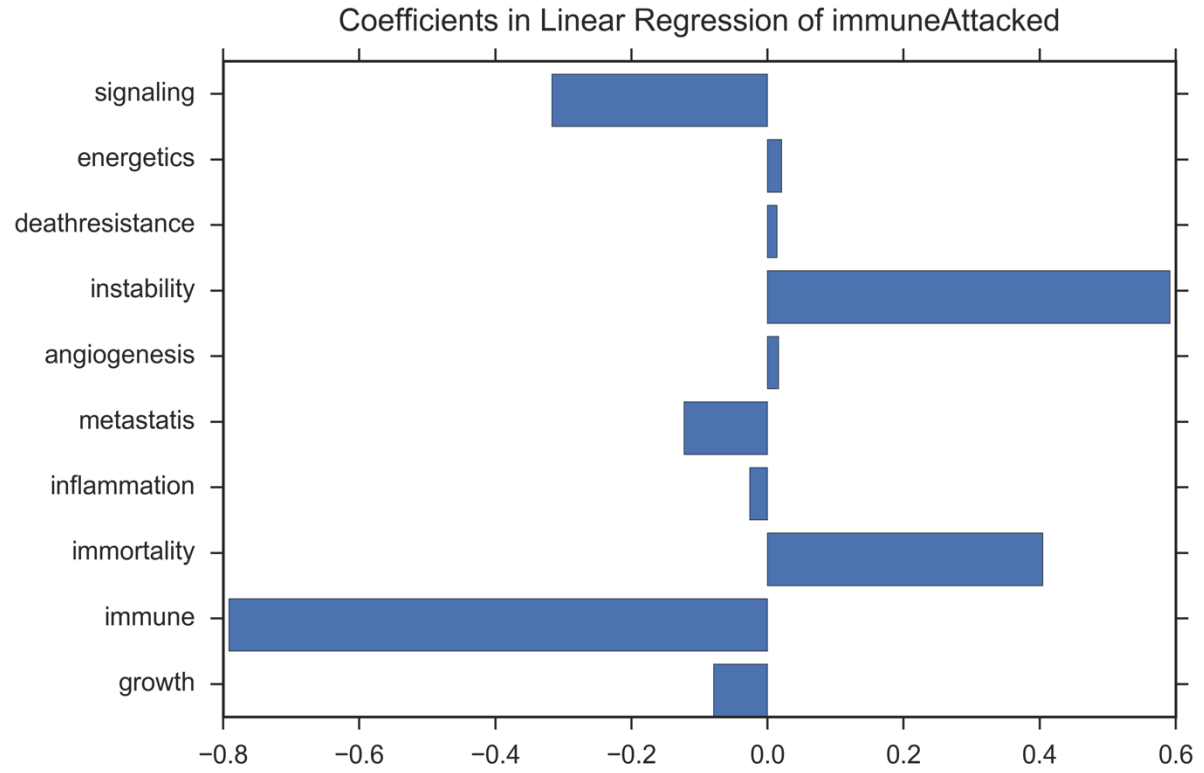


Figure 19 Coefficients in Linear Regression of immuneAttacked, Georg & Lau, 2016

Linear Regression of the results concerning the number of cells attacked by the immune system indicates, that both “instability”, standing for Genome Instability and Mutation, and “immortality”, standing for “Enabling Replicative Immortality, have a positive impact on a cell’s probability to be attacked by the immune system, while “signaling”, “metastasis” and “immune”, meaning Avoiding Immune Destruction, are negatively correlated. In light of the fact that both “Genome Instability and Mutation” and “Enabling Replicative Immortality”, despite short telomeres, cause chromosomes with high amounts of genetic failure and defects, it seems plausible, that these features increase the probability of the altered cell to be detected as foreign and eliminated in the subsequent (Fig. 26).
The protective effect of “Avoiding Immune Destruction” is evident. Moreover, the negative correlation with metastasis can be explained by the fact that cells leaving the dish are counted as dead and thereby cannot be attacked by the immune system anymore, even though they could be targeted in a real organism. The adverse effects of Sustaining Proliferative Signaling, on the other hand, remain unclear. One possible explanation might be a net effect because compared to the rising number of proliferating cells, the immune system kills relatively few cells. However, this phenomenon needs to be further examined.

5.1.3 Towards Simulating Hematopoiesis and Acute Myeloid Leukemia

The previously underestimated amount of undetermined and unexpected events occurring in the first two simulations led us to proceed to a hematopoiesis model. In hematopoiesis, normal tissue homeostasis could be modeled more easily and cells leaving the primary site (the bone marrow) were expected to produce realistic and applicable experiments and results. These results could be validated against known clinical parameters: peripheral blood counts, hematopoietic growth factor effects, and bone marrow cellularity under normal conditions and after leukemic transformation.
To increase validity and transparency, we decided to focus on three specific cell types, erythrocytes, granulocytes and thrombocytes, and their multipotent or lineage-restricted progenitor cells. We chose myelopoiesis and the related pathologic alterations because they belong to the best known and investigated processes of cell evolution and diseases resulting from dysregulation of these processes, such as aplastic anemia or acute myeloid leukemia (Doulatov et al. 2012). The great

evidence of these processes is used to program a simulation as close as possible to real biological and clinical processes, by the application of data extracted from published experiments and studies.

5.1.3.1 Establishing Physiological Hematopoiesis after Transplantation: Tissue Homeostasis

5.1.3.1.1 Biological Background

Blood cells consist of two different cell lineages, lymphoid and myeloid. While the lymphoid lineage contains T-, B- and NK-cells, acting both in the adaptive and the innate immune system, the myeloid cell branch produces granulocytes, like neutrophils, eosinophils, mast cells and basophils, monocytes, erythrocytes and megakaryocytes (Doulatov et al. 2012). As fully differentiated blood cells are mainly short-lived and undergo continuous turnover, establishment and maintenance of the blood system has to be provided by hematopoietic stem cells (HSCs). In adult mammals, small numbers of HSCs normally stay in the bone marrow and are responsible for replenishing multi-lineage progenitor and precursor cells to maintain the number of circulating blood cells (Orkin & Zon 2008). The exact number of HSCs actively participating in hematopoiesis at a given time point is not known, but it is estimated to be around 400 cells. It has been shown that about 116 stem cells are involved in the maintenance of hematopoiesis after bone marrow transplantation (Peixoto et al. 2011). Normally, stem cells divide slowly and are capable of self-renewal. Among HSCs “active” stem cells can be distinguished from “reserve compartment” cells. While “active” cells divide and contribute to hematopoiesis, the “reserve” cells remain dormant and inactive. Previous studies have shown that the stem cell division is asymmetrical, with each division resulting in one daughter cell, which stays in the HSC niche with the capability of self-renewal, and one daughter cell, which enters the pathway to differentiation (Peixoto et al. 2011).

The process of differentiation of hematopoietic cells is described as a hierarchical tree system (Fig. 27). First, HSCs give rise to multipotent progenitor cells of the two main lineages, myeloid (CMP) and lymphoid (MLP/CLP). These cells then divide into precursor cells devoted to one single or multiple determined pathways to then fully differentiated mature cells (Manesso et al. 2013). In the myeloid branch, CMPs produce GMPs, which result in granulocytes or monocytes, and MEPs, which give

rise to erythroid and megakaryocyte cells. In the lymphoid lineage, CLPs produce B-cell precursor cells and earliest thymic progenitors (ETPs), which are destined to develop into T- and NK-cells (Doulatov et al. 2012).

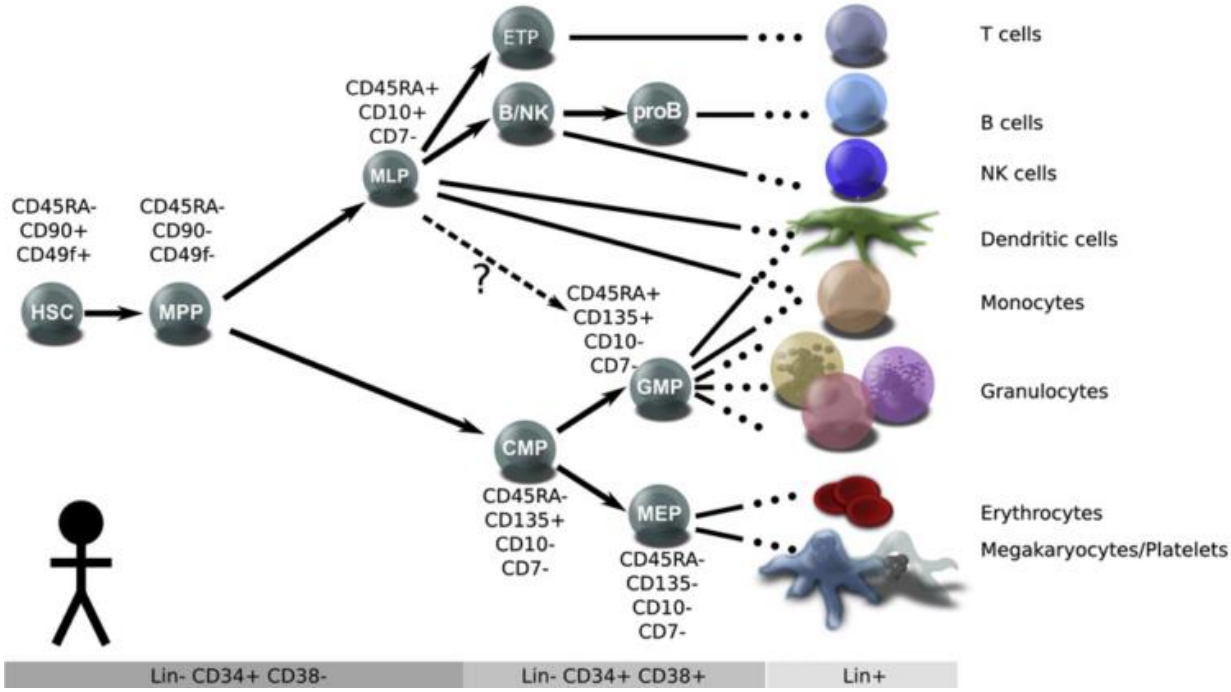


Figure 20 Hierarchical tree system of hematopoiesis, Hematopoiesis: A Human Perspective, Doulatov et al., 2012

The whole path of differentiation is assumed to be a unidirectional lineage specification process resulting from a series of irreversible decisions towards increasingly differentiated states with on the other hand decreasing potential for self-renewal and multipotentiality (Manesso et al. 2013).

The molecular mechanisms regulating this differentiation hierarchy are not yet completely understood. Many findings indicate that it is based on a plethora of direct and indirect interactions between cytokines, growth factors, transcription factors and feedback mechanisms as well as epigenetic mechanisms. The microenvironment of the HSC, the HSC niche is assumed to play a pivotal role in the specification process (Orkin & Zon 2008). To provide an overview, some of the most important transcription factors involved in hematopoietic differentiation are depicted in *Figure 28*.

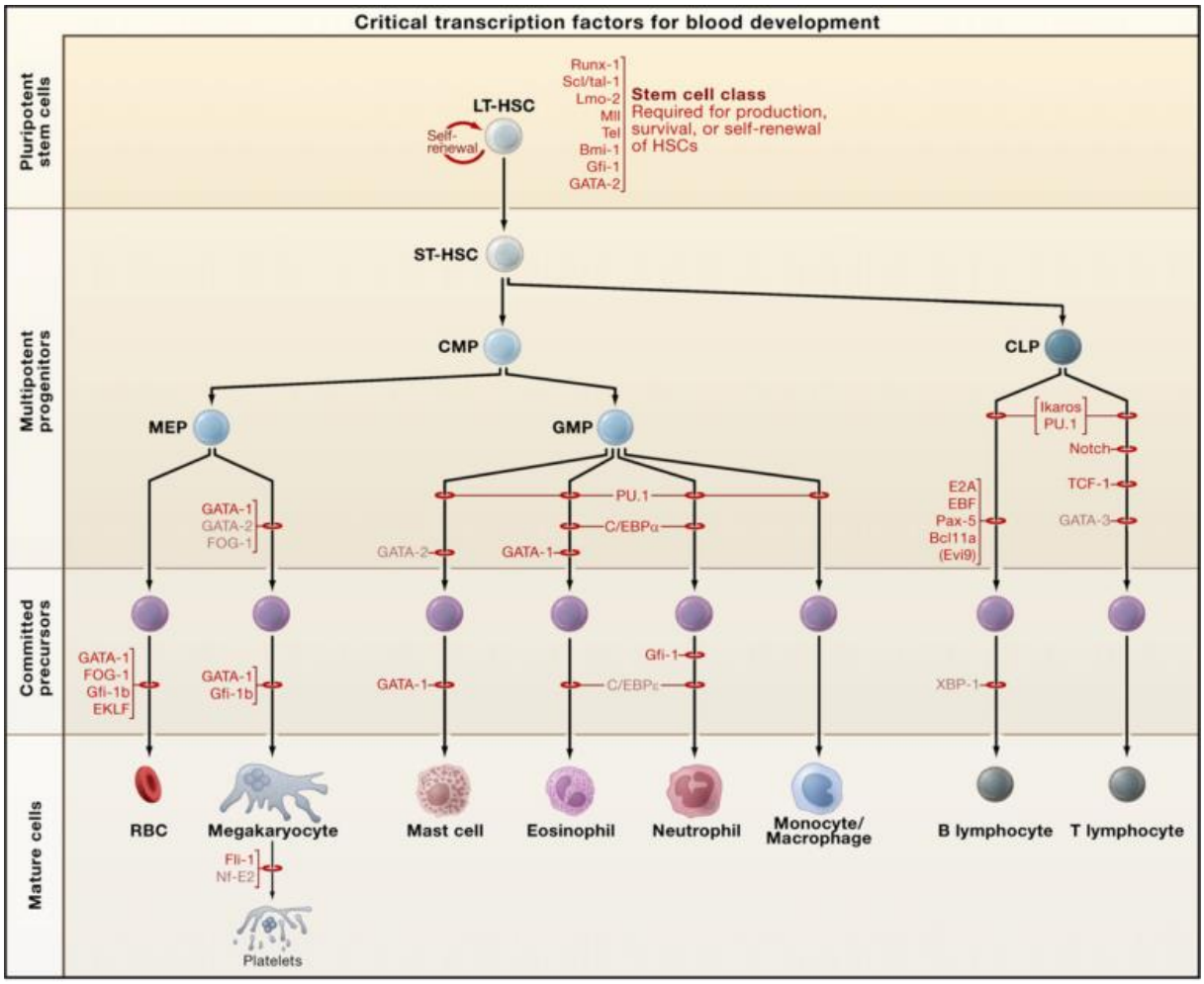


Figure 21 Critical transcription factors for blood development, Hematopoiesis: An Evolving Paradigm for Stem Cell Biology, Orkin & Zon, 2008

5.1.3.1.2 Logical Background and Programming

The first prototypes of the simulation were programmed in pure Python because of its ability to easily change and adapt the code. While the first complete and interactive version was written in Java, an improved simulation in Python, embeddable into a web-service with Bokeh, was written to combine interactivity and the possibility of fast repetitive simulations with different settings.

As mentioned above, in this model myelopoiesis in the bone marrow is simulated as the establishment of hematopoietic tissue after transplantation, as well as its impact on the peripheral blood composition. Each cell in the bone marrow is simulated individually with a lot of biological aspects, such as differentiation characteristics or the potential to undergo different genomic mutations or even apoptosis.

According to data extracted from the literature, the following cell types and their progenitor cells are considered (Fig. 29):

1) Dormant Stem Cells (cp_1)
    a. Capable of self-renewal
    b. Low division rate (enabled to divide every five ticks, depending on need)
2) Long Time Active Stem Cells (cp_2)
    a. Capable of self-renewal
3) Short Time Active Stem Cells (cp_3)
    a. Division in two higher differentiated daughter cells
4) Multipotent Progenitor Cells (cp_4)
5) Common Myeloid Progenitor Cells (cp_5)
6) Progenitor Cells (5)
    a. Myoblasts (gran_1 – gran_5)
    b. Megakaryocytes (throm_1 – throm_5)
    c. Erythrocyte Progenitor Cells (ery_1 – ery_5)
7) Differentiated Cell Types
    a. Granulocytes (32 for each gran_1 progenitor cell)
    b. Thrombocytes (300 for each throm_1 progenitor cell)
    c. Erythrocytes (32 for each ery_1 progenitor cell)

For transparency purposes, the differentiation process of each cell line has been reduced to five steps. The number of cell divisions is determined with the aid of an estimation, as data about the exact biological division rates are controversial.

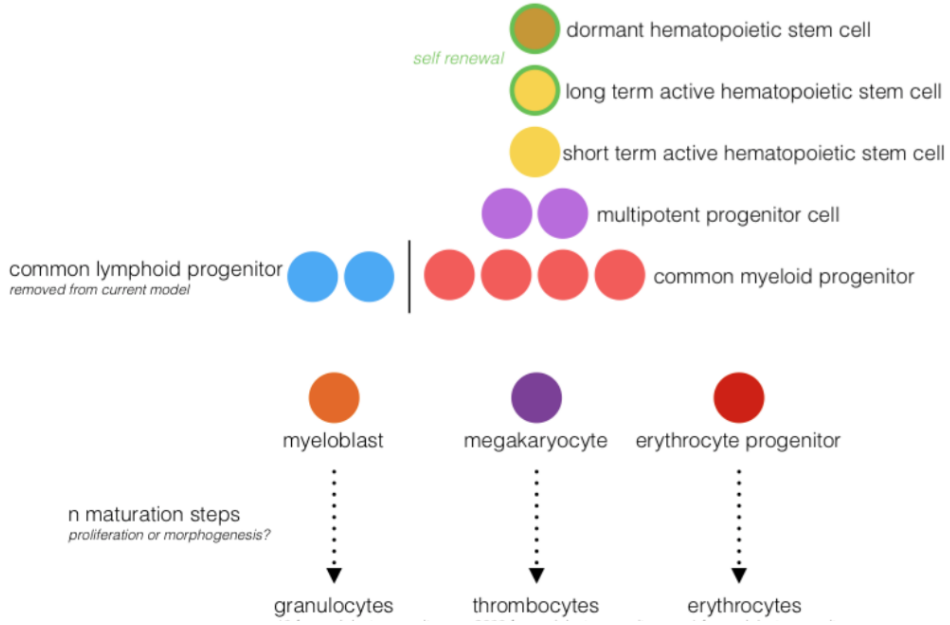


Figure 22 Cell types considered in the Simulation of Hematopoiesis, Georg, 2016

5.1.3.1.3 The Simulation Process

At the beginning of the simulation, one dormant stem cell divides and the division cascade of myelopoiesis proceeds depending on the apparent concentration of transmitters. The term “transmitter” is used here as the integral of stimulating and inhibitory molecules and mechanisms. The most important stimulators of the three pathways of myeloid blood cell production studied here are the three most influencing growth factors, erythropoietin, G-CSF and thrombopoietin.

In the current model, these transmitters are virtual countable values, which are determined by experimental trial to reach a steady state of normal hematopoiesis. Transmitters are eliminated for each cell, which leaves the bone marrow and enters the peripheral blood as a fully differentiated blood cell, whereby one transmitter is eliminated for one cell. One new transmitter is generated in the bone marrow for each cell, which dies in the peripheral blood.

In the simulation, the time unit is determined as “tick”, with 1 tick = 1 update loop. In our simulation 1 tick is defined as 1/10 day so that 10 ticks = 1 day.

In each tick, the following assessment proceededs chronologically.

1) In the first step, it is tested, which cells of the peripheral blood die and how many new transmitters are generated as a result.
2) All transmitters available in the bone marrow bind to progenitor cells, which lie on the pathway to the final cell of this transmitter, whereby the probability of binding increases with the grade of differentiation. More precisely, the binding likelihood depends on the binding capacity (= space) of a cell, with a probability distribution inverse to the transmitters required for cell division. The higher the level of differentiation, the less binding space, and the higher the binding probability. The process of transmitter binding can take more than one tick. Depending on the number of bound transmitters, cell division is initiated. Thereby, the number of transmitters needed to induce cell division equals the number of possible fully differentiated cells evolving from the current progenitor cell. As a result, a more differentiated cell requires fewer transmitters to divide, than a less differentiated cell. All cells at the same level of differentiation are considered equal in the simulation. They are not depicted as single variables but as a group of individual items. In other words, cells at the same level of differentiation bind transmitters randomly, and, as a result, divide randomly. Nevertheless, in one tick, single cells can be divided, as well

as several cells from the same level, as well as several cells from distinct differentiation levels.

If the amount of a single transmitter type does not reach the threshold of a certain cell for a division, all currently attached transmitters are summarized. The apparent dividable cell is then tested with the sum of all transmitters. If the sum is high enough to induce cell division, the differentiation type of the evolving cells depends on the ratio of transmitters. The transmitter with the largest percentage determines the resulting cell type.

3) For each fully differentiated cell, which is transferred to the peripheral blood, a transmitter of this type is eliminated. If a cell division does not result in a fully differentiated cell, the transmitters attached to this cell are released into the bone marrow and can bind to another cell.
4) Every cell, which does not divide, generally keeps the transmitters attached independent from the update loop. Only with a probability of 5% per tick, it can lose all attached transmitters at once.

Storage of cell information

Every item evolving in the simulation, as well as every event, is saved in tables and objects. For the particular storage tables, see below. The tables of all progenitor cells and the number of transmitters in the bone marrow change simultaneously so that only the current state can be examined.

A statistics module attached to the simulation collects all relevant information for analysis after a complete run.

Progenitor Cells in the Bone Marrow

All progenitor cells existent in the bone marrow are registered as object references in an array.

Each cell contains its own information about cell type, length of telomeres, telomere loss at cell division (normally 1), number of attached transmitters, amount of free transmitter space, next differentiation step (with number and type of required transmitters), Boolean state of dependence on transmitters and a Boolean state of being a cancer cell.

Transmitters in the Bone Marrow

Transmitters are registered in two dictionaries (Tab. 9). First, the entirety of all transmitters (attached and free) and second, the number of free transmitters.

Number of Transmitters per Cell Type	
Trans_ery	x
trans_gran	y
Trans_throm	z

Table 7 Exemplary dictionary of the number of transmitters per cell type, Worm, 2016

Peripheral Blood Cells

The number of cells in the peripheral blood is described via the storage of their determined point of death in a dictionary. When a new cell evolves, the probable lifetime is calculated by the average extracted from literature data and the Gaussian Distribution. Depending on the result, a cell is defined as a time of death in ticks and is added to *Table 10*, representing the number of dying cells per tick. For the depiction of all current

cells at tick x, the entirety of cells is summarized, from tick x to tick n.

Cell/Tick	1	2	…	n
Erythrocytes	20	15	…	x
Granulocytes	45	30	…	y
Thrombocytes	30	55	…	z

Table 8 Exemplary dictionary of determined points of death of peripheral blood cells, Worm, 2016

Graphic Depiction

Figure 30 shows the visible surface of the simulation. There are nine different figures. The figures in the first row represent the state and differentiation process of erythrocytes. The second row shows the behavior of the granulocytes and the third row the development of the thrombocytes. In the first column, the number of fully differentiated cells in the peripheral blood over time is depicted. In the second column, the corresponding number of total (red) and free (orange) transmitters in the bone marrow is shown over time. These graphs are built with the precision of 1 dot

per tick. In the last column, a bar chart demonstrates the number of progenitor cells at the distinct differentiation levels at the current state, with higher ciphers standing for higher differentiation. The fourth row shows three additional charts. On the left side, there is a description of the number of needed time in milliseconds for each simulation step. In the middle, the total number of cells in the bone marrow is represented (red) compared to the number of cells able to divide (orange). On the right sight, a pie chart gives an overview of the relative distribution of different progenitor cells in the bone marrow.

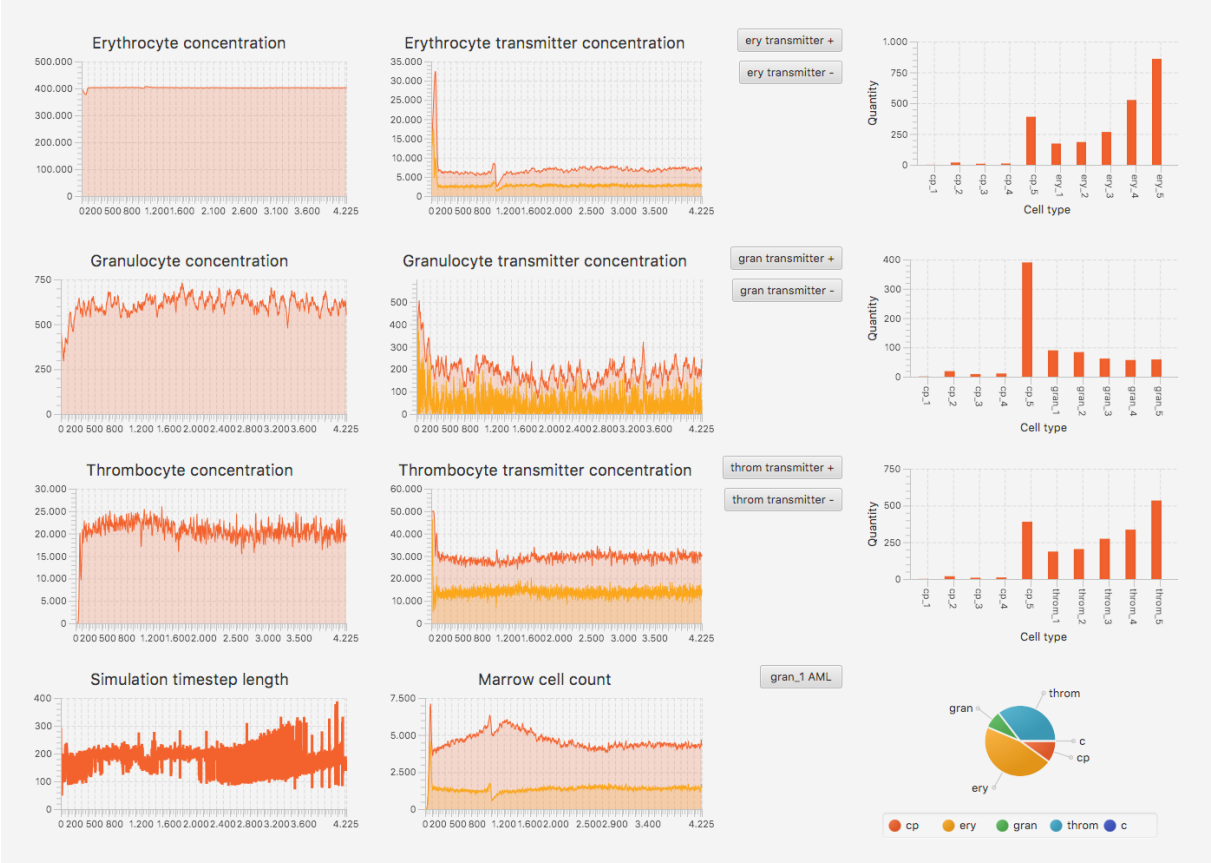


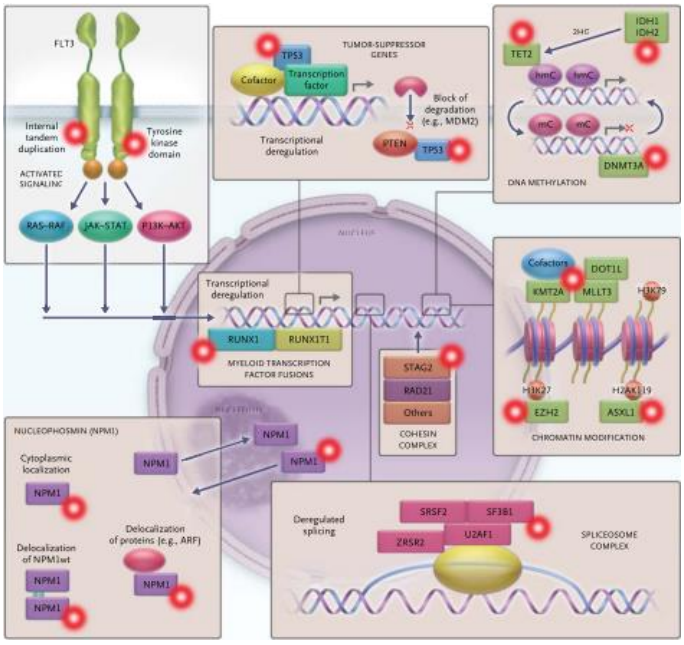
Figure 23 Surface of the Simulation of Hematopoiesis, Worm, 2016. The full simulation is provided via hem-model.psiori.com/hema_simulation.

5.1.3.2 Simulating Leukemogenesis: Acute Myeloid Leukemia

5.1.3.2.1 Biological Background

Acute Myeloid Leukemia (AML) can be described as the abnormal proliferation and poor differentiation of a clonal population of myeloid stem cells with typical characteristics, such as clonal expansion and infiltration of the bone marrow, blood and other tissues with subsequent hematopoietic impairment and bone marrow deficiencies (De Kouchkovsky & Abdul-Hay 2016; Dohner 2015; Papaemmanuil et al. 2016). It is the most common acute leukemia in adults, forming approximately 80% of

cases in adult patients. The incidence rises with age, ranging from about 1.3 per 100 000 population in patients at 65 years and younger to 12.2 per 100 000 population in patients older than 65 years (De Kouchkovsky & Abdul-Hay 2016). Even though acute myeloid malignancies can be ordered into favorable, intermediate and adverse-risk groups, prognosis widely varies, and mortality is still high (De Kouchkovsky & Abdul-Hay 2016). AML is considered to be a biologically and clinically heterogeneous disease since it can evolve by a previous hematological disorder, after a prior therapy or, most frequently, as a de novo malignancy (De Kouchkovsky & Abdul-Hay 2016). It can be classified into AML with recurrent genetic abnormalities, AML with myelodysplasia-related changes, therapy-related AML and AML not otherwise specified, due to the World Health Organization (WHO) Classification of Tumors of Hematopoietic and Lymphoid Tissues (Dohner 2015). Recent efforts to define adequate classifications try to exploit the increasingly apparent molecular heterogeneity of AML. AML is assumed to develop over time, with an increasing number of somatically acquired driver mutations, resulting in multiple competing clones. Recent findings indicate, that a plethora of 5234 driver mutations in 76 genes, evolving in combinations of two or more in 86 % of tested patients, can be identified in association with AML (Papaemmanuil et al. 2016). An approach to cluster the most frequent genetic mutations was performed by Döhner in 2015. In *Figure 31*, eight different groups of mutations can be distinguished resulting in the following molecular pathways.



1. Proliferative Advantage (*upper left*)
2. Transcriptional Deregulation and impaired hematopoietic differentiation (*center left*)
3. Aberrant localization of NPM1 and NPM1-interacting proteins (*lower left*)
4. Deregulated RNA processing (*lower right*)
5. Impaired chromosome segregation and transcriptional regulation (*center middle*)
6. Deregulation of chromatin modification (*center right*)
7. Deregulation of DNA methylation (*upper right*)
8. Transcriptional Deregulation and Impaired Degradation (*upper middle*)

According to our revised hallma

Figure 31 Cluster of the most frequent mutations in AML, Acute Myeloid Leukemia, Döhner, 2015

rk concept depicted above, one can assign each molecular pathway to one of the three major hallmark pathways, Growth/Apoptosis Balance (1), Genetic Fidelity/Immortality (2-8) and Differentiation Block/Stem Cell Features (2) (Dohner 2015).

5.1.3.2.2 The Simulation Process

To simulate the development of AML, we programmed a new cell type, a cancer cell, for each differentiation level. They contain an arbitrary combination of up to three different alterations, independence from the Hayflick-limit (telomere-shortening = 0), independence from transmitters (transmitter dependence = false) and a block of differentiation (as the next differentiation step the same cell type is determined). Also, for statistical reasons, the cancer state is turned ON.

5.1.3.2.3 Exemplary Experiments – Possible Implications

Similar to the validation process of the simulation of carcinogenesis, exemplary experiments have been simulated in this hematopoietic model.

Aiming to investigate the impact of each of the three proposed pathways of AML, both as single parameters as well as in the combination of two or three, leukemia has been simulated in several experimental runs for 10 000 ticks each, which equals 100 days in real time. The three different pathways have been activated in various sequences, at tick 2000, 5000 and 8000. Every setting has been repeated for five times. The visual part has been abandoned for these experiments to allow a faster simulation. In the beginning, 20 runs without any pathway activated have been documented as a control group. The following graphs are exemplary for all experimental runs, as the results were homogenous. For each set of experimental runs, four graphs have been plotted for the granulocyte concentration in the peripheral blood over time (Fig. 32), the thrombocyte concentration in the peripheral blood over time (Fig. 33), the erythrocyte concentration in the peripheral blood over time (Fig. 34) and the total number of cells in the bone marrow over time (Fig.35). Depicted in the graph are the healthy range in blue, the healthy average in orange and the mutated average in red.

In the first set of experiments, telomerase activation, which allows cell division

independent from the Hayflick-limit, is applied to one cp_5 progenitor cell at tick 2000. The term cancer is used in the legends to indicate malignant alteration.

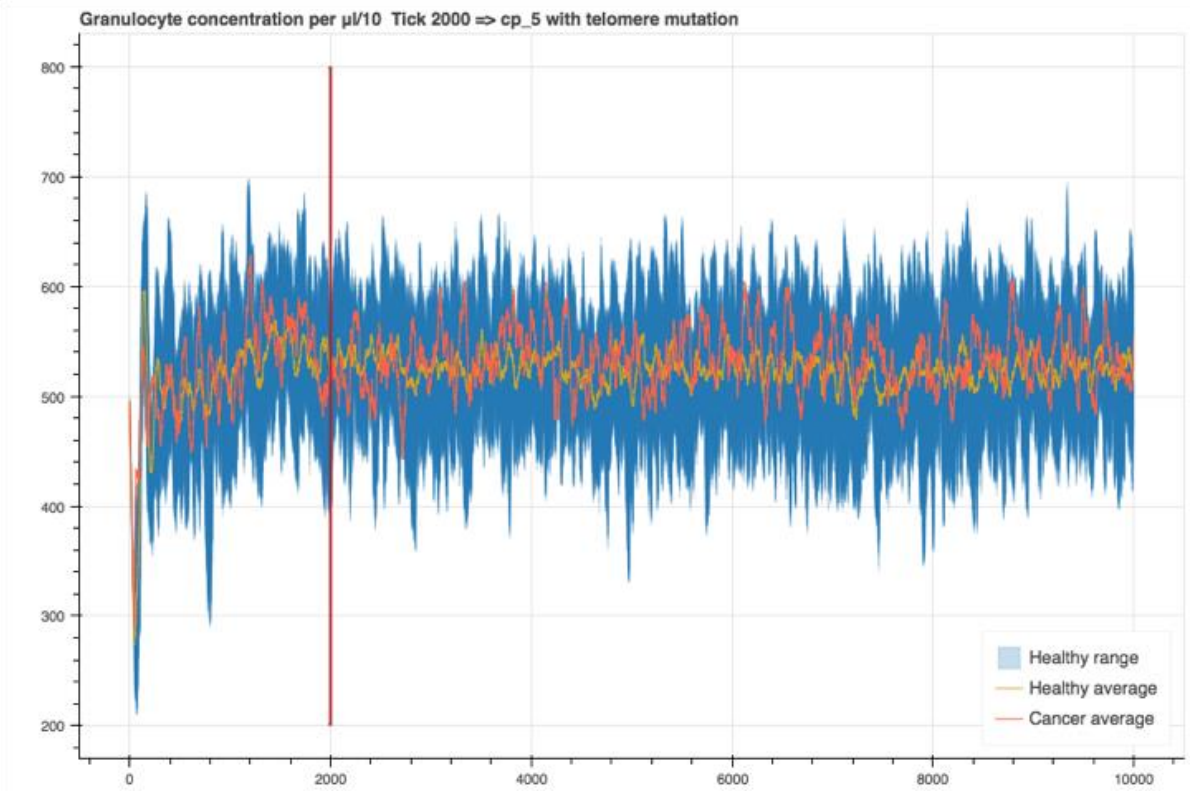


Figure 24 Granulocyte concentration per µl/10, Tick 2000: cp_5 cell with telomerase activation, Worm, 2016

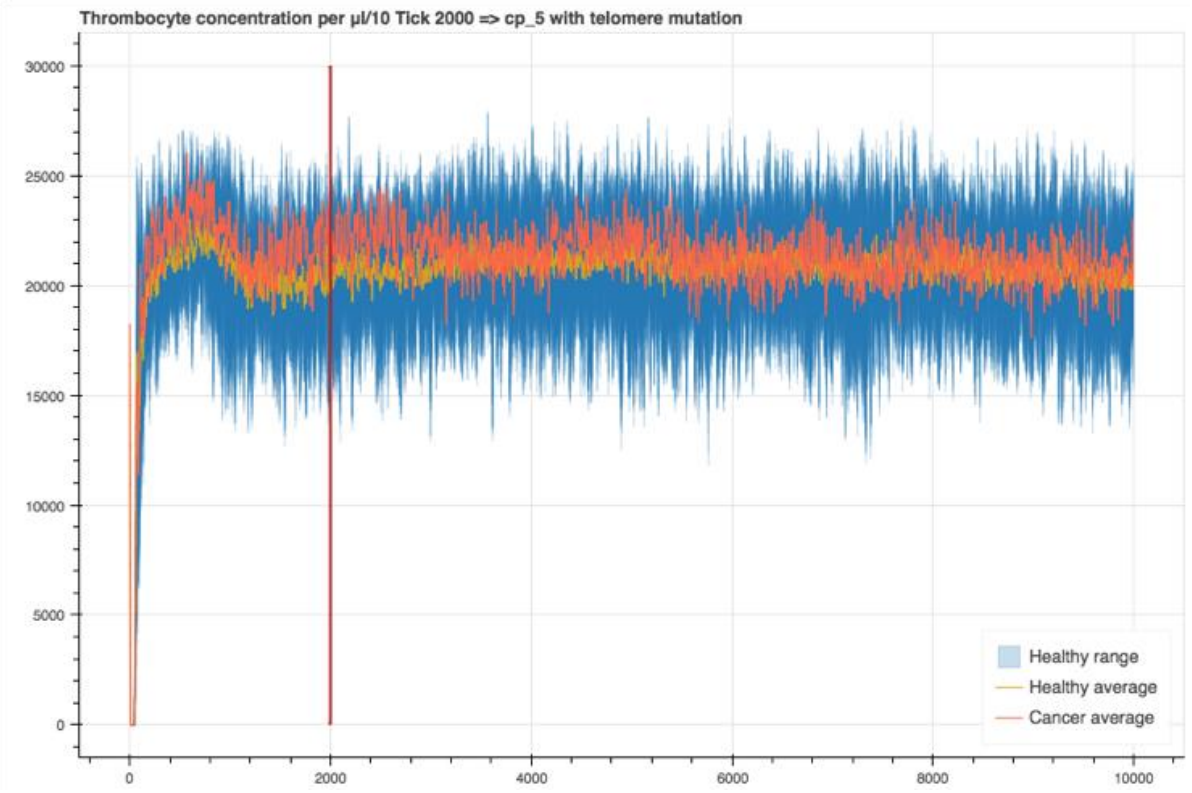


Figure 25 Thrombocyte concentration per µl/10, Tick 2000: cp_5 cell with telomerase activation, Worm, 2016

Figure 26 Erythrocyte concentration per µl/10, Tick 2000: cp_5 cell with telomerase activation, Worm, 2016

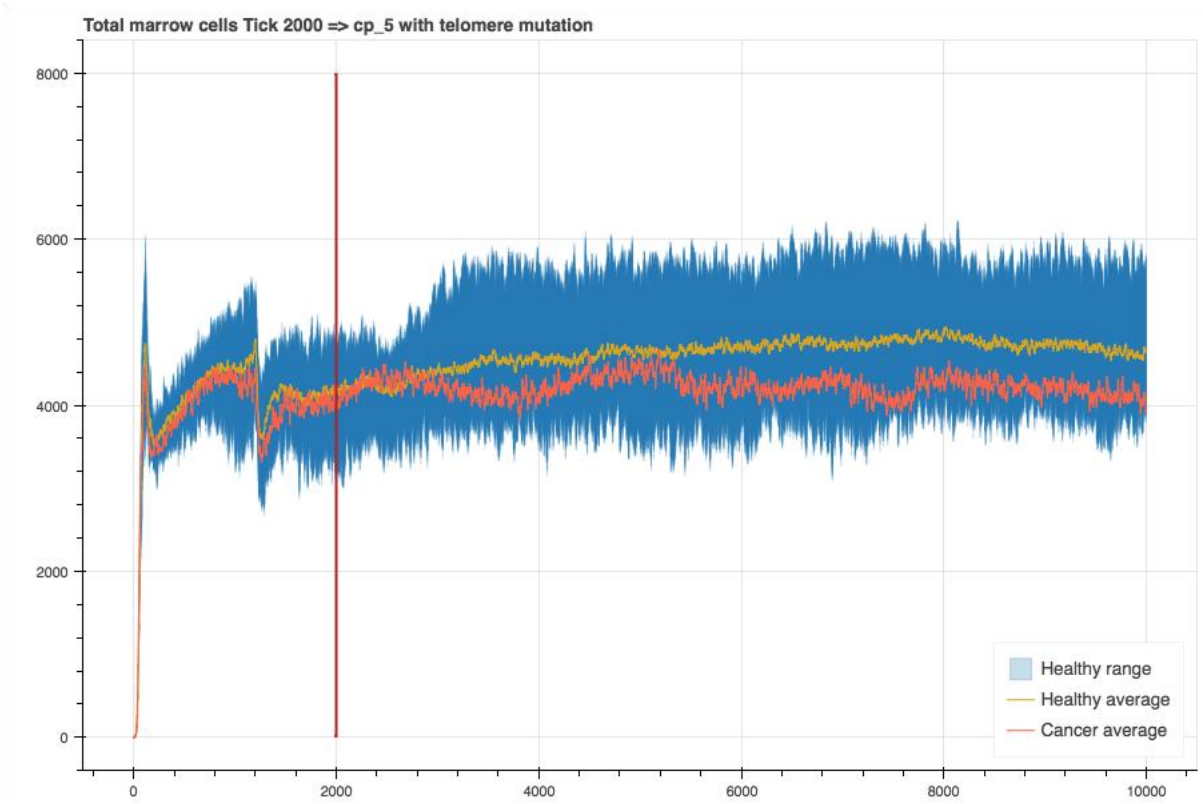


Figure 27 Total marrow cells, Tick 2000: cp_5 cell with telomerase activation, Worm, 2016

Except for a slightly decreasing number of total marrow cells from tick 2000 onwards, there is no significant effect visible in the graphs above.

In the second experimental run, a cp_5 cell with telomerase activation was added at tick 2000 and a cp_5 cell with both telomerase activation and a block of differentiation at tick 5000.

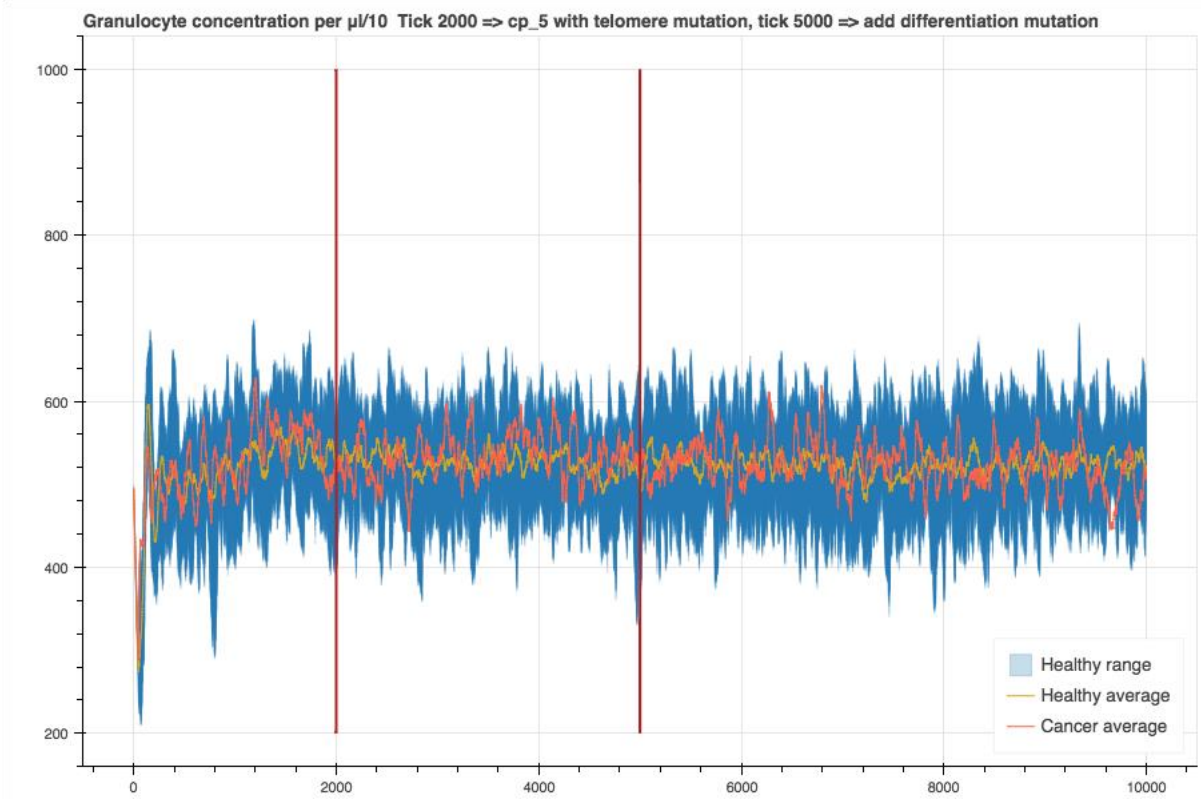


Figure 28 Granulocyte concentration per µl/10, Tick 2000: cp_5 cell with telomerase activation, Tick 5000: differentiation block, Worm, 2016

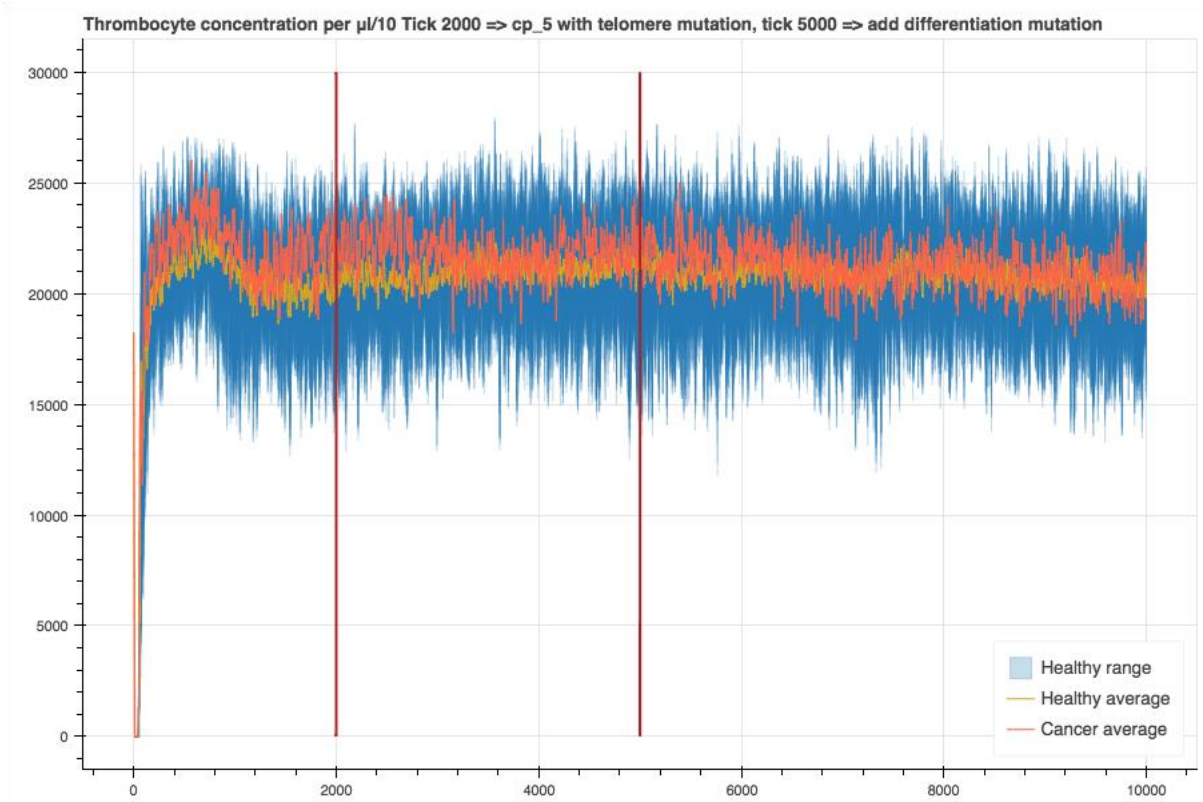


Figure 29 Thrombocyte concentration per µl/10, Tick 2000: cp_5 cell with telomerase activation, Tick 5000: differentiation block, Worm, 2016

Figure 30 Erythrocyte concentration per µl/10, Tick 2000: cp_5 cell with telomerase activation, Tick 5000: differentiation block, Worm, 2016

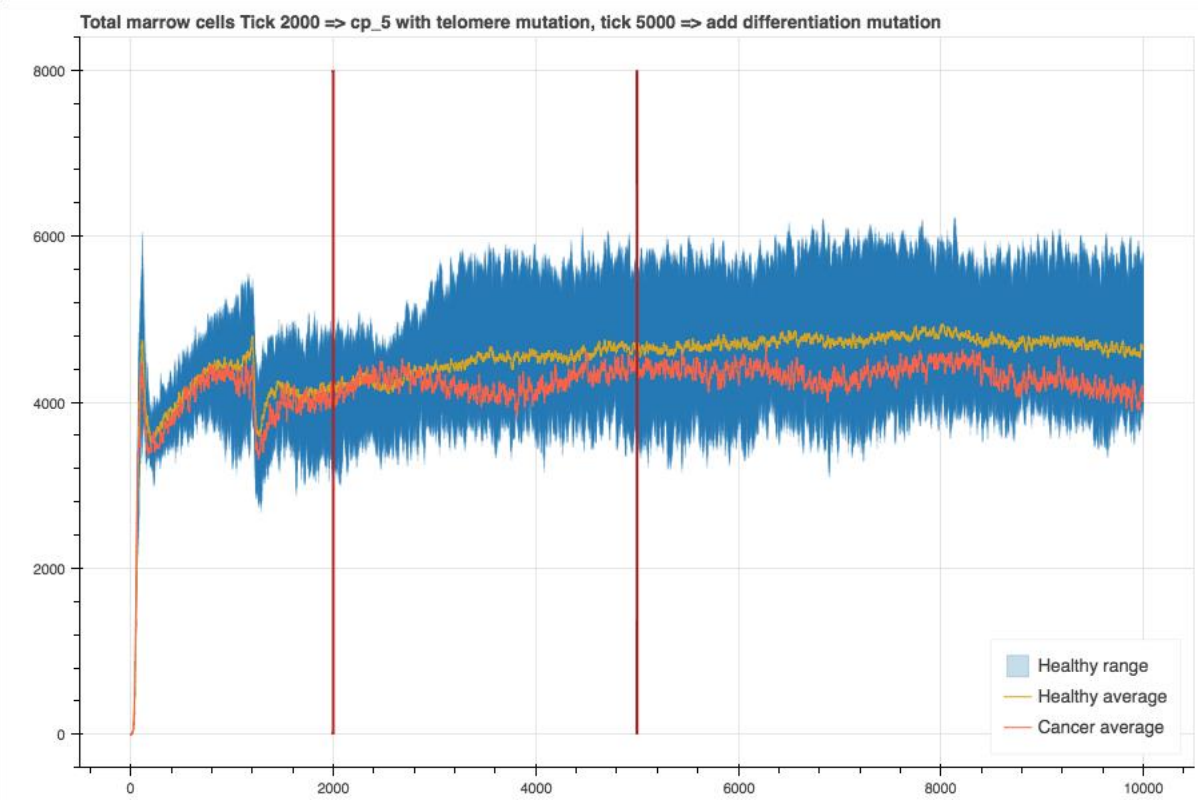


Figure 31 Total marrow cells, Tick 2000: cp_5 cell with telomerase activation, Tick 5000: differentiation block, Worm, 2016

Still, there is no obvious effect visible (Fig. 36 – 38), except for a lower cell count in the bone marrow from tick 2000 onwards (Fig.39).

In the last set of experimental runs, a cp_5 cell with a telomerase activation was added at tick 2000, a cp_5 cell with both a telomerase activation and a block of differentiation at tick 5000 and a cp_5 cell with a telomerase activation, a block of differentiation and transmitter independence at tick 8000.

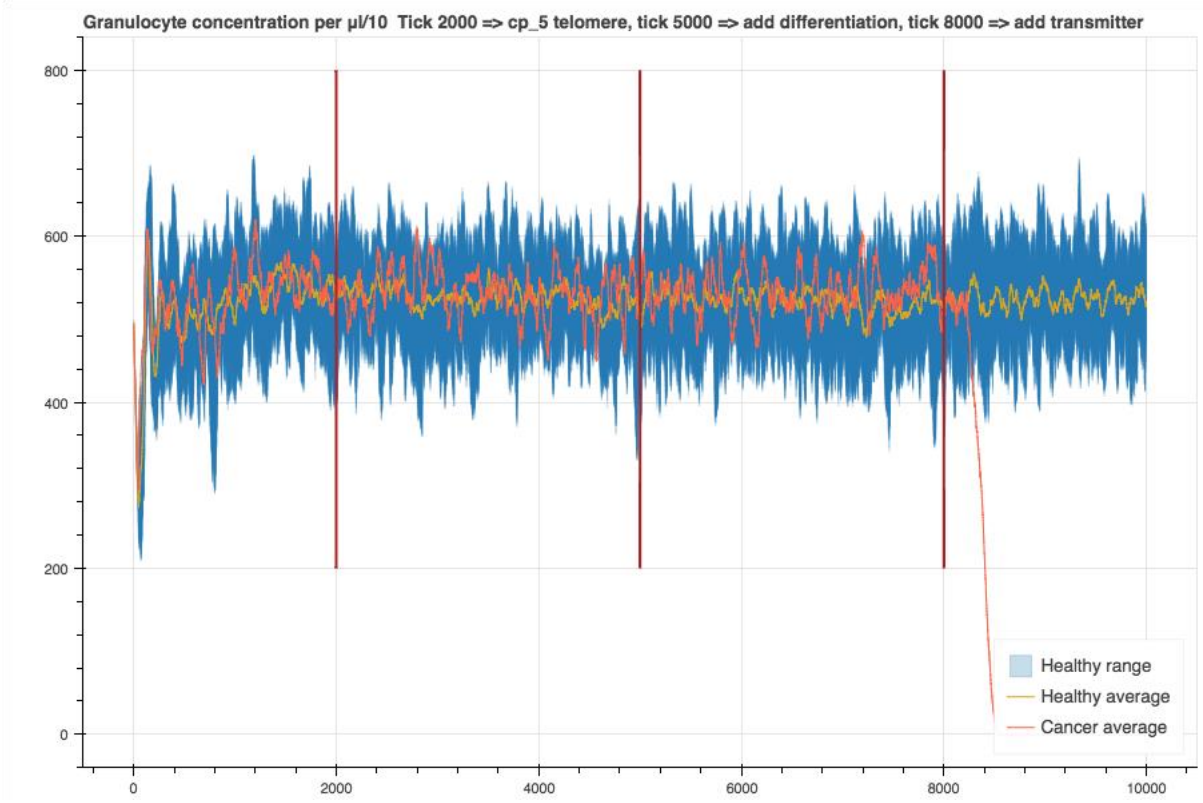


Figure 32 Granulocyte concentration per µl/10, Tick 2000: cp_5 cell with telomerase activation, Tick 5000: differentiation block, Tick 8000: transmitter-independent division, Worm, 2016

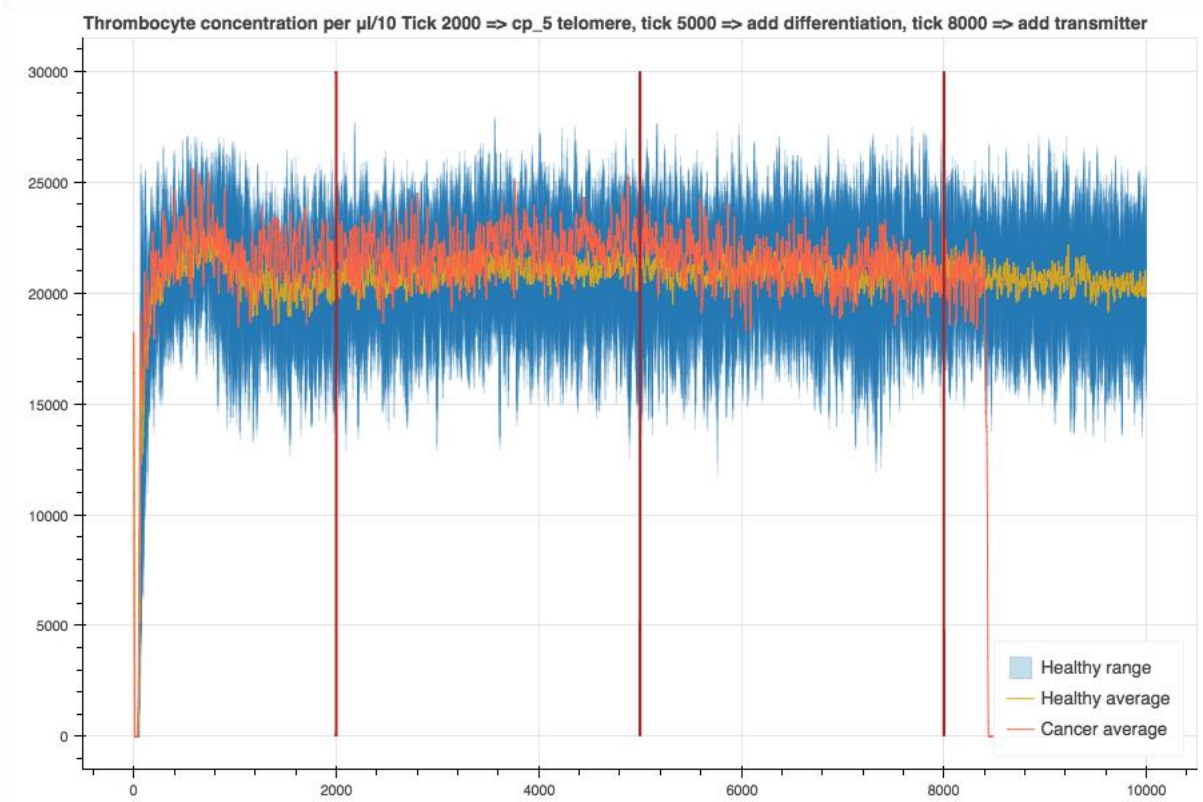


Figure 33 Thrombocyte concentration per µl/10, Tick 2000: cp_5 cell with telomerase activation, Tick 5000: differentiation block, Tick 8000: transmitter-independent division, Worm, 2016

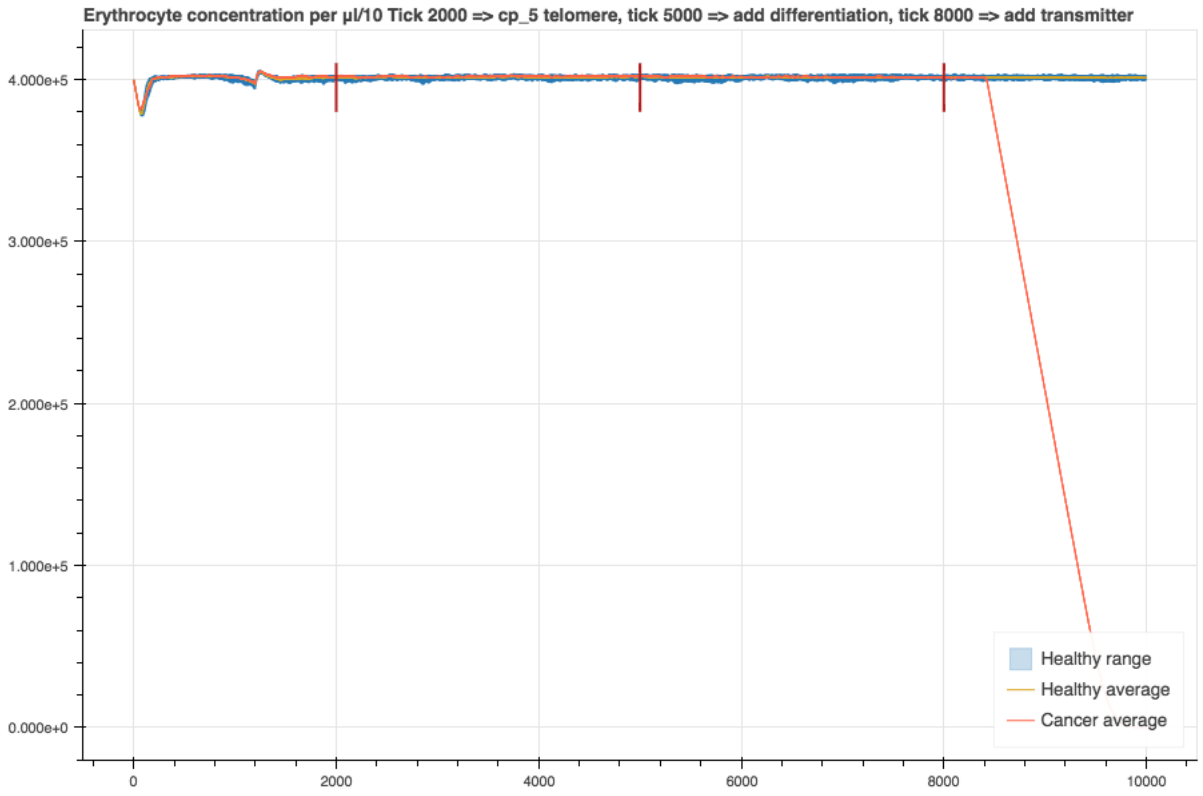


Figure 34 Erythrocyte concentration per µl/10, Tick 2000: cp_5 cell with telomerase activation, Tick 5000: differentiation block, Tick 8000: transmitter-independent division, Worm, 2016

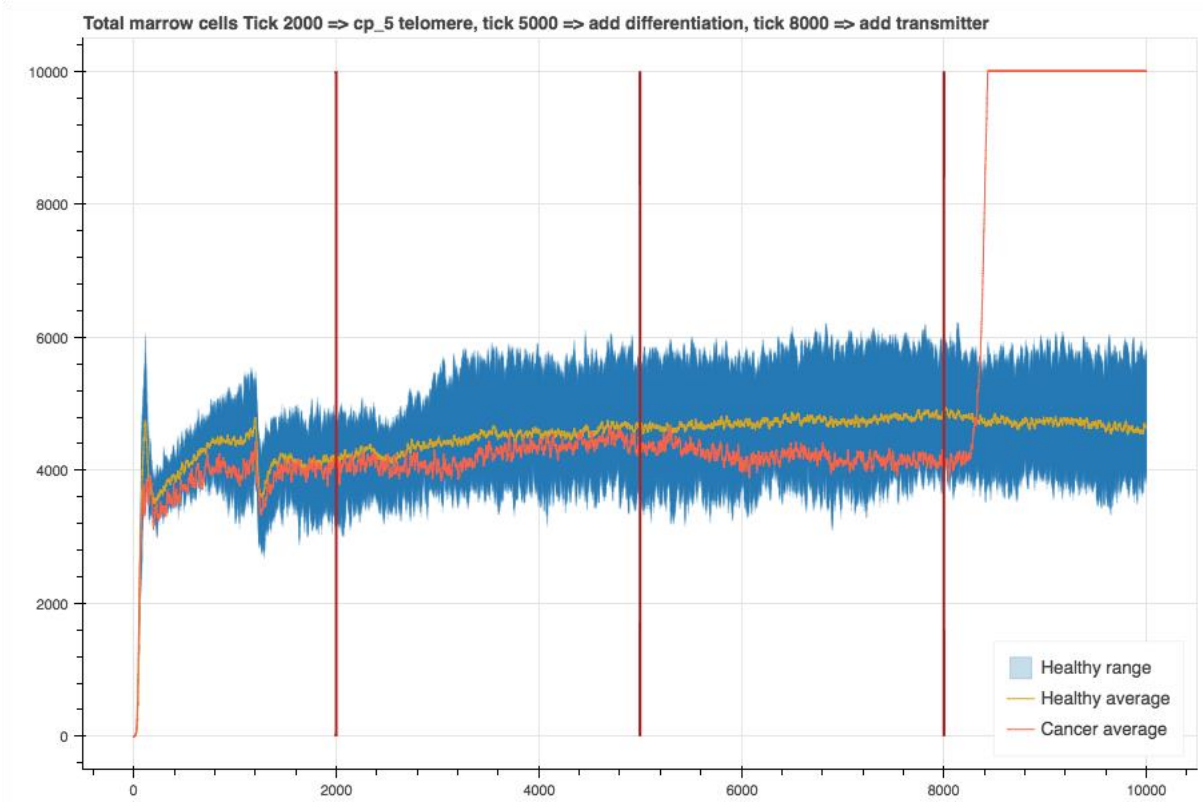


Figure 35 Total marrow cells, Tick 2000: cp_5 cell with telomerase activation, Tick 5000: differentiation block, Tick 8000: transmitter-independent division, Worm, 2016

In contrast to the first two sets of experimental runs, here a significant effect can be found shortly after tick 8000. While the concentrations of granulocytes, thrombocytes and erythrocytes decrease dramatically (Fig. 40 – 42), the total number of cells in the bone marrow increases significantly (Fig. 43). These findings correspond to the clinical course of AML and its effects on the bone marrow. While other tested orders showed similar results, one combination provided unexpected effects. In the set of experimental runs with a cp_5 cell with transmitter independent cell division at tick 2000 and an additional block of differentiation at tick 5000, the following effects were observed.

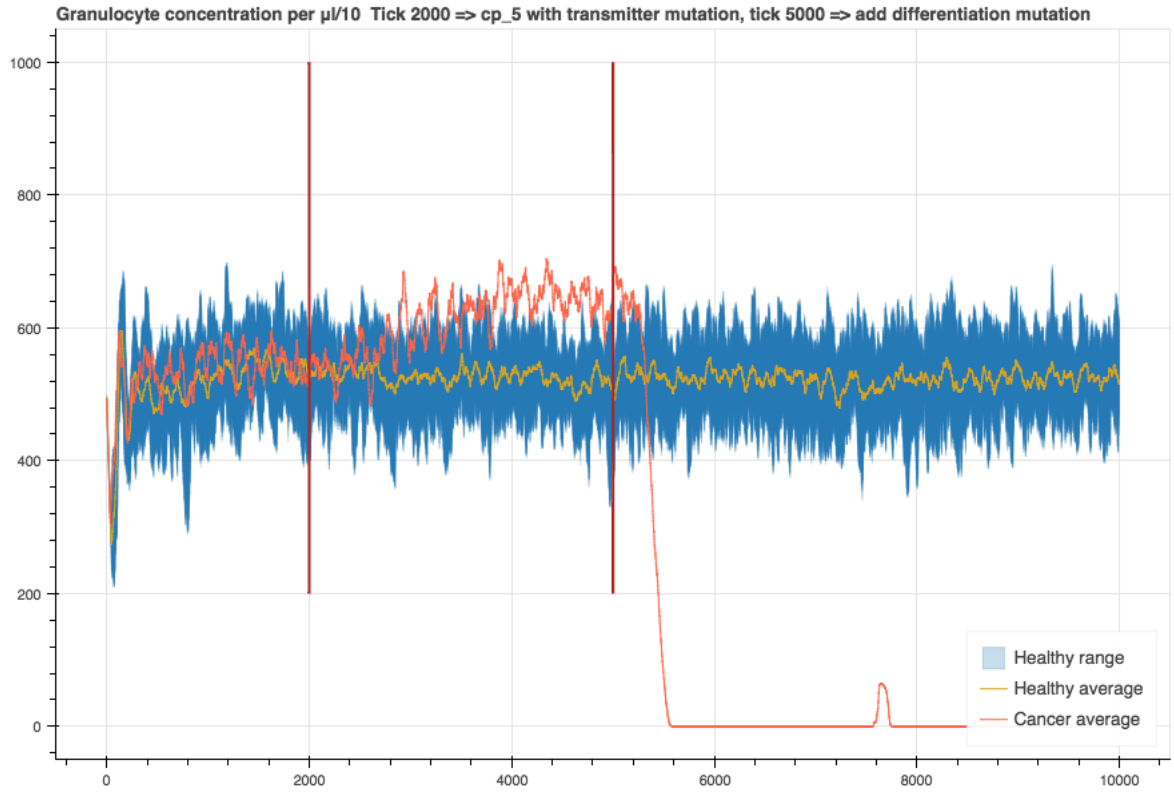


Figure 36 Granulocyte concentration per µl/10, Tick 2000: cp_5 cell with transmitter-independent division, Tick 5000: differentiation block, Worm, 2016

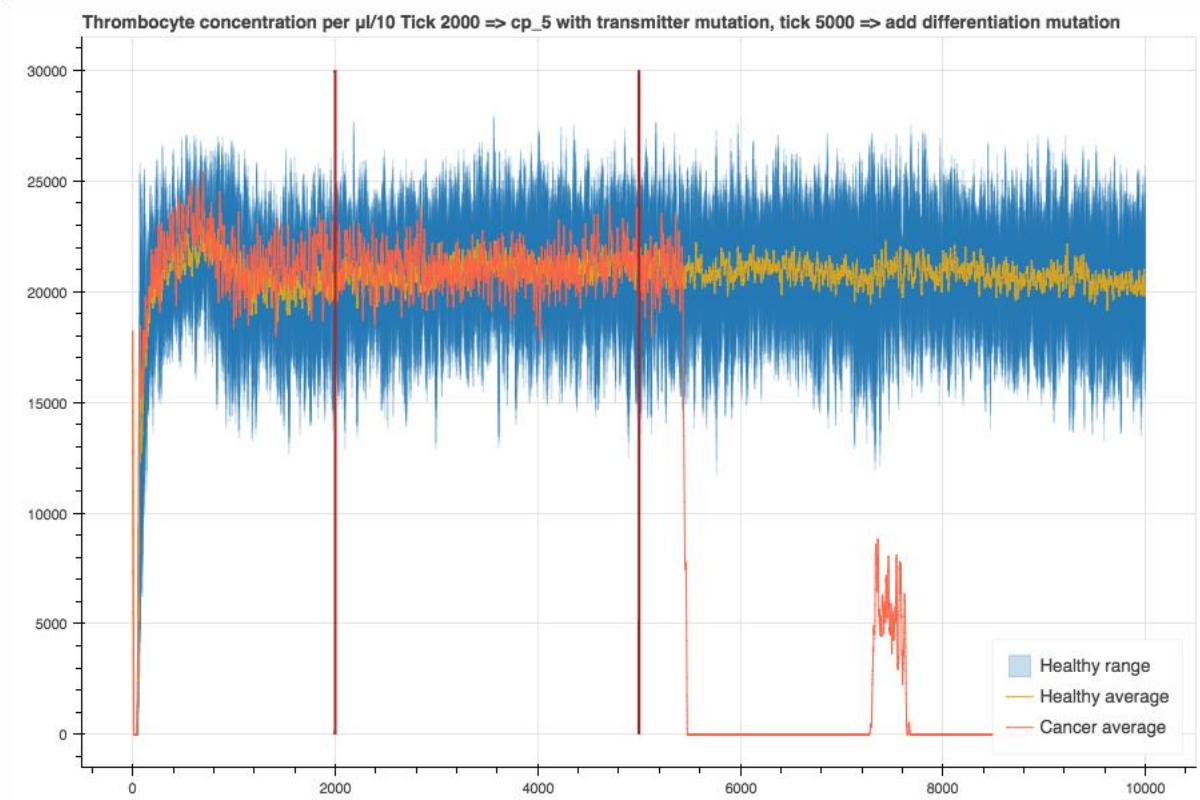


Figure 37 Thrombocyte concentration per µl/10, Tick 2000: cp_5 cell with transmitter-independent division, Tick 5000: differentiation block, Worm, 2016

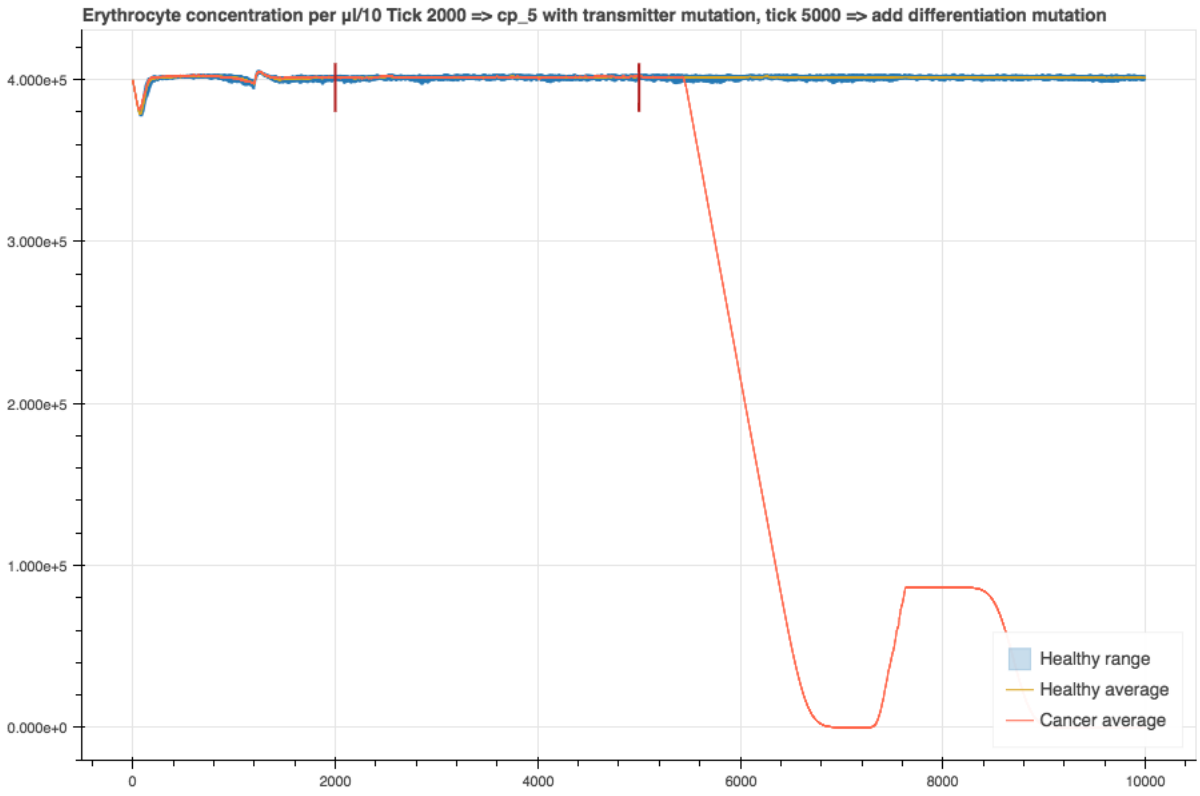


Figure 38 Erythrocyte concentration per µl/10, Tick 2000: cp_5 cell with transmitter-independent division, Tick 5000: differentiation block, Worm, 2016

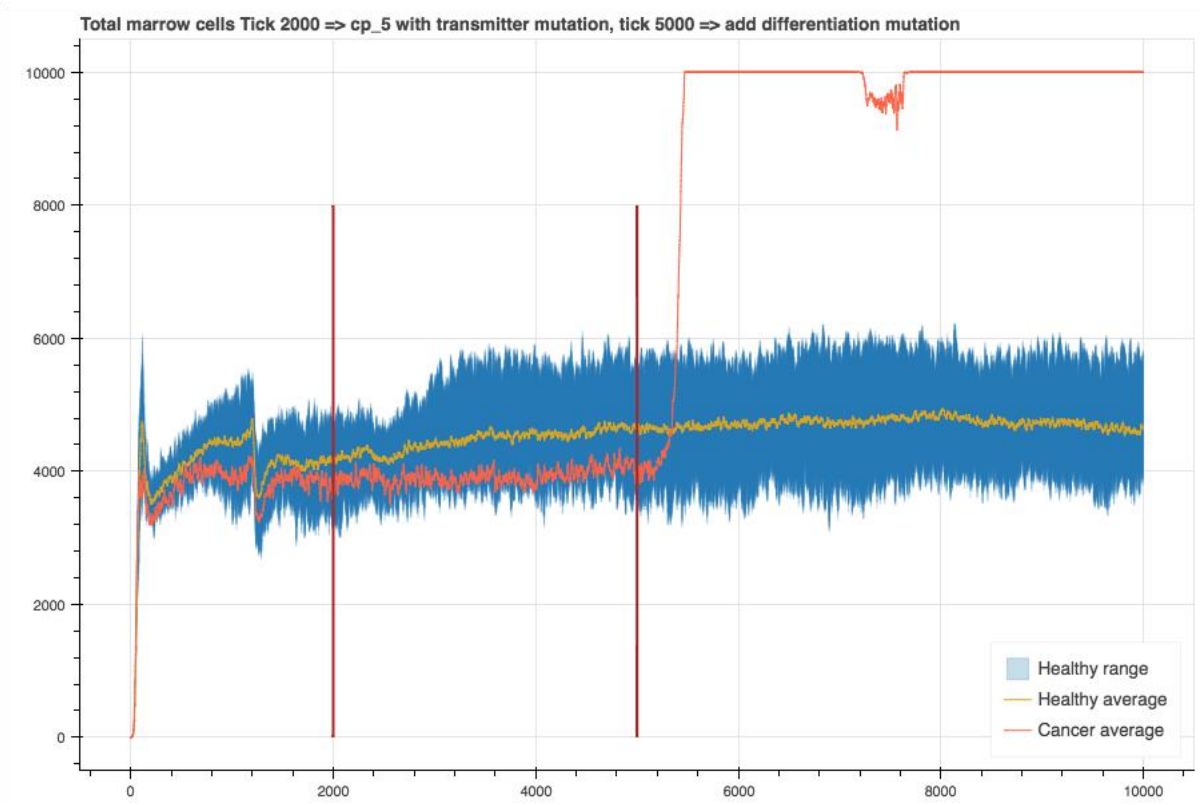


Figure 39 Total marrow cells, Tick 2000: cp_5 cell with transmitter-independent division, Tick 5000: differentiation block, Worm, 2016

While a low effect was seen after tick 2000, after the addition of a differentiation block at tick 5000, concentrations of granulocytes, thrombocytes and erythrocytes decreased extensively (Fig. 44 – 46), while the total number of bone marrow cells increased (Fig. 47). However, in contrast to the findings from the sets with the introduction of three alterations, all values seem to recover at tick 7000 to worsen again at tick 8000 (Fig. 44 – 47). While the exact mode of action remains unknown here, the effect of the addition of the differentiation block is remarkable. Still, the system shows an oscillating behavior instead of the picture of a full-blown AML, like in the sets with three different alterations.

In conclusion, our experiments indicate, that, except for the surprising outcome elucidated above, there are always three alterations necessary to initiate a highly malignant AML. However, the block of differentiation seems to play the most crucial role, as it already shows an effect, when added to transmitter independence.

5.2 The Hallmark Concept Revisited

The simulation models and the performed experiments depicted above to a large extent confirm the Hallmark concept extracted and synthesized from the literature review. Furthermore, our findings indicate that a reduction similar to the classification

presented by Vogelstein et al. can be reasonable, supplemented by an additional category for differentiation block and acquisition of stem cell features.

Hanahan & Weinberg 2011	Vogelstein et al. 2013	Groten et al. 2016				Torrente et al. 2016
		Exp 1	**Exp 2**	**Exp 3**	**Exp 4**	
Hallmarks of Cancer	Core Cellular Processes	CurrentCellsMax (Primary Tumor)	AllCells	ImmuneAttacked	Leukemia	Genes overexpressed in cancer
Evading Growth Suppressors	**Cell survival**	**1**	**1**			**EMR2**
Sustaining Proliferative Signaling		**2**	**2**	- 2	**I**	**PTP4A3 (**)**
						RPS6KB1
						RGS1
Tumor-Promoting Inflammation		**3**	**4**	**4**		**TREM2**
						MAP3K12
Evading Immune Destruction		**4a**				**TDO2**
						ANXA11
Enabling Replicative Immortality	**Genome maintenance**	**4b**	**3**	**2**	**II**	**?**
Genome Instability and Mutation				**1**		**DDX11**
						BLM
						NUDT1
						NUDT1
Activating Invasion and Metastasis	**Cell Fate**	*- 1*	*- 1*	*- 1*		**PTP4A3 (**)**
						MMP9
New: Stem Cell properties, Block of Differentiation, MET					**III**	**LEF1**

Resisting Cell Death		low effect		none
Inducing Angiogenesis				
Reprogramming Energy Metabolism				

Table 9 The Hallmark Concept Revisited, 1-4b: Relevance Range after Simulating Carcinogenesis, I-III: Final Relevance Range after Simulating Hematopoiesis (Synthesis), Hanahan & Weinberg, 2011, Vogelstein et al., 2013, Groten et al., 2016, Torrente et al., 2016

In *Table 11* the hallmarks from *Table 3* are compared to the results of the simulated experiments. The “Hallmarks of Cancer” have been ranged from 1 to 4 (4a and 4b, respectively) due to their calculated impact on the three different end points, “CurrentCellsMax”, “AllCells” and “ImmuneAttacked”, explained in detail in the paragraph “Simulating Carcinogenesis – The Validation Process”. A fourth column represents the three hallmarks applied in the simulation of hematopoiesis to develop a state of AML (see paragraph “Simulating Leukemogenesis”).

All three experiments in the simulation of carcinogenesis indicate that the two most crucial traits for cell growth are “Sustaining Proliferative Signaling” and “Evading Growth Suppressors”, followed by secondary alterations regarding “Tumor-Promoting Inflammation” and “Enabling Replicative Immortality”.

More generally speaking, these are the alterations in cellular processes of “Cell Survival” and “Genome Maintenance”, which dominate cell growth in the simulation of carcinogenesis. These two major hallmarks were transferred to the simulation of hematopoiesis. A hallmark presumably missing in the first simulation, as well as the common concepts of cancer traits, is the lack of differentiation and closely associated stem cell features of a cancer cell. We did not investigate the separate introduction of a differentiation blockade in the simulation model of carcinogenesis since aspects of differentiation were not included there. The model as presented essentially assumes a block of differentiation by exclusion. This gap becomes evident in the simulation of cancer in a specific tissue system, here the hematopoietic tissue. Applying the lack of differentiation to the simulation as a third cellular alteration, one can simulate leukemogenesis with largely realistic and plausible results. This result corresponds to the suggestions of Vogelstein et al. (Vogelstein et al. 2013; Vogelstein & Kinzler 2015a) and the distribution of genes found to overexpressed in different cancer types (Torrente et al. 2016).

6 Discussion

Cancer is still the second-leading cause of death worldwide. Recently, novel concepts and rethinking of previously published concepts are changing the oncologic research landscape (Lowy & Collins 2016). To address the rising need for innovative research approaches (Lowy & Collins 2016) the overall aim of this project was to investigate the essential phenotypic “hallmarks” of a cancer cell, oriented to the “hallmarks of cancer” suggested by Hanahan and Weinberg (Hanahan & Weinberg 2000; Hanahan & Weinberg 2011), and extended by findings of pertinent literature about cancer history, cancer hallmarks, genetic hallmarks, cancer therapy, biological and *somatic evolution*, entropy and chance and recent research objectives. Based on this literature research we developed an *in silico* simulation model through a hallmark synthesis and, vice versa revisited the identified characteristics via the modeled simulation. Aiming for a novel tool for teaching as well as for basic, clinical and therapeutic research, we thereby focused on visualization and interactivity of the simulation. Implementing this feature utilizing a simulation model, which bears the advantage of *time* as an important measurable parameter, we provide a model, which extends the investigative breadth of previous analytical models. Certainly, other simulation models have been developed in cancer research, for example, to simulate targeted therapies (Komarova & Wodarz 2016) or to investigate the correlation between spatial cancer cell expansion and tumor morphology (Waclaw, Bozic, Pittman, Hruban, Vogelstein, et al. 2015). In comparison to our model, the simulation models of targeted therapy provide small amounts of variable parameters, which are independent of each other and therefore need extensive mathematical descriptions and rules. Complex probability calculations have to be performed to achieve the desired simulation. Even though these simulations might appear more complex and elaborated, they do not allow the investigation of a high amount of parameters, nor end points. Especially, the possibility to investigate interactions of parameters is restricted. In contrast, in our model the true values of parameters result from the simulated dynamic processes and interactions, which are possible because of the simulation of single pathways by algorithms iteratively adapted during the simulation process. Additionally, both models mentioned above do not provide the possibility to interact simultaneously and adjust the simulation via a user-friendly

surface, which is enabled by both of our simulations, which contain interactive buttons which can be directly adjusted.

Qualitative validity, regarding the resemblance of outcomes to evidence-based findings, could be shown by extensive Balancing and Regression Analysis with the aid of a review of pertinent literature. This way, both simulations have shown considerable coherence and plausibility. Certainly, one could argue, that the simulation results could mainly be explained as a self-fulfilling prophecy. However, first, this effect is a sign for inner coherence of a system, in which only single parameters were defined at the beginning and autonomous interactions are still possible, and second, we could still observe undetermined and unexpected outcomes, like the negative impact of “metastasis” on the maximum number of current cells in the primary tumor in the simulation of carcinogenesis.

Even though the obvious results already provide a certain qualitative validity, an additional validation process called system identification is still inevitable to declare the simulations valid. Therefore, further experiments have to be simulated with higher numbers of repetition and be compared to new wet-lab experiments. Furthermore, aiming to apply Machine Learning tools to the simulation, one would also have to reintegrate an improved probability system with continuous parameters and Gauss distributions.

As mentioned above, the simulation of carcinogenesis resulted in both confirmations of obvious evidence as well as so far underappreciated effects. Although the full validity of the simulation can be doubted for the reasons mentioned above, we would like to take these results seriously, since the majority can be explained even though they seem counterintuitive at first sight. Our findings indicate that the ten “Hallmarks” proposed by Hanahan and Weinberg can be clustered in two different groups, “Growth/Apoptosis Balance” and “Genetic Fidelity/Immortality”, and that carcinogenesis requires just one alteration in each pathway group. Modelling Hematopoiesis finally revealed one missing Hallmark Capability, “Block of Differentiation”, which we have not specifically addressed in the carcinogenesis model, which starts at time 0 already with a cancer cell population. After having reviewed the literature on cancer evolution, we propose to assign this feature to the broader term “Stem Cell Features”. These findings largely correlate with earlier suggestions by Vogelstein et al. (Vogelstein et al. 2013; Vogelstein & Kinzler 2015b). In their earlier work, they assumed cancer cells to contain alterations in the three

core cellular processes “Cell Survival”, “Cell Fate” and “Genomic Maintenance”. Out of these three processes, two directly correlate to our suggestions (Tab. 11). Their third proposal, “Cell Fate”, primarily describes the capability of metastasis, which we would like to extend to “Stem Cell Features”, including the “Block of Differentiation”. For, this assertion results from our simulations and is, according to recent findings, a fundamental trait of cancer cells, which is a prerequisite for several other pathways (Jordan, Craig T. et al. 2006; Gupta et al. 2009). The fact that our simulation of hematopoiesis depends on three different pathway alterations to result in an overt AML corresponds to recent suggestions of Vogelstein *et al.*, that three driver mutations are sufficient to initiate the majority of malignancies (Vogelstein & Kinzler 2015b).

In conclusion, we believe to have developed two simulation models, which both confirm previous assertions as well as provide novel unexpected and possibly underestimated findings, including a new hallmark classification proposal, and should be increasingly considered in prospective cancer research.

In combination with Machine Learning tools, autonomous self-learning systems, our simulations promise to contribute to a novel type of evidence and hypothesis generation in cancer research with full exploitation of computational power. The possibly enormous impact of such an approach on the current oncologic research landscape clearly merits an intensive evaluation of tools of artificial intelligence to better understand the process of carcinogenesis. Given such a powerful tool to investigate multi-parametric processes in time-lapse experiments, one might no longer be able to justify a reductionist approach, which might not be sufficient when it comes to individual human beings. Once having developed valid tools to evaluate big data, one should take a step back to clinical trials and redefine the necessary amount of data, which has to be collected to investigate these processes with the maximum efficiency. The same applies to the widely spread assertion that EBM is the best way to address huge amounts of data. In contrast, PM might be the better way here (Sugarman 2012b), enabled by encompassing simulation models to clinical challenges. Furthermore, in light of our final synthesis of hallmark capabilities, as well the ability to investigate processes and alterations at any arbitrary point in time and therefore observe a sequence of events via a simulation, one can also doubt the common end points in current cancer research. These endpoints, usually, mainly focus on remission induction and treatment-free survival. However, in light of recent

findings like the “evolutionary double-bind” (Willyard 2016), it might be more reasonable to define new end points with respect to possibilities like living with cancer, i.e. overall survival irrespective of remission rates.

7 Summary

The overall aim of this project was to investigate the fundamental phenotypic traits of a cancer cell to develop an “in silico” simulation model and, vice versa redefine the identified characteristics via the established simulation model. Thus, the focus lay on visualization and interactivity of the simulation. To achieve this aim, we addressed the following objectives.

First, the essential “Hallmarks of Cancer” have been identified, based on a literature review (Groten et al., 2016).

As a result, the identified hallmark characteristics were evaluated and, finally, synthesized. Based on this synthesis, mathematical algorithms were developed to describe the hallmark pathways of carcinogenesis. Subsequently, a computational simulation of carcinogenesis has been drawn up employing these mathematical algorithms. In the next step, the proposed algorithms and correlations have been tested, validated and adapted through the simulation in several repetitive phases. To achieve a more reliable and valid simulation, we transferred the novel insights won from the first simulation to the simulation of processes in specific cell populations arising in hematopoiesis. This model was used to simulate normal hematopoietic tissue homeostasis, two clinical scenarios, the establishment of hematopoiesis after stem cell transplantation, as well as leukemogenesis. As a result, both simulation models were proved to be qualitatively valid regarding the resemblance of outcomes to evidence-based findings documented by pertinent literature. Also, both simulation models presented unexpected, but plausible outcomes, which were not directly defined by mathematical algorithms, and provide new insight into the probable process of carcinogenesis. Our findings indicate that the ten “Hallmarks” proposed by Hanahan and Weinberg can be assigned to two different groups, “Growth/Apoptosis Balance” and “Genetic Fidelity/Immortality”, and that carcinogenesis requires just one alteration in each pathway group. Modelling Hematopoiesis finally revealed one missing Hallmark Capability, “Block of Differentiation”, which we propose to assign to the broader term “Stem Cell Features”. These findings largely correlate with earlier suggestions by Vogelstein et al. (Vogelstein et al. 2013; Vogelstein & Kinzler 2015b). Beyond that, our classification proposal offers a novel and eventually more accurate perspective of carcinogenesis.

In conclusion, we believe to have developed two simulation models, which both depict previous assertions as well as provide novel unexpected, hypo generating and possibly underestimated insights and should be increasingly incorporated into prospective oncologic research. Certainly, further validation steps will have to be performed, among other things for quantitative predictability. However, in sight of the correctness of the basic concept, it promises to contribute to a novel type of evidence and hypothesis generation in cancer research objectives, especially in future conjunction with Machine Learning tools, which allow time-lapse experiments, independent self-learning of a system and, thus, full exploitation of computational power.

8 References

Alli I (2013) *101 Selected Sayings Mahatma Gandhi*,

Almendro V, Marusyk A & Polyak K (2013) Cellular Heterogeneity and Molecular Evolution in Cancer. *Annu. Rev. Pathol. Mech. Dis* 8, 277–302.

Alshalalfa M, D. Bader G, Bismar TA, Alhajj R, Martens-Uzunova E, Jalava S, Dits N, Leenders G, Moller S, Porkka K, Pfeiffer M, Waltering K, Vessella R, Tammela T, Trang P, Weidhaas J, Slack F, Ozen M, Creighton C, Ozdemir M, Ittmann M, Srivastava A, Suy S, Collins S, Kumar D, Brase J, Johannes M, Schlomm T, Falth M, Haese A, Schaefer A, Jung M, Mollenkopf H, Wagner I, Stephan C, Griffiths-Jones S, Saini H, Dongen S, Enright A, Esquela-Kerscher A, Slack F, Zhang B, Pan X, Cobb G, Anderson T, Goh W, Oikawa H, Sng J, Sergot M, Wong L, Sass S, Dietmann S, Burk U, Brabletz S, Lutter D, Hsu C, Juan H, Huang H, Liang H, Li W, Sualp M, Can T, Xu J, Li C, Lv J, Li Y, Xiao Y, Schmeier S, Schaefer U, Essack M, Bajic V, Huang J, Babak T, Corson T, Chua G, Khan S, Zhang W, Edwards A, Fan W, Flemington E, Zhang K, Zhang W, Edwards A, Fan W, Zhu D, Zhang K, Alter O, Brown P, Botstein D, Varshavsky R, Gottlieb A, Linial M, Horn D, Ruepp A, Brauner B, Dunger-Kaltenbach I, Frishman G, Montrone C, Xiao F, Zuo Z, Cai G, Kang S, Gao X, Hsu S, Lin F, Wu W, Liang C, Huang W, Subramanian A, Tamayo P, Mootha V, Mukherjee S, Ebert B, Wach S, Nolte E, Szczyrba J, Stohr R, Hartmann A, Blower P, Verducci J, Lin S, Zhou J, Chung J, Varambally S, Yu J, Laxman B, Rhodes D, Mehra R, Yu Y, Landsittel D, Jing L, Nelson J, Ren B, Sboner A, Demichelis F, Calza S, Pawitan Y, Mucci L, Pawitan Y, Bjohle J, Amler L, Borg A, Egyhazi S, Priness I, Maimon O, Ben-Gal I, Moon Y, Rajagopalan B, Lall U, Sumazin P, Yang X, Chiu H, Chung W, Iyer A, Huang D, Sherman B, Lempicki R, Ringner M, Fredlund E, Hakkinen J, Staaf J, Pang Y, Young C, Yuan H, Yang XG, Abbas A, Gupta S, Lin K, Wang Y, Chen C, Ho C, Su W, Kiviniemi J, Kallajoki M, Kujala I, Matikainen M, Alanen K, Chen J, Li J, Kiriluk K, Rosen A, Paner G, Ding Z, Wu C, Chu G, Xiao Y, Ho D, Zhang L, Altuwaijri S, Deng F, Chen L, Lal P, Merico D, Isserlin R, Stueker O, Emili A & Bader G (2013) Coordinate MicroRNA-Mediated Regulation of Protein Complexes in Prostate Cancer P. V. Benos, ed. *PLoS One* 8, e84261.

Batut B, Parsons DP, Fischer S, Beslon G, Knibbe C, McCutcheon J, Moran N, Moran N, McCutcheon J, Nakabachi A, Moran N, McLaughlin H, Sorek R, Giovannoni S, Tripp H, Givan S, Podar M, Vergin K, Baptista D, Bibbs L, Eads J, Richardson T, Noordewier M, Rappé M, Short J, Carrington J, Mathur E, Rocap G, Larimer F, Lamerdin J, Malfatti S, Chain P, Ahlgren N, Arellano A, Coleman M, Hauser L, Hess W, Johnson Z, Land M, Lindell D, Post A, Regala W, Shah M, Shaw S, Steglich C, Sullivan M, Ting C, Tolonen A, Webb E, Zinser E, Chisholm S, Partensky F, Garczarek L, Yu T, Li J, Yang Y, Qi L, Chen B, Zhao F, Bao Q, Wu J, Luo H, Friedman R, Tang J, Hughes A, Hindré T, Knibbe C, Beslon G, Schneider D, Nilsson A, Koskiniemi S, Eriksson S, Kugelberg E, Hinton J, Andersson D, Barrick J, Yu D, Yoon S, Jeong H, Oh T, Schneider D, Lenski R, Kim J, Adami C, Mozhayskiy V, Tagkopoulos I, Parsons D, Knibbe C, Beslon G, Misevic D, Frénoy A, Parsons D, Taddei F, Frénoy A, Taddei F, Misevic D, Beslon G, Parsons D, Sanchez-Dehesa Y, Peña J-M, Knibbe C, Cuypers T, Hogeweg P, Lenski R, Ofria C, Collier T, Adami C, Wilke C, Wang J, Ofria C, Lenski R, Adami C, Lynch M, Conery J, Kuo C-H, Moran N, Ochman H, Mira A, Moran N, Partensky F, Hess W, Vaulot D, Knibbe C, Coulon A, Mazet O,

Fayard J-M, Beslon G, Daubin V, Moran N, Mira A, Ochman H, Moran N, Moran N, Plague G, Knibbe C, Mazet O, Chaudier F, Fayard J-M & Beslon G (2013) In silico experimental evolution: a tool to test evolutionary scenarios. *BMC Bioinformatics* 14, S11.

Beerenwinkel N, Schwarz RF, Gerstung M & Markowetz F (2015) Cancer evolution: mathematical models and computational inference. *Syst. Biol.* 64, e1-25.

Blokh D & Stambler I (2016) The application of information theory for the research of aging and aging-related diseases. *Prog. Neurobiol.*

Blokh D, Stambler I, Afrimzon E, Shafran Y, Korech E, Sandbank J, Orda R, Zurgil N, Deutsch M, Floyd CE, Yun AJ, Sullivan D, Kornguth P, Furundzic D, Djordjevic M, Bekic AJ, Pendharkar PC, Roger JA, Yaverbaum GJ, Herman N, Benner M, Wolberg WH, Mangasarian OL, Mangasarian OL, Street WN, Wolberg WH, Tou JT, Gonzalez RC, Verhayen CJDM, Duin RPW, Groen FCA, Joosen JC, Verbeek PW, Gelfand IM, Rosenfeld BI, Shifrin MA, Nieddu L, Patrizi G, Ben-Ze'ev A, Bershadsky AD, Geiger B, Rosen D, Berke G, Sunray M, Zurgil N, Shafran Y, Deutsch M, Eisenthal R, Cornish-Bowden A, Tinoco I, Sauer K, Wang JC, Puglisi JD, Yourno J, Burkart P, Lizzi F, Tartaglia A, Elghetany MT, Feng JM, Wu JS, Compagnoni AT, Chen WF, Neubauer A, Valet G, Huhn D, Zschunke F, Salmassi A, Kriepe H, Parwaresch MR, Radzun HJ, Battiti R, Kwak N, Choi CH, Tourassi GD, Frederick ED, Floyd CE, Lucas PJ, Abu-Hanna A, Podgorelec V, Kokol P, Stiglic B, Rozman I, Adam BL, Qu Y, Davis JW, Ward MD, Clements MA, Cazares LH, al. et, Jerez-Aragonez JM, Gomez-Ruiz JA, Ramos-Jimenez G, Munoz-Perez J, Alba-Conejo E, Kao JPY, Rosen GM, Blokh D, Afrimzon E, Stambler I, Korech E, Shafran Y, Zurgil N, al. et, Chou WC, Neifeld MA, Xuan R, Zvarova J, Studeny M, Khinchin AI, Cover TM, Thomas JA, Shannon CE, Nicolis G, Prigogine I, Fraser AM, Swinney HL, Hill DLG, Batchelor PG, Holden M, Hawkes DJ, Quinlan JR, Blokh AS, Shalyto AA, Head JF, Elliott RL, McCoy JL, Wiltschke C, Krainer M, Budinsky AC, Berger A, Muller C, Zeillinger R, al. et, Schwartz RH, Watson JV, Dive C, Bruheim P, Eimhjellen K, Zurgil N, Shafran Y, Fixler D, Deutsch M, Eisenthal A, Marder O, Dotan D, Baron S, Lifschitz-Mercer B, Chaitchik S, al. et, Kaplan MR, Trubnikov E, Berke G, Babcock DF, Malin-Berdel J, Valet G, Deutsch M, Kaufman M, Shapiro H, Zurgil N, Chen TL, Passos-Coelho JL, Noe DA, Kennedy MJ, Black KC, Colvin OM, al. et, Katz-Brull R, Seger D, Rivenson-Segal D, Rushkin E, Degani H, Greaves MF & Bauminger S (2007) The information-theory analysis of Michaelis-Menten constants for detection of breast cancer. *Cancer Detect. Prev.* 31, 489–98.

Bundesamt S (2014) Todesursachen in Deutschland. *Stat. Bundesamt* 12.

Cairns J (1975) Mutation selection and the natural history of cancer. *Nature* 255, cp1-.

De Carvalho DD, Sharma S, You JS, Su S-F, Taberlay PC, Kelly TK, Yang X, Liang G & Jones PA (2012) DNA Methylation Screening Identifies Driver Epigenetic Events of Cancer Cell Survival. *Cancer Cell* 21, 655–667.

Coussens LM & Werb Z (2002) Inflammation and Cancer. *Nature*.

Deutsches Krebsforschungszentrum (2016) Zahlen und Fakten - Deutsches Krebsforschungszentrum. Available at: https://www.dkfz.de/de/dkfz/quick-facts.html [Accessed October 4, 2016].

Dohner H (2015) Acute Myeloid Leukemia. *N. Engl. J. Med.* 373, 1136–52.

Doulatov S, Notta F, Laurenti E & Dick JE (2012) Hematopoiesis: A human perspective. *Cell Stem Cell* 10, 120–136.

Ernest Belfort Bax (1786) *The Metaphysical Foundations of Natural Science*,

Fan R, Zhong M, Wang S, Zhang Y, Andrew A, Karagas M, Chen H, Amos CI, Xiong M & Moore JH (2011) Entropy-based information gain approaches to detect and to characterize gene-gene and gene-environment interactions/correlations of complex diseases. *Genet. Epidemiol.* 35, 706–21.
Greaves M & Maley CC (2012) Clonal evolution in cancer. *Nature* 481, 306–313.
Groten J, Borner C, Mertelsmann R (2016) Understanding and Controlling Cancer: The Hallmark Concept Revisited - Evolution and Entropy
Gupta PB, Chaffer CL & Weinberg RA (2009) Cancer stem cells: mirage or reality?
Hanahan D & Weinberg RA (2011) Hallmarks of Cancer: The Next Generation. *Cell* 144, 646–674.
Hanahan D & Weinberg RA (2000) The Hallmarks of Cancer. *Cell* 100, 57–70.
Hayes DN & Kim WY (2015) The next steps in next-gen sequencing of cancer genomes. *J. Clin. Invest.* 125, 462–468.
IARC (2016a) About IARC - IARC History. Available at: https://www.iarc.fr/en/about/iarc-history.php [Accessed October 4, 2016].
IARC (2016b) Globocan 2012. *Http://Www-Dep.Iarc.Fr/* GLOBOCAN, 2012–2013.
Jordan, Craig T. PD, Guzman, Monica L. PD & Noble, Mark PD (2006) Cancer Stem Cells. *N. Engl. J. Med.* 355, 1253–61.
Justesen N, Mahlmann T & Togelius J Online Evolution for Multi-Action Adversarial Games.
Klein CA (2013) Selection and adaptation during metastatic cancer progression. *Nature* 501, 365–372.
Komarova NL & Wodarz D (2016) Targeted Cancer Treatment in Silico Small Molecule Inhibitors and Oncolytic Viruses.
De Kouchkovsky I & Abdul-Hay M (2016) “Acute myeloid leukemia: a comprehensive review and 2016 update.” *Blood Cancer J.* 6, e441.
Lowy DR & Collins FS (2016) Aiming High — Changing the Trajectory for Cancer. *N. Engl. J. Med.* 374, 1901–1904.
Luzzatto L & Pandolfi PP (2015) Causality and Chance in the Development of Cancer. *N. Engl. J. Med.* 1, 84–88.
Mackey MC, Santillán M, Tyran-Kamińska M & Zeron ES (2015) The utility of simple mathematical models in understanding gene regulatory dynamics. *In Silico Biol.* 12, 23–53.
Manesso E, Teles J, Bryder D & Peterson C (2013) Dynamical modelling of haematopoiesis: an integrated view over the system in homeostasis and under perturbation. *J. R. Soc. Interface* 10, 20120817.
Merlo LMF, Pepper JW, Reid BJ & Maley CC (2006) Cancer as an evolutionary and ecological process. *Nature* 6, 924–935.
Mertelsmann R & Georg M (2016) Cancer : Modeling evolution and natural selection , the „ Mitosis Game “.
National Cancer Institute (2016a) National Cancer Act of 1937. *Natl. Cancer Inst.* Available at: http://www.cancer.gov/about-nci/legislative/history/national-cancer-act-1937.
National Cancer Institute (2016b) National Cancer Act of 1971. *Natl. Cancer Inst.* Available at: http://www.cancer.gov/about-nci/legislative/history/national-cancer-act-1971.
Nature (2016) Bioinformatics. *Nature*. Available at: http://www.nature.com/subjects/bioinformatics [Accessed July 5, 2016].
Nowell PC (1976) The clonal evolution of tumor cell populations. *Science* 194, 23–8.
Obama B (2016) Memorandum -- White House Cancer Moonshot Task Force.

Orkin SH & Zon LI (2008) Hematopoiesis: An Evolving Paradigm for Stem Cell Biology. *Cell* 132, 631–644.
Papaemmanuil E, Gerstung M, Bullinger L, Gaidzik V, Paschka P, Roberts N, Potter N, Heuser M, Thol F, Bolli N, Gundem G, Van, Loo P, Martincorena I, Ganly P, Mudie L, McLaren S, O'Meara S, Raine K, Jones D, Teague J, Butler A, Greaves M, Ganser A, Döhner K, Schlenk R, Döhner H & Campbell P (2016) Genomic classification and prognosis in acute myeloid leukemia. *N Engl J Med* in press.
Pedregosa et. al. (2011) Scikit-learn: Machine Learning in Python. *JMLR* 12, 2825–2830.
Peixoto D, Dingli D & Pacheco JM (2011) Modelling hematopoiesis in health and disease. *Math. Comput. Model.* 53, 1546–1557.
Pelossof R, Singh I, Yang JL, Weirauch MT, Hughes TR & Leslie CS (2015) Affinity regression predicts the recognition code of nucleic acid–binding proteins. *Nat. Biotechnol.* 33, 1242–1249.
Riedmiller M, Gabel T, Hafner R & Lange S (2009) Reinforcement learning for robot soccer. *Auton. Robots* 27, 55–73.
Schell J (2015) *The Art of Game Design: A Book of Lenses, Second Edition*,
Shay JW & Wright WE (2000) Hayflick, his limit, and cellular ageing. *Nat. Rev. Mol. Cell Biol.* 1, 72–76.
Sugarman J (2012a) Questions Concerning the Clinical Translation of Cell-Based Interventions under an Innovation Pathway. *J. Law, Med. Ethics* 40, 945–950.
Sugarman J (2012b) Questions concerning the Clinical Translation of Cell-Based Interventions under an Innovation Pathway. *J. Law, Med. Ethics* 40.
Suthaharan S (2016) *Machine Learning Models and Algorithms for Big Data Classification*, Boston, MA: Springer US.
Sütterlin T (2015) *Entwurf und Realisierung eines computergestützten Systems zur in silico Modellierung und Simulation von Epithelgeweben*.
Tomasetti C & Vogelstein B (2015) Variation in cancer risk among tissues can be explained by the number of stem cell divisions. *Science (80-.).* 347, 78–81.
Torrente A, Lukk M, Xue V, Parkinson H, Rung J & Brazma A (2016) Identification of Cancer Related Genes Using a Comprehensive Map of Human Gene Expression. *PLoS One* 11, e0157484.
UICC (2014) Introduction to UICC.
Vogelstein B & Kinzler KW (2015a) The Path to Cancer - Three Strikes and You're out. *N. Engl. J. Med.* 37320.
Vogelstein B & Kinzler KW (2015b) The Path to Cancer — Three Strikes and You're Out. *N. Engl. J. Med.* 373, 1895–1898.
Vogelstein B, Papadopoulos N, Velculescu VE, Zhou S, Diaz LA, Kinzler KW, Wood LD, Parsons DW, Jones S, Govindan R, Gryfe R, Gallinger S, Palles C, Nowell PC, Fearon ER, Vogelstein B, Kinzler KW, Vogelstein B, Jones S, Bozic I, Tomasetti C, Vogelstein B, Parmigiani G, Laurenti E, Dick JE, Welch JS, Yachida S, Kerbel RS, Bernards R, Weinberg RA, Yu M, Stott S, Toner M, Maheswaran S, Haber DA, Komori J, Boone L, DeWard A, Hoppo T, Lagasse E, Pelizzola M, Ecker JR, Parmigiani G, Meyerson M, Gabriel S, Getz G, Carter H, Youn A, Simon R, Kaminker JS, Zhang Y, Watanabe C, Zhang Z, Michaelson JJ, Nik-Zainal S, Thiagalingam S, Forbes SA, Yan H, Zhao S, Ward PS, Dang L, Pear WS, Aster JC, Nicolas M, Weng AP, Agrawal N, Stransky N, Agrawal N, Parsons DW, Pasqualucci L, Morin RD, Grasso CS, Ellis MJ, Wiegand KC, Jones S, Jones S, Wang K, Huang J, Imielinski M, Rudin CM, Delhommeau F,

Schwartzentruber J, Wu G, Ley TJ, Dalgliesh GL, Suvà ML, Riggi N, Bernstein BE, Papaemmanuil E, Graubert TA, Yoshida K, Jiao Y, Cesare AJ, Reddel RR, Heaphy CM, Heiden MG Vander, Cantley LC, Thompson CB, Lu C, Turcan S, Stephens PJ, Bass AJ, Tomlins SA, Soda M, Armitage P, Doll R, Böttcher R, Forster M, Ding J, Beal MA, Glenn TC, Somers CM, Mertes F, Gundry M, Vijg J, Biankin A V., Yan H, Kinzler KW, Vogelstein B, Huang FW, Horn S, Xue W, Solimini NL, Beggs AD, Feinberg AP, Tycko B, Jones PA, Baylin SB, Höglund M, Gisselsson D, Säll T, Mitelman F, Shibata D, Schaeffer J, Li ZH, Capella G, Perucho M, Sottoriva A, Spiteri I, Shibata D, Curtis C, Tavaré S, Shah SP, Anderson K, Navin N, Nik-Zainal S, Gerlinger M, Xu X, Campbell PJ, Wagle N, Komarova NL, Wodarz D, Turke AB, Durrett R, Moseley S, Diaz LA, Kreso A, Stephens PJ, Pang H, Chen J, Ye Y, Sun H, Shi G, Chapman PB, Kwak EL, Ljungman M, Lane DP, Perrimon N, Pitsouli C, Shilo BZ, Kerbel RS, Chung AS, Ferrara N, Baish JW, Hynes NE, Lane HA, Turner N, Grose R, Yun J, Ying H, Araten DJ, Kunkel TA, Zhou B-BS, Elledge SJ, Medema RH, Macůrek L, Derheimer FA, Kastan MB, Ciriello G, Cerami E, Sander C, Schultz N, Yeang CH, McCormick F, Levine A, Sharma S V., Bell DW, Settleman J, Haber DA, McLeod HL, Brock DW, Lemmon MA, Schlessinger J, Arkin MR, Wells JA, Bienstock RJ, Besson A, Dowdy SF, Roberts JM, Morin PJ, He TC, Wetering M van de, Farmer H, Irshad S, Ashworth A, Tutt A, Grueneberg DA, Mao M, Kirkwood JM, Barrett T, Segal NH, Castle JC, Sampson JH, Matsushita H, DuPage M, Mazumdar C, Schmidt LM, Cheung AF, Jacks T, Challa-Malladi M, Hodi FS, Topalian SL, Dunn BK, Jegalian K, Greenwald P, Lopes LC, Barberato-Filho S, Costa AC, Osorio-de-Castro CGS, Colditz GA, Wolin KY & Gehlert S (2013) Cancer genome landscapes. *Science* 339, 1546–58.

Waclaw B, Bozic I, Pittman ME, Hruban RH, Nowak MA, Guthrie P, Road T, Square OB, Street OO, Goldman S, Cancer P, Hopkins J, Cancer K & Biology E (2015) A spatial model predicts that dispersal and cell turnover limit intratumour heterogeneity. *Nature* 525, 261–264.

Waclaw B, Bozic I, Pittman ME, Hruban RH, Vogelstein B & Nowak MA (2015) A spatial model predicts that dispersal and cell turnover limit intratumour heterogeneity. *Nature* 525, 261–264.

Walker DC & Southgate J (2009) The virtual cell--a candidate co-ordinator for “middle-out” modelling of biological systems. *Brief. Bioinform.* 10, 450–461.

Weber K, Thomaschewski M, Warlich M, Volz T, Cornils K, Niebuhr B, Täger M, Lütgehetmann M, Pollok J-M, Stocking C, Dandri M, Benten D & Fehse B (2011) RGB marking facilitates multicolor clonal cell tracking. *Nat. Med.* 17, 504–509.

WHO (2016a) WHO | Cancer. *WHO*. Available at: http://www.who.int/mediacentre/factsheets/fs297/en/ [Accessed June 9, 2016].

WHO (2016b) World Health Statistics 2016: monitoring health for the SDGs, sustainable development goals. , 1–136.

Willyard C (2016) Cancer : An evolving threat. *Nature* 532, 166–168.

9 Figures

10 Tables

11 Acknowledgements

I would like to thank Prof. em. Dr. Drs. h.c. Roland Mertelsmann and

Prof. Dr. Dr. h.c. Christoph Borner for their intellectual input, stimulating discussions and constant and friendly encouragement.

Furthermore, I deeply appreciate the professional support, challenging discussions and the expertise in transforming medical and biological concepts into algorithms and highly instructive computer-assisted visualizations of Maximilian Georg, Oliver Worm, Dr. Boris Lau and Dr. Sascha Lange.

I am grateful to Evguenia Alechine for proofreading the manuscript. Last not least I would like to appreciate the generous scholarship for this project provided by the Biothera Foundation, Freiburg, Germany.

12 Appendix

12.1 Full Simulation Codes

Full Code Carcinogenesis
(Maximilian Georg & Boris Lau, PSIORI GmbH)

Simulation Commented
(Maximilian Georg & Boris Lau, PSIORI GmbH)

```
var args = process.argv;

var num_expected_params = 11;


function usage_exit() {
    console.log("Usage: node mitosis_science_batch.js NUMSTEPS P1 P2 p3");
    console.log("                NUMSTEPS: number of steps the simulation is run");
    console.log("        cyclinScaleCount: [0,..,4]");
    console.log("     immuneActScaleCount: [0,..,4]");
    console.log("          teloScaleCount: [0,..,4]");
    console.log("     selectiveScaleCount: [0,..,4]");
    console.log("           hgfScaleCount: [0,..,4]");
    console.log("          vegfScaleCount: [0,..,4]");
    console.log("          parpScaleCount: [0,..,4]");
    console.log("  proapoptoticScaleCount: [0,..,4]");
    console.log("       aerobicScaleCount: [0,..,4]");
    console.log("          egfrScaleCount: [0,..,4]");
    process.exit()
}

if (args.length!= 2+num_expected_params) usage_exit();

//**** PARSE PARAMETERS ****
var num_steps = parseInt(args[2+0]);
if (isNaN(num_steps) || num_steps<0) usage_exit();



/* INIT */

// size is used to adjust the whole simulation to different screen sizes and represents a
number of pixels (this version is without visualisation)
var size = 1200;
// petri dish size
var resultCircleSize = 0.205 * size;

// standard duration of reproduction interval in ticks (30 ticks represent 1 day)
var reproCountdownStart = 30;
// halflick limit for initial cells
var hayflickStart = 72;
// standard value for hayflick reduction after proliferation
var hayflickReductionStart = 1;
// standard value for cell Speed after proliferation (pixels per tick)
var cellSpeedStart = 0.5;
// standard probability for proliferation
var growthChanceStart = 0.5;
// standard probability for avoiding to get attacked by immune system
var immuneAvoidStart = 0.95;
// standard duration of immune attack intervals
var immuneCountdownStart = 30;
// standard time it take will an attacked cell dies
var immuneResistanceStart = 40;
// standard inflammation value (this is used to modify the probability of events which depent
on inflammation)
var inflammationStart = 1;
```

```
// standard values for angiogenese, avoiding aproptosis and reprogramming energetics, this
values are used calculated the probability for avoiding cell death
var angioStart = 0.5;
var proapoptoticStart = 0.5;
var aerobicStart = 0.5;
// all tacked output values
var allCells = 0;
var deadCells = 0;
var deadCellsTemp = 0;
var newCellsTemp = 0;
var currentCells = 0;
var overlappingCells = 0;
var hayflickReached = 0;
var immuneAttacked = 0;
var cellChange = 0;
var ccCollect = 0;
var ccCount = 0;
var ccAverage = 0;
var currentCellsMin = Number.POSITIVE_INFINITY;
var currentCellsMax = -1;
var newCellsMin = Number.POSITIVE_INFINITY;
var newCellsMax = -1;
var deadCellsMin = Number.POSITIVE_INFINITY;
var deadCellsMax = -1;

// the emitter stores all imfomation needed to cerate new cells
var emitter = {
    x: 0.5 * size,
    y: 0.286 * size,
    cellSize: 5, //2.8,
    cellColor: "#daf650",
    cellStrokeColor: "#daf650",
    breakCap: 10,
    cellSpeed: 0.2,//0.1,
    reproCountdown: reproCountdownStart,
    hayflick: hayflickStart,
    hayflickReduction: hayflickReductionStart,
    growthChance: growthChanceStart,
    immuneAvoid: immuneAvoidStart,
    immuneCountdown: immuneCountdownStart,
    immuneResistance: immuneResistanceStart,
    inflammation: inflammationStart,
    angio: angioStart,
    proapoptotic: proapoptoticStart,
    aerobic: aerobicStart
};


//**** CONFIGURE VALUES ****

// stubs for unused and thus unimplemented buttons
var evo = {state: 0, inhib: []};
var entro = {state: 0};


//all values according to dossages assigned in batch.js and standard values get calculated

//Cyclin-dependent kinase inhibitors // Evading growth suppressors
var cyclinScaleCount = parseInt(args[2+1]);
if (isNaN(cyclinScaleCount) || cyclinScaleCount<0 || cyclinScaleCount>4) usage_exit();
if (entro.state && evo.inhib.indexOf(0)) emitter.growthChance = growthChanceStart * 2;
else if (evo.state && evo.inhib[evo.inhib.length-1] == 0) emitter.growthChance =
growthChanceStart * 2;
else {
    var factor = [2, 1.5, 1, 0.5, 0.2].reverse();
    emitter.growthChance = growthChanceStart * factor[cyclinScaleCount]
}

//Immune activating anti_CTLA4 mAb // Avoiding immune destruction
var immuneActScaleCount = parseInt(args[2+2]);
if (isNaN(immuneActScaleCount) || immuneActScaleCount<0 || immuneActScaleCount>4)
usage_exit();
if (entro.state && evo.inhib.indexOf(1)) emitter.immuneCountdown = immuneCountdownStart * 3;
else if (evo.state && evo.inhib[evo.inhib.length-1] == 1) emitter.immuneCountdown =
immuneCountdownStart * 3;
else {
    var factor = [3, 1.5, 1, 1/1.5, 1/3.0].reverse();
```

```
    emitter.immuneCountdown = immuneCountdownStart * factor[immuneActScaleCount];
}

//Telomerase Inhibitors // Enabling replicative immortality
var teloScaleCount = parseInt(args[2+3]);
if (isNaN(teloScaleCount) || teloScaleCount<0 || teloScaleCount>4) usage_exit();
if (entro.state && evo.inhib.indexOf(2)) emitter.hayflickReduction = 0;
else if (evo.state && evo.inhib[evo.inhib.length-1] == 2) emitter.hayflickReduction = 0;
else {
    var values = [0, 0.5, hayflickReductionStart, 1.25, 1.5].reverse();
    emitter.hayflickReduction = values[teloScaleCount];
}

// Tumor - promoting inflammation
var selectiveScaleCount = parseInt(args[2+4]);
if (isNaN(selectiveScaleCount) || selectiveScaleCount<0 || selectiveScaleCount>4)
usage_exit();
if (entro.state && evo.inhib.indexOf(3)) emitter.inflammation = inflammationStart - 0.2;
else if (evo.state && evo.inhib[evo.inhib.length-1] == 3) emitter.inflammation =
inflammationStart - 0.2;
else {
    var offset = [-0.2, -0.1, 0, 0.1, 0.2];
    emitter.inflammation = inflammationStart + offset[selectiveScaleCount];
}

//Inhibitors of HGF/c-Met // Activating invasion & metastasis
var hgfScaleCount = parseInt(args[2+5]);
if (isNaN(hgfScaleCount) || hgfScaleCount<0 || hgfScaleCount>4) usage_exit();
if (entro.state && evo.inhib.indexOf(4)) emitter.cellSpeed = cellSpeedStart * 2;
else if (evo.state && evo.inhib[evo.inhib.length-1] == 4) emitter.cellSpeed = cellSpeedStart *
2;
else {
    var factor = [2, 1.5, 1, 0.5, 0.2].reverse();
    emitter.cellSpeed = cellSpeedStart * factor[hgfScaleCount];
}

//Inhibitors of VEGF signaling // Inducing anglogenesis
var vegfScaleCount = parseInt(args[2+6]);
if (isNaN(vegfScaleCount) || vegfScaleCount<0 || vegfScaleCount>4) usage_exit();
var values = [1, 0.75, 0.5, 0.25, 0].reverse();
emitter.angio = values[vegfScaleCount];

//PARP inhibitors // Genome instability & mutation
var parpScaleCount = parseInt(args[2+7]);
if (isNaN(parpScaleCount) || parpScaleCount<0 || parpScaleCount>4) usage_exit();
if (entro.state && evo.inhib.indexOf(6)) emitter.immuneAvoid = immuneAvoidStart + 0.04;
else if (evo.state && evo.inhib[evo.inhib.length-1] == 6) emitter.immuneAvoid =
immuneAvoidStart + 0.04;
else {
    var offset = [0.04, 0.02, 0, -0.02, -0.05]; //.reverse();
    emitter.immuneAvoid = immuneAvoidStart + offset[parpScaleCount];
}

//Proapoptotic BH3 mimetics // Resisting cell death
var proapoptoticScaleCount = parseInt(args[2+8]);
if (isNaN(proapoptoticScaleCount) || proapoptoticScaleCount<0 || proapoptoticScaleCount>4)
usage_exit();
var values = [1, 0.75, 0.5, 0.25, 0].reverse();
emitter.proapoptotic = values[proapoptoticScaleCount];

//Aerobic glycolysis inhibitors // Reprogramming cellular energetics
var aerobicScaleCount = parseInt(args[2+9]);
if (isNaN(aerobicScaleCount) || aerobicScaleCount<0 || aerobicScaleCount>4) usage_exit();
var values = [1, 0.75, 0.5, 0.25, 0].reverse();
emitter.aerobic = values[aerobicScaleCount];

//EGFR inhibitors // Stimulating proliferative signaling
var egfrScaleCount = parseInt(args[2+10]);
if (isNaN(egfrScaleCount) || egfrScaleCount<0 || egfrScaleCount>4) usage_exit();
if (entro.state && evo.inhib.indexOf(9)) emitter.reproCountdown = reproCountdownStart / 3;
else if (evo.state && evo.inhib[evo.inhib.length-1] == 9) emitter.reproCountdown =
reproCountdownStart / 3;
else {
    var factor = [1/3.0, 1/1.5, 1.0, 1.5, 3].reverse();
    emitter.reproCountdown = reproCountdownStart * factor[egfrScaleCount];
}
```

```
function StageMock() {
    this.cells = [];
    this.getNumChildren = function () {
        return this.cells.length;
    };
    this.getChildAt = function(idx) {
        return this.cells[idx];
    };
    this.addChild = function(cell) {
        this.cells.push(cell);
    }
    this.removeChildAt = function(idx) {
        this.cells.splice(idx, 1);
    }
}

// function to create new cells (functions used for visualisation are inactive)
function createCell(parent) {
    var cell = {} //new createjs.Shape(); /*visualisation*/
    //cell.graphics.setStrokeStyle(emitter.cellSize *
0.4).beginStroke(emitter.cellStrokeColor).beginFill(emitter.cellColor).drawCircle(0, 0,
emitter.cellSize); /*visualisation*/

    // a new cell cannnot reproduce itselfe in the same tick it is created
    cell.canReproduce = false;

    // if the created cell is the initial cell it is placed in the middle of the petri dish
    if (parent == motherID) {
        cell.x = emitter.x;
        cell.y = emitter.y;
        cell.canReproduce = true;
        cell.hayflick = emitter.hayflick;
    }
    // all other cells are placed next to its mother cell in a random angle
    else {
        motherCell = stage.getChildAt(parent);
        cell.x = motherCell.x + Math.sin(Math.random() * 360) * emitter.cellSize * 2;
        cell.y = motherCell.y + Math.cos(Math.random() * 360) * emitter.cellSize * 2;
        cell.hayflick = motherCell.hayflick;
    };

    // cell get values from emitter
    cell.speedX = (Math.random() * (2 * emitter.cellSpeed)) - emitter.cellSpeed;
    cell.speedY = (Math.random() * (2 * emitter.cellSpeed)) - emitter.cellSpeed;
    cell.time = 0;
    cell.reproIn = emitter.reproCountdown;
    cell.hayflickReduction = emitter.hayflickReduction;
    cell.growthChance = emitter.growthChance;
    cell.immuneAvoid = emitter.immuneAvoid;
    cell.immuneCountdown = emitter.immuneCountdown;
    cell.parried = false;
    cell.immuneResistance = emitter.immuneResistance;
    cell.angio = emitter.angio;
    cell.proapoptotic = emitter.proapoptotic;
    cell.aerobic = emitter.aerobic;

    // cell object is added to array of all objects and output value "allCells" is counted up
by 1
    stage.addChild(cell);
    allCells ++;
}

// function to create a group of initial cells
function createCells(count) {
    for (var i = 0; i < count; i++) {
        var cell = {};
        cell.canReproduce = false;
        cell.x = emitter.x + Math.sin((360 / count) * i) * size * 0.05;
        cell.y = emitter.y + Math.cos((360 / count) * i) * size * 0.05;
        cell.hayflick = emitter.hayflick;
        cell.speedX = (Math.random() * (2 * emitter.cellSpeed)) - emitter.cellSpeed;
        cell.speedY = (Math.random() * (2 * emitter.cellSpeed)) - emitter.cellSpeed;
        cell.time = 0;
        cell.reproIn = emitter.reproCountdown;
        cell.hayflickReduction = emitter.hayflickReduction;
        cell.growthChance = emitter.growthChance;
```

```
        cell.immuneAvoid = emitter.immuneAvoid;
        cell.immuneCountdown = emitter.immuneCountdown;
        cell.parried = false;
        cell.immuneResistance = emitter.immuneResistance;
        cell.angio = emitter.angio;
        cell.proapoptotic = emitter.proapoptotic;
        cell.aerobic = emitter.aerobic;

        stage.addChild(cell);
        allCells ++;
    }
}

// function to calculate distance between 2 objects
function sqrLineDistance(point1, point2) {
    var xs = 0;
    var ys = 0;

    xs = point2.x - point1.x;
    xs = xs * xs;

    ys = point2.y - point1.y;
    ys = ys * ys;

    return (xs + ys);
}

stage = new StageMock();
motherID = 0;
//create initinial population of 10 cells;
createCells(10);

//**** RUN MAIN LOOP *****
for (var iteration=0; iteration<num_steps; iteration++) {

    var elapsed = 50;

    deadCellsTemp = 0;
    newCellsTemp = 0;

    // not in use
    evo.countdown --;
    if (evo.countdown <= 0) {
        evo.countdown = 50;
        evo.inhib.push (Math.floor(Math.random() * 10));
        if (evo.inhib.length > 3) {
            evo.inhib.shift();
        }
        //console.log(evo.inhib);
        //console.log(evo.inhib[evo.inhib.length-1]);
    }

    // values for tolerated overlapping of cells are set
    refDistOverlap = (emitter.cellSize * 1.2) * (emitter.cellSize * 1.2);
    refDistCircle = (resultCircleSize - emitter.cellSize) * (resultCircleSize -
emitter.cellSize)

    // the following happens for each cell
    for (var i=0; i < stage.getNumChildren(); i++) {
        var cell = stage.getChildAt(i);
        cell.growthChance = emitter.growthChance;

        // check if cell may reproduce
        if (cell.canReproduce == true && cell.reproIn <= 0) {
            if (Math.random() <= (cell.growthChance + (0.05 * emitter.inflammation))/* &&
stage.getNumChildren() < 1800*/) { //3000) {

                if (cell.hayflick>1 && cell.hayflick-cell.hayflickReduction<=1) {
                    hayflickReached++;
                }

                cell.hayflick -= cell.hayflickReduction;


                createCell(i);
                newCellsTemp ++;
            }
```

```
            cell.reproIn = emitter.reproCountdown;
        }

        deleted = false;
        for (var j=i+1; j < stage.getNumChildren(); j++) {
            var otherCell = stage.getChildAt(j);
            if (sqrLineDistance (cell, otherCell) < refDistOverlap) {
                stage.removeChildAt(i)
                //console.log("overlapping");
                overlappingCells ++;
                //deadCells ++;
                deadCellsTemp ++;
                i--;
                deleted = true;
                break;
            }
        }

        if (!deleted) {

            // check if cell is attacked by immune system
            if (cell.immuneCountdown == 0 && Math.random() > cell.immuneAvoid) {
                cell.parried = true;
                immuneAttacked ++;
            }

            // if cell is attacked by immune system the attack duration is reduced by 1
            if (cell.parried == true) {
                //cell.graphics.clear().setStrokeStyle(emitter.cellSize *
0.25).beginStroke(minusColor).beginFill(hgfColor).drawCircle(0, 0, emitter.cellSize);
/*visualisation*/
                cell.immuneResistance --;
            }
            else if (cell.immuneCountdown < 0) {
                cell.immuneCountdown = emitter.immuneCountdown;
            }

            // cell speed gets reduced
            cell.time += elapsed / 1000;
            var cellBreak = cell.time + 1;
            if (cellBreak > emitter.breakCap) {
                cellBreak = emitter.breakCap;
            };
            // cell is moved according to speed
            cell.x += cell.speedX / cellBreak;
            cell.y += cell.speedY / cellBreak;

            cell.canReproduce = true;
            cell.reproIn -= emitter.inflammation;
            cell.immuneCountdown --;

            cell.deathResistance = (cell.angio + cell.proapoptotic + cell.aerobic) / 3;

            // check if cell dies
            if (((cell.hayflick <= 0 || cell.immuneResistance <= 0) && Math.random() >=
cell.deathResistance) || sqrLineDistance(cell, emitter) > refDistCircle) {
                stage.removeChildAt(i);
                i--;
                deadCellsTemp ++;
            }
        }
    }

    // output values get updated
    deadCells += deadCellsTemp;
    currentCells = allCells - deadCells;

    if (newCellsTemp>0 || deadCellsTemp>0) {
        cellChange = (newCellsTemp / (newCellsTemp + deadCellsTemp));
    } else {
        cellChange = 0;
    }

    currentCellsMax = Math.max(currentCellsMax, currentCells);
    currentCellsMin = Math.min(currentCellsMin, currentCells);
    deadCellsMax = Math.max(deadCellsMax, deadCellsTemp);
    deadCellsMin = Math.min(deadCellsMin, deadCellsTemp);
```

```
    newCellsMax = Math.max(newCellsMax, newCellsTemp);
    newCellsMin = Math.min(newCellsMin, newCellsTemp);

    ccCollect += cellChange;
    ccCount ++;
    ccAverage = ccCollect / ccCount;

}

// output values are added to protocol
var outputFields = [allCells, newCellsMin, newCellsMax, currentCells, currentCellsMin,
currentCellsMax,
          deadCells, deadCellsMin, deadCellsMax, overlappingCells, hayflickReached,
immuneAttacked, ccAverage];

console.log(args.slice(2).join(",") + "," + outputFields.join(","));
```

Batch Commented

```
(Boris Lau, PSIORI GmbH)

var childProcess = require('child_process');
var async = require('async')

// number of simulation steps per experiment
num_iterations = 6000
num_params = 10
// number of parallel processes (this is only to increase processing speed and has no impact
on simulation)
num_processes = 6
// number of repetitions per experiment to avoid exeptional values
num_repetitions = 5

node_cmd = process.argv[0]

// we use an array called "combinations" to store all combinations of dossages we want to
simulate (each combination will be one experiment)
// we start with all parameters being normal (dossage 2, dossages 0 and 1 are for inhibition,
dosages 3 and 4 are for experssion)
combinations = [{}]

// dossage for changed values is set to 4 (maximum expression)
var change_value = 4

// get combinations of parameter indexes that should be changed (we create an array of all
combinations for 1, 2 and 3 chaged values (dossage 4))
for (var a=0; a<num_params; a++) {
    changes = {};
    changes[a] = change_value;
    combinations.push(changes)
    for (var b=a+1; b<num_params; b++) {
        changes = {};
        changes[a] = change_value; changes[b] = change_value;
        combinations.push(changes)
        for (var c=b+1; c<num_params; c++) {
            changes = {};
            changes[a] = change_value; changes[b] = change_value; changes[c] = change_value;
            combinations.push(changes)
        }
    }
}

// set default parameters to 2
default_params = []
for (var i=0; i<num_params; i++) {
    default_params.push(2)
}

// repeat combinations equal to "num_repetitions" times
repeated_combinations = []
for (var i=0; i<num_repetitions; i++) {
    repeated_combinations = repeated_combinations.concat(combinations);
}

// here we print the informatoin which is used to initialize an experiment in our protocol
console.log('num_steps,growth,immune,immortality,inflammation,metastatis,angiogenesis,instabil
ity,deathresistance,energetics,signaling,allCells,newCellsMin,newCellsMax,currentCells,current
CellsMin,currentCellsMax,deadCells,deadCellsMin,deadCellsMax,overlappingCells,hayflickReached,
```

```
immuneAttacked,ccAverage');

// execute the simulation parallelized (num_processes, only to increase processing speed, no
impact on simulation)
async.eachLimit(repeated_combinations, num_processes, function(c, callback) {
    // clone the default parameter array, and modify
    p = default_params.slice(0);
    for (var idx in c) {
        if (c.hasOwnProperty(idx)) {
            p[idx] = c[idx]
        }
    }
    cmd = node_cmd + " mitosis_science_simulation.js " + num_iterations + " " + p.join(" ");

    // callback for returning simulation process
    function res(error, stdout, stderr) {
        console.log(stdout.trim());
        callback(error); // callback for the eachLimit handler
    }
    // async call to exec, calls res on return
    childProcess.exec(cmd, res);
});
```

Full Code Hematopoiesis
(Oliver Worm, PSIORI GmbH)

Simulation Master

```
# @author Oliver Worm, PSIORI GmbH

# generate all the models and objects needed
from BloodStream import *
from BoneMarrow import *
from StatConfig import *

from Statistics import *
from MarrowCell import *

import time
import csv

class SimulationMaster():

    def __init__(self):

        # generate cell_stats
        self.cell_stats = StatConfig()

        blood_map = {"ery": 400000, "gran": 500, "throm": 20000}
        self.blood_stream = BloodStream(blood_map=blood_map, cell_stats=self.cell_stats)
        self.bone_marrow = BoneMarrow(cell_stats=self.cell_stats)
        self.bone_marrow.addCell(MarrowCell("cp_1", 70, 0, False))
        marrow_map = {"ery": 8000, "gran": 250, "throm": 40000} # 9000, 300, 35000
        self.bone_marrow.addTransmitters(marrow_map)
        self.statistics = Statistics(blood_stream=self.blood_stream,
bone_marrow=self.bone_marrow)



    def simulationStep(self, tick):
        time_series = {}
        current_time = time.time()
        # remove dead blood cells from the blood stream
        dead_blood_cells = self.blood_stream.checkDeadCells(tick)
        time_series["remove_blood_cells"] = time.time() - current_time
        current_time = time.time()
        # add transmitters of these dead cells to the bone marrow
        self.bone_marrow.addTransmitters(dead_blood_cells)
        time_series["add_transmitters"] = time.time() - current_time
        current_time = time.time()
        # assign free transmitters to marrow cells
        self.bone_marrow.assignFreeTransmitters()
        time_series["assign_transmitters"] = time.time() - current_time
        current_time = time.time()
        # split cells at get all cells that will be transmitted to the blood stream
```

```
        new_blood_cells = self.bone_marrow.splitCells(tick)
        time_series["split_cells"] = time.time() - current_time
        current_time = time.time()
        # add these cells to the blood stream
        self.blood_stream.addBloodCells(new_blood_cells, tick)
        time_series["add_blood_cells"] = time.time() - current_time
        current_time = time.time()
        # update statistics
        self.statistics.addBloodCellValue(tick)
        self.statistics.addTransmitterValue(tick)
        self.statistics.addMarrowCellValue(tick)
        time_series["update_statistics"] = time.time() - current_time
        self.statistics.addTimeValue(tick, time_series)


    def dataExport(self, filename):
        # export all information to a .csv for analysis
        filename += ".csv"

        blood_ery = self.statistics.getBloodLine("ery")
        blood_gran = self.statistics.getBloodLine("gran")
        blood_throm = self.statistics.getBloodLine("throm")

        total_cells = self.statistics.getMarrowCellLine("total")
        filled_cells = self.statistics.getMarrowCellLine("filled")

        cp_cells = self.statistics.getMarrowCellGroupLine("cp")
        ery_cells = self.statistics.getMarrowCellGroupLine("ery")
        gran_cells = self.statistics.getMarrowCellGroupLine("gran")
        throm_cells = self.statistics.getMarrowCellGroupLine("throm")

        all_cell_types = list(self.bone_marrow.getCellTypes())
        all_cell_types.sort()
        all_cell_lines = {}
        for cell_type in all_cell_types:
            all_cell_lines[cell_type] = self.statistics.getMarrowCellLine(cell_type)




        with open(filename, 'w') as new_file:
            wrtr = csv.writer(new_file, delimiter=';', quotechar='"')
            title_row = ['tick', 'blood_ery', 'blood_gran', 'blood_throm',
'total_marrow_cells', 'filled_marrow_cells', 'cp_group', 'ery_group', 'gran_group',
'throm_group']
            title_row.extend(all_cell_types)
            wrtr.writerow(title_row)
            for i in range(len(total_cells)):
                data_row = [i, blood_ery[i], blood_gran[i], blood_throm[i], total_cells[i],
filled_cells[i], cp_cells[i], ery_cells[i], gran_cells[i], throm_cells[i]]
                data_row.extend([all_cell_lines[cell_type][i] for cell_type in
all_cell_types])
                wrtr.writerow(data_row)
```

BloodStream

```
# @author Oliver Worm, PSIORI GmbH

# blood stream object, containing all relevant information about its cells

import random

class BloodStream():
    def __init__(self, blood_map, cell_stats):
        # map over each tick and the cells dying in it
        self.tick_cells = {}
        # total counter of present blood cells
        self.blood_cell_count = {}
        # stats of cells (life span)
        self.cell_stats = cell_stats

        # generating cells for the initial setup
        for cell_type, quantity in blood_map.items():
            self.blood_cell_count[cell_type] = quantity
            if not cell_type in self.cell_stats.getBloodCellStats():
                # if this cell type is unknown in the blood, this cell wil not die
                max_life_span = float('inf')
```

```
            else:
                # otherwise get the average life span from the stats
                max_life_span = self.cell_stats.getBloodCellStats()[cell_type].getLifeSpan()
            for c in range(quantity):
                # for all cells calculate when it will die and add it to the map
                death_tick = random.randint(0, max_life_span)
                if not death_tick in self.tick_cells:
                    self.tick_cells[death_tick] = {}
                if not cell_type in self.tick_cells[death_tick]:
                    self.tick_cells[death_tick][cell_type] = 0
                self.tick_cells[death_tick][cell_type] += 1

    # check which cells die in this tick
    def checkDeadCells(self, tick):

        if tick not in self.tick_cells:
            return {}
        death_hash = self.tick_cells[tick]
        # check the cell types in that map entry
        for cell_type in death_hash:
            if not cell_type in self.cell_stats.getBloodCellStats():
                del death_hash[cell_type]
            if cell_type in self.blood_cell_count:
                self.blood_cell_count[cell_type] -= death_hash[cell_type]
        # this map entry can be deleted as this tick will never appear again
        del self.tick_cells[tick]
        return death_hash

    # add new cells to the blood stream
    def addBloodCells(self, blood_map, tick):

        # works in the same way as init
        for cell_type, quantity in blood_map.items():

            if not cell_type in self.blood_cell_count:
                self.blood_cell_count[cell_type] = 0
            self.blood_cell_count[cell_type] += quantity

            if not cell_type in self.cell_stats.getBloodCellStats():
                death_tick = float('inf')
                if not death_tick in self.tick_cells:
                    self.tick_cells[death_tick] = {}
                if not cell_type in self.tick_cells[death_tick]:
                    self.tick_cells[death_tick][cell_type] = 0
                self.tick_cells[death_tick][cell_type] += quantity
            else:
                life_span = self.cell_stats.getBloodCellStats()[cell_type].getLifeSpan()
                for c in range(quantity):
                    death_tick = (int)(tick + random.gauss(life_span, life_span * 0.1))
                    if not death_tick in self.tick_cells:
                        self.tick_cells[death_tick] = {}
                    if not cell_type in self.tick_cells[death_tick]:
                        self.tick_cells[death_tick][cell_type] = 0
                    self.tick_cells[death_tick][cell_type] += 1

    # get the number of blood cells of a specific type
    def getBloodCellCount(self, cell_type):
        if not cell_type in self.blood_cell_count:
            return 0
        return self.blood_cell_count[cell_type]

    # check which cell types are currently present in the blood
    def getBloodCellTypes(self):
        return self.blood_cell_count.keys()
```

BoneMarrow

```
# @author Oliver Worm, PSIORI GmbH

import random
from MarrowCell import *

import time
class BoneMarrow:
```

```
    def __init__(self, cell_stats):
        self.marrow_cells = []
        self.total_transmitters = {}
        self.free_transmitters = {}
        self.marrow_cell_count = {}
        self.cell_stats = cell_stats
        # number of cells that have space ein the bone marrow
        self.marrow_space = 10000

    def addCell(self, new_cell):
        # add a new cell to the bone marrow
        cell_type = new_cell.getType()
        new_cell.setSplitStats(self.cell_stats.getMarrowCellStats(cell_type))
        self.marrow_cells.append(new_cell)
        # add it to the cell counters
        if not cell_type in self.marrow_cell_count:
            self.marrow_cell_count[cell_type] = 0
        self.marrow_cell_count[cell_type] += 1

    def addTransmitters(self, dead_cells):
        # add free transmitters to the marrow from dead blood cells
        for cell_type, quantity in dead_cells.items():
            trans_type = cell_type + "_trans"
            if not trans_type in self.total_transmitters:
                self.total_transmitters[trans_type] = 0
                self.free_transmitters[trans_type] = 0
            self.total_transmitters[trans_type] += quantity
            self.free_transmitters[trans_type] += quantity

    def assignFreeTransmitters(self):

        # first some cells may lose some transmitter
        for cell in self.marrow_cells:
            if random.random() > 0.95:
                for trans_type in cell.getStats().getApplicableTransmitters():
                    # should not happen, but if the free transmitter dict doesn't know this
type...
                    if trans_type not in self.free_transmitters:
                        self.free_transmitters[trans_type] = 0
                    self.free_transmitters[trans_type] += cell.removeTransmitter(trans_type)
        # which transmitters are present?
        applicable_transmitters = list(self.free_transmitters.keys())
        # as long as transmitters can still be applied
        while (len(applicable_transmitters) > 0):

            # choose one at random
            chosen_transmitter = applicable_transmitters[random.randint(0,
len(applicable_transmitters) - 1)]

            # do we even have transmitter left?
            if self.free_transmitters[chosen_transmitter] <= 0:
                applicable_transmitters.remove(chosen_transmitter)
                continue

            # see if that transmitter can be applied
            transmitter_applicable = False
            for cell in self.marrow_cells:
                if not cell.getFilled() and chosen_transmitter in
cell.getStats().getApplicableTransmitters():
                    transmitter_applicable = True
                    break
            # no cell present that this transmitter can bind to? remove it and continue
            if not transmitter_applicable:
                applicable_transmitters.remove(chosen_transmitter)
                continue
            # check all cells, the ones not already filled are mapped according to their
filling state and total space
            orig_distribution_map = []
            index_map = []
            for index, cell in enumerate(self.marrow_cells):
                if cell.getFilled():
                    continue
                # make it inverse so cells further down the road have a higher likelihood of
receiving transmitter
                transmitter_influence =
cell.getStats().getTransmitterSpace(chosen_transmitter)
                if transmitter_influence > 0.0:
```

```
                    # insert the new value
                    orig_distribution_map.append(1.0 / transmitter_influence)
                    index_map.append(index)


            # choose x random cells from that distribution_map, shortening the total
iterations needed
            chosen_cells = []
            cell_loop_count = 1
            if len(orig_distribution_map) > 1:
                cell_loop_count = random.randint(1, len(orig_distribution_map) - 1)

            for x in range(cell_loop_count):
                # break if none of that transmitter is left
                if self.free_transmitters[chosen_transmitter] <= 0:
                    break

                distribution_map = orig_distribution_map.copy()

                # normalize distribution map and sum up
                distribution_sum = sum(orig_distribution_map)
                distribution_map[0] = float(orig_distribution_map[0] / distribution_sum)
                for i in range(1, len(distribution_map)):
                    distribution_map[i] = distribution_map[i - 1] +
float(orig_distribution_map[i] / distribution_sum)

                random_dist = random.random()
                # find the cell we just chose through the index map
                chosen_index = 0
                while distribution_map[chosen_index] < random_dist:
                    chosen_index += 1
                # multiple cells are taken, we have to make sure no doubles occur
                # simply go to the next cell, loop at the end
                while chosen_index in chosen_cells:
                    chosen_index += 1
                    if chosen_index >= len(distribution_map):
                        chosen_index = 0

                # get the cell chosen by probability
                chosen_cell = self.marrow_cells[index_map[chosen_index]]
                # how much of the transmitter can this cell actually take?
                free_transmitter_space =
chosen_cell.getFreeTransmitterSpace(chosen_transmitter)

                # can we attach it all? or only some of it?
                bind_transmitter = min(self.free_transmitters[chosen_transmitter],
free_transmitter_space)
                chosen_cell.attachTransmitter(chosen_transmitter, bind_transmitter)
                # remove the attached transmitter from the quantity of free transmitter
                self.free_transmitters[chosen_transmitter] -= bind_transmitter

                del orig_distribution_map[chosen_index]
                del index_map[chosen_index]
            # we are done with this transmitter for this tick!
            applicable_transmitters.remove(chosen_transmitter)

    def splitCells(self, tick):
        # keep track of all cells that were split in this method call
        split_cells = []
        blood_stream_cells = {}
        # go through all cells in the marrow
        for cell in self.marrow_cells:
            # if this cell is dorment or needs transmitter and is not yet filled
            if cell.getActivationTick() > tick or (cell.getStats().getTransDep() and not
cell.getFilled()):
                continue

            # is there even space? if not, only cancer cells can split
            if len(self.marrow_cells) >= self.marrow_space and not cell.getCancerous():
                continue

            # cancer cells can't split all the time
            if cell.getCancerous() and random.random() < 0.95:
                continue

            # this cell can be split and can later be deleted
            split_cells.append(cell)
```

```
            # check if this cell even has enough telomere to split. otherwise delete.
            if cell.getTelomereLength() - cell.getStats().getSplitTelomereLoss() <= 0:
                for trans_type in cell.getStats().getApplicableTransmitters():
                    # should not happen, but if the free transmitter dict doesn't know this
type...
                    if trans_type not in self.free_transmitters:
                        self.free_transmitters[trans_type] = 0
                    self.free_transmitters[trans_type] += cell.removeTransmitter(trans_type)
                continue
            # what will this cell differentiate into?
            split_result = cell.getSplitResult()

            # is this an end-product?
            if split_result == "ery":
                if not split_result in blood_stream_cells:
                    blood_stream_cells[split_result] = 0
                blood_stream_cells[split_result] += 2
                for trans_type in cell.getStats().getApplicableTransmitters():
                    if trans_type not in self.total_transmitters:
                        self.total_transmitters[trans_type] = 0
                    self.total_transmitters[trans_type] -=
cell.getStats().getTransmitterAttached(trans_type)
                continue
            elif split_result == "gran":
                if not split_result in blood_stream_cells:
                    blood_stream_cells[split_result] = 0
                blood_stream_cells[split_result] += 2
                for trans_type in cell.getStats().getApplicableTransmitters():
                    if trans_type not in self.total_transmitters:
                        self.total_transmitters[trans_type] = 0
                    self.total_transmitters[trans_type] -=
cell.getStats().getTransmitterAttached(trans_type)
                continue
            elif split_result == "throm":
                if not split_result in blood_stream_cells:
                    blood_stream_cells[split_result] = 0
                blood_stream_cells[split_result] += 20
                for trans_type in cell.getStats().getApplicableTransmitters():
                    if trans_type not in self.total_transmitters:
                        self.total_transmitters[trans_type] = 0
                    self.total_transmitters[trans_type] -=
cell.getStats().getTransmitterAttached(trans_type)
                continue

            # return the bound transmitter to the marrow if the cell was not end-product
            for trans_type in cell.getStats().getApplicableTransmitters():
                # should not happen, but if the free transmitter dict doesn't know this
type...
                if trans_type not in self.free_transmitters:
                    self.free_transmitters[trans_type] = 0
                self.free_transmitters[trans_type] += cell.removeTransmitter(trans_type)

            # create first new cell
            new_telomere_length = cell.getTelomereLength() -
cell.getStats().getSplitTelomereLoss()
            dormency = self.cell_stats.getMarrowCellStats(split_result).getSplitDormency()
            new_activation_tick = int(tick + random.gauss(dormency, dormency * 0.1))
            new_cancerous = self.cell_stats.getMarrowCellStats(split_result).getCancerous()
            new_cell_1 = MarrowCell(split_result, new_telomere_length, new_activation_tick,
new_cancerous)
            self.addCell(new_cell_1)
            # create second new cell depending in whether the cell is self-replicating
            if cell.getStats().getSelfReplicating():
                new_telomere_length = cell.getTelomereLength() -
cell.getStats().getSplitTelomereLoss()
                dormency =
self.cell_stats.getMarrowCellStats(cell.getType()).getSplitDormency()
                new_activation_tick = int(tick + random.gauss(dormency, dormency * 0.1))
                new_cancerous =
self.cell_stats.getMarrowCellStats(cell.getType()).getCancerous()
                new_cell_2 = MarrowCell(cell.getType(), new_telomere_length,
new_activation_tick, new_cancerous)
                self.addCell(new_cell_2)
            else:
                new_cancerous =
self.cell_stats.getMarrowCellStats(split_result).getCancerous()
```

```
                new_telomere_length = cell.getTelomereLength() -
cell.getStats().getSplitTelomereLoss()
                dormency = self.cell_stats.getMarrowCellStats(split_result).getSplitDormency()
                new_activation_tick = int(tick + random.gauss(dormency, dormency * 0.1))
                new_cell_2 = MarrowCell(split_result, new_telomere_length,
new_activation_tick, new_cancerous)
                self.addCell(new_cell_2)
        # remove all cells that were split!
        for cell in split_cells:
            self.marrow_cells.remove(cell)
            self.marrow_cell_count[cell.getType()] -= 1

        # remove random cells as long as we are over the limit
        while(len(self.marrow_cells) > self.marrow_space):
            cell = self.marrow_cells[random.randint(0, len(self.marrow_cells) - 1)]
            if cell.getStats().getAnkered():
                # if this cell is ankered it can not be removed
                continue
            # return the bound transmitter to the marrow if the cell was not end-product
            for trans_type in cell.getStats().getApplicableTransmitters():
                # should not happen, but if the free transmitter dict doesn't know this
type...
                if trans_type not in self.free_transmitters:
                    self.free_transmitters[trans_type] = 0
                self.free_transmitters[trans_type] += cell.removeTransmitter(trans_type)
            self.marrow_cells.remove(cell)
            self.marrow_cell_count[cell.getType()] -= 1

        # return the cells that go into the blood stream
        return blood_stream_cells


    def getTotalTransmitterStrength(self, trans_type):
        if not trans_type in self.total_transmitters:
            return 0
        return self.total_transmitters[trans_type]

    def getFreeTransmitterStrength(self, trans_type):
        if not trans_type in self.free_transmitters:
            return 0
        return self.free_transmitters[trans_type]

    def getTransmitterTypes(self):
        return self.total_transmitters.keys()

    def getTotalCellCount(self):
        return len(self.marrow_cells)

    def getFilledCellCount(self):
        count = 0
        for c in self.marrow_cells:
            if c.getFilled():
                count += 1
        return count

    def getAverageTelomereLength(self):
        if len(self.marrow_cells) == 0:
            return 0
        total = 0
        for c in self.marrow_cells:
            total += c.getTelomereLength()
        return float(total / len(self.marrow_cells))

    def getCellTypes(self):
        return self.marrow_cell_count.keys()

    def getCellCount(self, cell_type):
        if not cell_type in self.marrow_cell_count:
            return 0
        return self.marrow_cell_count[cell_type]

    def getCell(self, index):
        return self.marrow_cells[index]

    def printCellInformation(self):
        print("INFORMATION:")
        for c in self.marrow_cells:
```

```
            c.printCellInformation()
            print("----")
```

MarrowCell

```
# @author Oliver Worm, PSIORI GmbH

from copy import deepcopy
from math import ceil
class MarrowCell:

    def __init__(self, cell_type, telomere_length, activation_tick, cancerous):
        self.cell_type = cell_type
        self.telomere_length = telomere_length
        self.activation_tick = activation_tick
        self.split_stats = None
        self.filled = False
        self.cancerous = cancerous

    def setSplitStats(self, split_stats):
        self.split_stats = deepcopy(split_stats)

    def attachTransmitter(self, trans_type, trans_quantity):
        self.filled = self.split_stats.attachTransmitter(trans_type, trans_quantity)

    def removeTransmitter(self, trans_type):
        self.filled = False
        return self.split_stats.removeTransmitter(trans_type)

    def getType(self):
        return self.cell_type

    def getSplitResult(self):
        return self.split_stats.getSplitResult()

    def getStats(self):
        return self.split_stats

    def getActivationTick(self):
        return self.activation_tick

    def getFreeTransmitterSpace(self, trans_type):
        # catch the case that this cell can't accept this transmitter
        if not trans_type in self.split_stats.getApplicableTransmitters():
            return 0
        # how filled is this cell already?
        transmitter_percentage = self.split_stats.getFilledPercentage()
        # how much of a certain transmitter can be accepted given that percentage?
        free_space = ceil((1.0 - transmitter_percentage) *
self.split_stats.getTransmitterSpace(trans_type))
        return free_space

    def getFilled(self):
        return self.filled

    def getCancerous(self):
        return self.cancerous

    def getTelomereLength(self):
        return self.telomere_length

    def printCellInformation(self):
        print("Cell type:", self.cell_type)
        print("Telomere length:", self.telomere_length)
```

MarrowCellStats

```
# @author Oliver Worm, PSIORI GmbH

from random import randint

class MarrowCellStats:

    def __init__(self, cell_type, split_telomere_loss, split_dormency, transmitter_space,
split_results, self_rep, cancerous, trans_dependent, is_ankered):
```

```
        self.cell_type = cell_type
        self.split_telomere_loss = split_telomere_loss
        self.transmitter_space = transmitter_space
        self.transmitter_attached = {}
        self.split_results = split_results
        self.split_dormency = split_dormency
        self.self_rep = self_rep
        self.filled_percentage = 0.0
        self.cancerous = cancerous
        self.trans_dependent = trans_dependent
        self.is_ankered = is_ankered

    def attachTransmitter(self, trans_type, trans_quantity):
        if not trans_type in self.transmitter_space:
            return self.filled_percentage >= 1.0
        # make space if necessary
        if not trans_type in self.transmitter_attached:
            self.transmitter_attached[trans_type] = 0
        # attach transmitter
        self.transmitter_attached[trans_type] += trans_quantity
        self.filled_percentage += float(trans_quantity / self.transmitter_space[trans_type])
        return self.filled_percentage >= 1.0

    def removeTransmitter(self, trans_type):
        # remove all transmitter of that type from the cell
        if not trans_type in self.transmitter_attached:
            return 0
        # reset the filled percentage of that cell to the state without that transmitter
        self.filled_percentage -= float(self.transmitter_attached[trans_type] /
self.transmitter_space[trans_type])
        return self.transmitter_attached.pop(trans_type)

    def getSplitResult(self):
        max_trans = 0
        # assign a random transmitter to catch cancer cells without transmitter
        best_result = list(self.transmitter_space.keys())[randint(0,
len(self.transmitter_space.keys()) - 1)]
        # check which transmitter is (percentage-wise) strongest in this cell
        for trans_type in self.transmitter_attached:
            trans_strength = float(self.transmitter_attached[trans_type] /
self.transmitter_space[trans_type])
            if trans_strength > max_trans:
                max_trans = trans_strength
                best_result = trans_type
        return self.split_results[best_result]


    def getTransmitterSpace(self, trans_type):
        # how much of this transmitter can attach to this cell?
        if not trans_type in self.transmitter_space:
            return 0
        return self.transmitter_space[trans_type]

    def getTransmitterAttached(self, trans_type):
        if not trans_type in self.transmitter_attached:
            return 0
        return self.transmitter_attached[trans_type]

    def getApplicableTransmitters(self):
        return self.transmitter_space.keys()

    def getFilledPercentage(self):
        return self.filled_percentage

    def getSplitTelomereLoss(self):
        return self.split_telomere_loss

    def getSplitDormency(self):
        return self.split_dormency

    def getSelfReplicating(self):
        return self.self_rep

    def getCancerous(self):
        return self.cancerous

    def getTransDep(self):
```

```
        return self.trans_dependent

    def getAnkered(self):
        return self.is_ankered
```

BloodCellStats

```
# @author Oliver Worm, PSIORI GmbH

class BloodCellStats:

    def __init__(self, cell_type, life_span):
        self.cell_type = cell_type
        self.life_span = life_span

    def getLifeSpan(self):
        return self.life_span
```

StatConfig

```
# @author Oliver Worm, PSIORI GmbH

from BloodCellStats import *
from MarrowCellStats import *

from itertools import combinations
from copy import deepcopy
class StatConfig():

    def __init__(self):
        # generate healthy blood cell stats
        self.blood_cell_stats = {}
        for cell_type in all_blood_cell_types():
            cell_stats = BloodCellStats(cell_type, blood_cell_life_span(cell_type))
            self.blood_cell_stats[cell_type] = cell_stats
        # generate healthy marrow cell stats
        self.marrow_cell_stats = {}
        for cell_type in all_marrow_cell_types():
            cell_stats = MarrowCellStats(cell_type, marrow_cell_telomere_loss(cell_type),
                                         marrow_cell_split_dormency(cell_type),
                                         marrow_cell_transmitter_space(cell_type),
                                         marrow_cell_split_results(cell_type),
                                         marrow_cell_self_rep(cell_type),
                                         marrow_cell_cancerous(cell_type),
                                         marrow_cell_trans_dependent(cell_type),
                                         marrow_cell_ankered(cell_type))
            self.marrow_cell_stats[cell_type] = cell_stats

        # generate list of all possible combinations of defects
        defects = ["diff", "tel", "tra"]
        defect_combinations = []

        final_stages = ["ery_5", "gran_5", "throm_5"]

        cancer_cell_stats = {}
        for i in range(len(defects)):
            defect_combinations.extend(list(combinations(defects, i + 1)))
        for df in defect_combinations:
            print(list(df))
            for mc in self.marrow_cell_stats:
                new_stats = deepcopy(self.marrow_cell_stats[mc])
                final_stage = new_stats.cell_type in final_stages
                # append mutations to cell name
                name_app = ""
                for mut in df:
                    name_app += "_" + mut
                new_stats.cell_type = new_stats.cell_type + name_app
                new_stats.cancerous = True
                # diff prevents further differentiation = same level!
                if "diff" in df:
                    for r in new_stats.split_results:
                        new_stats.split_results[r] = new_stats.cell_type
                # no diff? then cancer cells of the next stage with same mutations are
produced
                elif not final_stage:
```

```
                    for r in new_stats.split_results:
                        new_stats.split_results[r] = new_stats.split_results[r] + name_app
                if "tel" in df:
                    # no telomere loss
                    new_stats.split_telomere_loss = 0
                if "tra" in df:
                    # cells will reproduce slower but without transmitter
                    new_stats.split_dormency = 20
                    new_stats.trans_dependent = False
                cancer_cell_stats[new_stats.cell_type] = new_stats

        # append the cancer types to the stats array
        self.marrow_cell_stats.update(cancer_cell_stats)

    # get these statistics
    def getBloodCellStats(self):
        return self.blood_cell_stats

    # get only marrow cell relevant statistics
    def getAllMarrowCellStats(self):
        return self.marrow_cell_stats

    # get cell stats of a specific cell type
    def getMarrowCellStats(self, cell_type):
        return self.marrow_cell_stats[cell_type]

# what kind of cells are present in normal bone marrow
def all_marrow_cell_types():
    return ["cp_1", "cp_2", "cp_3", "cp_4", "cp_5",
            "ery_1", "ery_2", "ery_3", "ery_4", "ery_5",
            "gran_1", "gran_2", "gran_3", "gran_4", "gran_5",
            "throm_1", "throm_2", "throm_3", "throm_4", "throm_5"]

# how much telomere do they lose in a split
def marrow_cell_telomere_loss(cell_type):
    general_loss = 1
    telomere_loss = {"cp_1": 0,
                     "cp_2": general_loss,
                     "cp_3": general_loss,
                     "cp_4": general_loss,
                     "cp_5": general_loss,
                     "ery_1": general_loss,
                     "ery_2": general_loss,
                     "ery_3": general_loss,
                     "ery_4": general_loss,
                     "ery_5": general_loss,
                     "gran_1": general_loss,
                     "gran_2": general_loss,
                     "gran_3": general_loss,
                     "gran_4": general_loss,
                     "gran_5": general_loss,
                     "throm_1": general_loss,
                     "throm_2": general_loss,
                     "throm_3": general_loss,
                     "throm_4": general_loss,
                     "throm_5": general_loss}
    if not cell_type in telomere_loss:
        return general_loss
    return telomere_loss[cell_type]

# how long do they have to wait after splitting
def marrow_cell_split_dormency(cell_type):
    general_dormency = 5
    cancer_dormency = 50
    split_dormency = {"cp_1": general_dormency,
                      "cp_2": general_dormency,
                      "cp_3": general_dormency,
                      "cp_4": general_dormency,
                      "cp_5": general_dormency,
                      "ery_1": general_dormency,
                      "ery_2": general_dormency,
                      "ery_3": general_dormency,
                      "ery_4": general_dormency,
                      "ery_5": general_dormency,
                      "gran_1": general_dormency,
                      "gran_2": general_dormency,
                      "gran_3": general_dormency,
```

```
                      "gran_4": general_dormency,
                      "gran_5": general_dormency,
                      "throm_1": general_dormency,
                      "throm_2": general_dormency,
                      "throm_3": general_dormency,
                      "throm_4": general_dormency,
                      "throm_5": general_dormency}
    if not cell_type in split_dormency:
        return general_dormency
    return split_dormency[cell_type]

# what transmitter types and how much do they need
def marrow_cell_transmitter_space(cell_type):
    transmitter_space = {"cp_1": {"ery_trans": 1024, "gran_trans": 1024, "throm_trans":
10240},
                         "cp_2": {"ery_trans": 512, "gran_trans": 512, "throm_trans": 5120},
                         "cp_3": {"ery_trans": 256, "gran_trans": 256, "throm_trans": 2560},
                         "cp_4": {"ery_trans": 128, "gran_trans": 128, "throm_trans": 1280},
                         "cp_5": {"ery_trans": 64, "gran_trans": 64, "throm_trans": 640},
                         "ery_1": {"ery_trans": 32},
                         "ery_2": {"ery_trans": 16},
                         "ery_3": {"ery_trans": 8},
                         "ery_4": {"ery_trans": 4},
                         "ery_5": {"ery_trans": 2},
                         "gran_1": {"gran_trans": 32},
                         "gran_2": {"gran_trans": 16},
                         "gran_3": {"gran_trans": 8},
                         "gran_4": {"gran_trans": 4},
                         "gran_5": {"gran_trans": 2},
                         "throm_1": {"throm_trans": 320},
                         "throm_2": {"throm_trans": 160},
                         "throm_3": {"throm_trans": 80},
                         "throm_4": {"throm_trans": 40},
                         "throm_5": {"throm_trans": 20}}
    if not cell_type in transmitter_space:
        return {}
    return transmitter_space[cell_type]

# what kind of cells are produced in a split
def marrow_cell_split_results(cell_type):
    split_results = {"cp_1": {"ery_trans": "cp_2", "gran_trans": "cp_2", "throm_trans":
"cp_2"},
                     "cp_2": {"ery_trans": "cp_3", "gran_trans": "cp_3", "throm_trans":
"cp_3"},
                     "cp_3": {"ery_trans": "cp_4", "gran_trans": "cp_4", "throm_trans":
"cp_4"},
                     "cp_4": {"ery_trans": "cp_5", "gran_trans": "cp_5", "throm_trans":
"cp_5"},
                     "cp_5": {"ery_trans": "ery_1", "gran_trans": "gran_1", "throm_trans":
"throm_1"},
                     "ery_1": {"ery_trans": "ery_2"},
                     "ery_2": {"ery_trans": "ery_3"},
                     "ery_3": {"ery_trans": "ery_4"},
                     "ery_4": {"ery_trans": "ery_5"},
                     "ery_5": {"ery_trans": "ery"},
                     "gran_1": {"gran_trans": "gran_2"},
                     "gran_2": {"gran_trans": "gran_3"},
                     "gran_3": {"gran_trans": "gran_4"},
                     "gran_4": {"gran_trans": "gran_5"},
                     "gran_5": {"gran_trans": "gran"},
                     "throm_1": {"throm_trans": "throm_2"},
                     "throm_2": {"throm_trans": "throm_3"},
                     "throm_3": {"throm_trans": "throm_4"},
                     "throm_4": {"throm_trans": "throm_5"},
                     "throm_5": {"throm_trans": "throm"}}
    if not cell_type in split_results:
        return {}
    return split_results[cell_type]

# does this cell reproduce itself or do both children differentiate
def marrow_cell_self_rep(cell_type):
    self_rep = {"cp_1": True,
                "cp_2": True,
                "cp_3": False,
                "cp_4": False,
                "cp_5": True,
                "ery_1": False,
```

```
                "ery_2": False,
                "ery_3": False,
                "ery_4": False,
                "ery_5": False,
                "gran_1": False,
                "gran_2": False,
                "gran_3": False,
                "gran_4": False,
                "gran_5": False,
                "throm_1": False,
                "throm_2": False,
                "throm_3": False,
                "throm_4": False,
                "throm_5": False}
    if not cell_type in self_rep:
        return True
    return self_rep[cell_type]

# is this cell cancerous
def marrow_cell_cancerous(cell_type):
    cancerous = {"cp_1": False,
                "cp_2": False,
                "cp_3": False,
                "cp_4": False,
                "cp_5": False,
                "ery_1": False,
                "ery_2": False,
                "ery_3": False,
                "ery_4": False,
                "ery_5": False,
                "gran_1": False,
                "gran_2": False,
                "gran_3": False,
                "gran_4": False,
                "gran_5": False,
                "throm_1": False,
                "throm_2": False,
                "throm_3": False,
                "throm_4": False,
                "throm_5": False}
    if not cell_type in cancerous:
        return False
    return cancerous[cell_type]

# does this cell need transmitters for proliferation
def marrow_cell_trans_dependent(cell_type):
    dependent = {"cp_1": True,
                 "cp_2": True,
                 "cp_3": True,
                 "cp_4": True,
                 "cp_5": True,
                 "ery_1": True,
                 "ery_2": True,
                 "ery_3": True,
                 "ery_4": True,
                 "ery_5": True,
                 "gran_1": True,
                 "gran_2": True,
                 "gran_3": True,
                 "gran_4": True,
                 "gran_5": True,
                 "throm_1": True,
                 "throm_2": True,
                 "throm_3": True,
                 "throm_4": True,
                 "throm_5": True}

    if not cell_type in dependent:
        return True
    return dependent[cell_type]

# make this cell ankered in the marrow no matter what happens
def marrow_cell_ankered(cell_type):
    ankered = {"cp_1": True,
                "cp_2": False,
                "cp_3": False,
                "cp_4": False,
```

```
                "cp_5": False,
                "ery_1": False,
                "ery_2": False,
                "ery_3": False,
                "ery_4": False,
                "ery_5": False,
                "gran_1": False,
                "gran_2": False,
                "gran_3": False,
                "gran_4": False,
                "gran_5": False,
                "throm_1": False,
                "throm_2": False,
                "throm_3": False,
                "throm_4": False,
                "throm_5": False}
    if not cell_type in ankered:
        return False
    return ankered[cell_type]

# what kind of blood cells do we have
def all_blood_cell_types():
    return ["ery", "gran", "throm"]

# how long do blood cells live
def blood_cell_life_span(cell_type):
    life_span = {"ery": 1200, "gran": 100, "throm": 10}
    # if the requested cell type is not standard, we assume it to be immortal
    if not cell_type in life_span:
        return float('inf')
    return life_span[cell_type]
```

Statistics

```
# @author Oliver Worm, PSIORI GmbH

class Statistics:

    def __init__(self, blood_stream, bone_marrow):
        self.blood_cell_series = {}
        self.marrow_cell_series = {"total": [], "filled": [], "average_telomere_length": []}
        self.total_transmitter_series = {}
        self.free_transmitter_series = {}
        self.blood_stream = blood_stream
        self.bone_marrow = bone_marrow
        self.time_stats = {}

    def addBloodCellValue(self, tick):
        for cell_type in self.blood_stream.getBloodCellTypes():
            if not cell_type in self.blood_cell_series:
                self.blood_cell_series[cell_type] = [0 for x in range(tick)]

self.blood_cell_series[cell_type].append(self.blood_stream.getBloodCellCount(cell_type))

    def addTransmitterValue(self, tick):
        for trans_type in self.bone_marrow.getTransmitterTypes():
            if not trans_type in self.total_transmitter_series:
                self.total_transmitter_series[trans_type] = [0 for x in range(tick)]
                self.free_transmitter_series[trans_type] = [0 for x in range(tick)]

self.total_transmitter_series[trans_type].append(self.bone_marrow.getTotalTransmitterStrength(
trans_type))

self.free_transmitter_series[trans_type].append(self.bone_marrow.getFreeTransmitterStrength(tr
ans_type))

    def addMarrowCellValue(self, tick):
        self.marrow_cell_series["total"].append(self.bone_marrow.getTotalCellCount())
        self.marrow_cell_series["filled"].append(self.bone_marrow.getFilledCellCount())

self.marrow_cell_series["average_telomere_length"].append(self.bone_marrow.getAverageTelomereL
ength())
```

```
        for cell_type in self.bone_marrow.getCellTypes():
            if not cell_type in self.marrow_cell_series:
                self.marrow_cell_series[cell_type] = [0 for x in range(tick)]

self.marrow_cell_series[cell_type].append(self.bone_marrow.getCellCount(cell_type))

    def addTimeValue(self, tick, steps):
        for method, used_time in steps.items():
            if not method in self.time_stats:
                self.time_stats[method] = [0 for x in range(tick)]
            self.time_stats[method].append(used_time)


    def getLastBloodValue(self, cell_type):
        if not cell_type in self.blood_cell_series:
            return 0
        return self.blood_cell_series[cell_type][-1]

    def getLastTotalTransmitterValue(self, trans_type):
        if not trans_type in self.total_transmitter_series:
            return 0
        return self.total_transmitter_series[trans_type][-1]

    def getLastFreeTransmitterValue(self, trans_type):
        if not trans_type in self.free_transmitter_series:
            return 0
        return self.free_transmitter_series[trans_type][-1]

    def getLastTotalMarrowCellValue(self):
        return self.marrow_cell_series["total"][-1]

    def getLastFilledMarrowCellValue(self):
        return self.marrow_cell_series["filled"][-1]

    def getLastAverageTelomereLength(self):
        return self.marrow_cell_series["average_telomere_length"][-1]

    def getLastTimeValue(self, method):
        if not method in self.time_stats:
            return 0
        return self.time_stats[method][-1]

    def getMarrowCellLine(self, type):
        if not type in self.marrow_cell_series:
            return []
        return self.marrow_cell_series[type]

    def getBloodLine(self, type):
        if not type in self.blood_cell_series:
            return []
        return self.blood_cell_series[type]

    def getMarrowCellGroupLine(self, type):

        cell_types = [type + "_" + str(i) for i in range(1, 6)]
        random_cell_type = list(self.bone_marrow.getCellTypes())[0]
        combined_values = [0 for x in range(len(self.getMarrowCellLine(random_cell_type)))]
        for cell_type in cell_types:
            cell_line = self.getMarrowCellLine(cell_type)
            for i in range(len(cell_line)):
                combined_values[i] += cell_line[i]
        return combined_values
```

12.2 Ridge Regression – Standard Deviations

	growth	immune	immortality	inflammation	metastatis	angiogenesis	instability	deathresistance	energetics	signaling	num_steps	allCells	newCells Max	currentCells	currentCells Min	currentCells Max	deadCells	deadCells Max	overlapping Cells	hayflickReached	immuneAttacked	ccAverage
0	2	2	2	2	2	2	2	2	2	2	0	0,063073667	0,020840209	0,656254016	0	0,015398155	0,067031167	0,029151322	0,063034307	0,174606382	0,071786064	0,001884849
1	2	2	2	2	2	2	2	2	2	4	0	0,063605547	0,011390026	0,546542436	0	0,011745687	0,06342562	0,018624281	0,06778434	0,036587551	0,07412034	0,000735437
2	2	2	2	2	2	2	2	2	4	2	0	0,027794401	0,026502585	0,507086128	0	0,020916084	0,027359673	0,023806322	0,032946832	0,163322034	0,02891397	0,002548695

3	2	2	2	2	2	2	2	2	4	4	0	0,045052782	0,017003406	0,649999587	0	0,01857143 6	0,045349253	0,0328066 9	0,05199274 1	0,04482733 7	0,035965528	0,001473165
4	2	2	2	2	2	2	2	4	2	2	0	0,037613909	0,009835402	0,058253968	0	0,01984247 7	0,037649 77	0,0121043 66	0,0405416	0,08355973 4	0,038963711	0,003130555
5	2	2	2	2	2	2	2	4	2	4	0	0,039952 88	0,030269809	0,505513131	0	0,02585418 4	0,039234344	0,0355120 2	0,04418553 7	0,05478434 9	0,053664726	0,000923822
6	2	2	2	2	2	2	2	4	4	2	0	0,081449784	0,030654566	0,550935883	0	0,02707992 8	0,079611682	0,0467596 54	0,08895481 8	0,09346209 6	0,083523553	0,002523295
7	2	2	2	2	2	2	2	4	4	4	0	0,078413186	0,007319371	0,298856599	0	0,00959311 1	0,078470443	0,0232740 88	0,08239524 4	0,06051381 9	0,071800045	0,001834802
8	2	2	2	2	2	2	4	2	2	2	0	0,009096502	0,011327393	0,376749 96	0	0,00789060 9	0,012256353	0,0465525 99	0,01049306 1	0,19964538 4	0,007484042	0,002033586
9	2	2	2	2	2	2	4	2	2	4	0	0,054410867	0,006996833	0,523988546	0	0,01142490 9	0,054264417	0,0475402	0,05799327 8	0,04788729 5	0,053262137	0,001535527
10	2	2	2	2	2	2	4	2	4	2	0	0,032055 69	0,012765053	0,576479898	0	0,02047389 1	0,033748121	0,0266198 57	0,03125662 3	0,11337490 8	0,03825885	0,003630912
11	2	2	2	2	2	2	4	2	4	4	0	0,061198127	0,017465731	0,547114469	0	0,00607451 8	0,060540546	0,0270669 39	0,06895229 5	0,03221389 3	0,065522475	0,001197529
12	2	2	2	2	2	2	4	4	2	2	0	0,061860414	0,014820477	0,467625 85	0	0,01612321 3	0,064640468	0,0168622 72	0,06389949 7	0,12431362 8	0,059195685	0,004804304
13	2	2	2	2	2	2	4	4	2	4	0	0,032836545	0,027380839	0,432123154	0	0,01493380 1	0,032723 69	0,0208056 78	0,03468078 6	0,02741376 1	0,039116878	0,000734 24
14	2	2	2	2	2	2	4	4	4	2	0	0,038841279	0,029859273	0,371001 55	0	0,01364441 7	0,039060817	0,0242044 44	0,04117050 4	0,17807270 5	0,045218376	0,003724673
15	2	2	2	2	2	4	2	2	2	2	0	0,038562939	0,021356353	0,410480957	0	0,01771451 6	0,040676525	0,0340975 17	0,03607865 8	0,24321398 9	0,03586515	0,004769444
16	2	2	2	2	2	4	2	2	2	4	0	0,060365918	0,014910225	0,520040197	0	0,00716996	0,060626957	0,0226694 13	0,06621906	0,05494437 3	0,064221225	0,001279671
17	2	2	2	2	2	4	2	2	4	2	0	0,050591418	0,021367476	0,416604932	0	0,01075926	0,050669656	0,0387365 19	0,05275615 7	0,04193095 4	0,059353454	0,001885 99
18	2	2	2	2	2	4	2	2	4	4	0	0,055452395	0,021209588	0,566643129	0	0,01568875 4	0,055604 02	0,0276200 1	0,05959606 5	0,04875898 8	0,053862555	0,001669713
19	2	2	2	2	2	4	2	4	2	2	0	0,079072403	0,025276654	0,550665082	0	0,01467997 9	0,080044 71	0,0132551 34	0,08730300 9	0,14286356 6	0,07472012	0,003594 39
20	2	2	2	2	2	4	2	4	2	4	0	0,041748269	0,026977641	0,733636304	0	0,01367217 7	0,041442752	0,0471409 86	0,04485553 5	0,02510363 4	0,046150806	0,002181426
21	2	2	2	2	2	4	2	4	4	2	0	0,005412 32	0,022226258	0,016250448	0	0,01730752 1	0,005482 76	0,0264072 72	0,00537388 9	0,02871010 1	0,010830751	0,001963928
22	2	2	2	2	2	4	4	2	2	2	0	0,067314707	0,011053324	0,291411089	0	0,01376493 8	0,069249998	0,0661127 43	0,07232866 9	0,10951455 2	0,064820994	0,002890052
23	2	2	2	2	2	4	4	2	2	4	0	0,025120349	0,010312889	0,797699153	0	0,00485160 1	0,024782788	0,0373387 82	0,02758641 9	0,04250761	0,022551187	0,001173391
24	2	2	2	2	2	4	4	2	4	2	0	0,031013305	0,032229 31	0,597199082	0	0,01486505 5	0,031842689	0,0374686 89	0,03307970 1	0,18213454 8	0,027607054	0,001624321
25	2	2	2	2	2	4	4	4	2	2	0	0,046126849	0,021123142	0,621406 57	0	0,02609960 6	0,046541782	0,0517083 68	0,05211007 1	0,10269872 3	0,04866278	0,001642016
26	2	2	2	2	4	2	2	2	2	2	0	0,023244409	0,016961971	0,330933949	0,04151677 5	0,01232154 5	0,024280249	0,0546064 59	0,02476980 4	0,12565224 1	0,032378984	0,001854766
27	2	2	2	2	4	2	2	2	2	4	0	0,031109851	0,022018072	0,428035191	0	0,00848198 1	0,031735414	0,0442287 52	0,02916263 3	0,07321854 8	0,050787353	0,001737 09
28	2	2	2	2	4	2	2	2	4	2	0	0,013221851	0,036487674	0,329480364	0	0,01954554 5	0,013055449	0,0406824 22	0,01684524 2	0,09834262	0,013957499	0,001486102
29	2	2	2	2	4	2	2	2	4	4	0	0,022869064	0,014537316	0,344855401	0	0,01407006 8	0,022084799	0,0252260 67	0,02849311 7	0,05690132 1	0,028669335	0,000919946
30	2	2	2	2	4	2	2	4	2	2	0	0,030644705	0,024524891	0,276675252	0	0,01356934 7	0,029074157	0,0372499 07	0,03934620 5	0,14337801 4	0,037524587	0,001419755
31	2	2	2	2	4	2	2	4	2	4	0	0,033365202	0,010341975	0,358704851	0	0,00973020 4	0,034530349	0,0199512 66	0,03743939	0,05186622 3	0,051288157	0,001556 3
32	2	2	2	2	4	2	2	4	4	2	0	0,042465548	0,034160852	0,261194319	0,04151677 5	0,02569946 8	0,043913428	0,0392419 88	0,04896161 4	0,17039961 8	0,045541612	0,002553971
33	2	2	2	2	4	2	4	2	2	2	0	0,030478337	0,020375575	0,407452104	0,08806947 6	0,01804775 4	0,029395 59	0,0217391 3	0,03566143 5	0,13947941 3	0,045007252	0,004556532
34	2	2	2	2	4	2	4	2	2	4	0	0,012945563	0,031485779	0,508308196	0	0,01441415 5	0,012943 27	0,0476720 36	0,01684078 4	0,05107374 8	0,012423429	0,001760313
35	2	2	2	2	4	2	4	2	4	2	0	0,031438483	0,023874169	0,134029452	0	0,01058222 5	0,031628983	0,0379525 19	0,02993123 2	0,12580689 7	0,03288558	0,002573323
36	2	2	2	2	4	2	4	4	2	2	0	0,026625693	0,015328883	0,286722152	0,05342046	0,01559614 7	0,028346896	0,0183881 32	0,02381486 3	0,16574150 1	0,02916537	0,002499 27
37	2	2	2	2	4	4	2	2	2	2	0	0,034999061	0,022281104	0,097225067	0	0,02106382 4	0,035223848	0,0457416 46	0,04173715	0,09826897 6	0,034055548	0,004247604
38	2	2	2	2	4	4	2	2	2	4	0	0,039711213	0,010811393	0,368312198	0	0,00931493 4	0,039422272	0,0259146 08	0,04723501 6	0,07481370 5	0,041550559	0,002213236
39	2	2	2	2	4	4	2	2	4	2	0	0,023615117	0,037069826	0,232639595	0	0,01921988 5	0,024282909	0,0523344 48	0,02046989 1	0,16352556 2	0,029805551	0,003475181
40	2	2	2	2	4	4	2	4	2	2	0	0,031098205	0,024729142	0,272274437	0,04151677 5	0,02555556 3	0,032587176	0,0548078 6	0,03305229 4	0,13280007	0,032034842	0,001877265
41	2	2	2	2	4	4	4	2	2	2	0	0,020196764	0,038377957	0,155962346	0	0,02120711 5	0,020529336	0,0827750 91	0,02040706 8	0,12068891 9	0,024013781	0,002350888
42	2	2	2	4	2	2	2	2	2	2	0	0,046484 02	0,019813183	0,528420095	0	0,01466792 6	0,048063738	0,0320857 51	0,05174340 4	0,09203301 3	0,042300732	0,003876791
43	2	2	2	4	2	2	2	2	2	2	4	0	0,026826834	0,016954624	0,683790171	0,12892051 3	0,00592933 4	0,026804891	0,0301136 05	0,02782251 3	0,04962645 8	0,041013842	0,001809709
44	2	2	2	4	2	2	2	2	4	2	0	0,020360848	0,017565993	0,589967909	0	0,00994303 3	0,020556465	0,0430985 94	0,01775959 3	0,06518831 7	0,014262707	0,002899768
45	2	2	2	4	2	2	2	2	4	4	0	0,040285491	0,022461 19	0,502426502	0	0,01221631 5	0,041050226	0,0298314 04	0,04226568 5	0,04308485 8	0,032654332	0,001673144
46	2	2	2	4	2	2	2	4	2	2	0	0,039068536	0,018259 13	0,382809593	0	0,00934310 3	0,038050721	0,0291024 11	0,04264992 2	0,11129117 9	0,042173953	0,000769849
47	2	2	2	4	2	2	2	4	2	4	0	0,057239 32	0,019456117	0,804117788	0	0,01223193 1	0,056841261	0,0445330 66	0,06029744 6	0,04975175 7	0,074035969	0,001293447

48	2	2	2	4	2	2	2	4	4	2	0	0,038259757	0,030641216	0,365597739	0	0,024525036	0,039224775	0,028821748	0,041331306	0,049077591	0,046043841	0,001806373
49	2	2	2	4	2	2	4	2	2	2	0	0,025300459	0,042491498	0,461203738	0	0,025822377	0,027804699	0,034264296	0,029730766	0,062448764	0,024582791	0,003352396
50	2	2	2	4	2	2	4	2	2	4	0	0,042796819	0,019024827	0,206804779	0	0,014615009	0,042328372	0,024556511	0,042912037	0,045287707	0,051529027	0,001162349
51	2	2	2	4	2	2	4	2	4	2	0	0,066778105	0,014110204	0,743055377	0	0,013519472	0,065095566	0,030200847	0,070540637	0,036282986	0,066695396	0,003347282
52	2	2	2	4	2	2	4	4	2	2	0	0,030816946	0,040630567	0,522085138	0	0,032943351	0,031408452	0,052562661	0,032026361	0,061472277	0,026328879	0,003008 8
53	2	2	2	4	2	4	2	2	2	2	0	0,036079416	0,026523795	0,570959938	0	0,015825233	0,037960972	0,031032254	0,039493841	0,110168956	0,033273793	0,002774272
54	2	2	2	4	2	4	2	2	2	4	0	0,040483045	0,009625 01	0,770077 19	0	0,007225907	0,040129826	0,027903083	0,041598045	0,050945474	0,04420217	0,001722045
55	2	2	2	4	2	4	2	2	4	2	0	0,032627 35	0,022736209	0,372409484	0	0,014207277	0,035134092	0,038100026	0,035455891	0,083899761	0,048560238	0,001437934
56	2	2	2	4	2	4	2	4	2	2	0	0,037069308	0,011088 33	0,522326811	0	0,007297097	0,035918027	0,023495012	0,040887846	0,059143173	0,022085921	0,002874253
57	2	2	2	4	2	4	4	2	2	2	0	0,056630995	0,031028715	0,230042289	0	0,013974842	0,056285 58	0,022699998	0,057699772	0,083047574	0,053041104	0,001656 51
58	2	2	2	4	4	2	2	2	2	2	0	0,015166366	0,023496395	0,108299339	0	0,017754828	0,014754297	0,030246681	0,017179673	0,093256871	0,016246634	0,002402 74
59	2	2	2	4	4	2	2	2	2	4	0	0,024259134	0,020700406	0,572715124	0	0,014470906	0,024147646	0,020244003	0,024995388	0,053667011	0,035719794	0,000962534
60	2	2	2	4	4	2	2	2	4	2	0	0,029618643	0,038367051	0,195591 22	0	0,014392417	0,028828591	0,043500953	0,027696501	0,090244656	0,036569512	0,003017212
61	2	2	2	4	4	2	2	4	2	2	0	0,035409632	0,023068719	0,249472216	0	0,012790584	0,034389834	0,020217896	0,041028112	0,063226033	0,02894415	0,002686287
62	2	2	2	4	4	2	4	2	2	2	0	0,043613335	0,022172648	0,256118662	0	0,025230206	0,042984339	0,047890965	0,044863022	0,095992042	0,045569594	0,002481078
63	2	2	2	4	4	4	2	2	2	2	0	0,025024681	0,017819455	0,170225078	0	0,007792337	0,025125679	0,039067123	0,030928307	0,068602537	0,040246231	0,002632892
64	2	2	4	2	2	2	2	2	2	2	0	0,029283995	0,019208841	0,026649512	0	0,019852408	0,029439006	0,013335043	0,028938078		0,025974518	0,002471 88
65	2	2	4	2	2	2	2	2	2	4	0	0,002359197	0,010284033	0,014755622	0	0,006755258	0,002375381	0,036695 24	0,002607439		0,022046905	0,001294483
66	2	2	4	2	2	2	2	2	4	2	0	0,014075015	0,010770411	0,037665804	0	0,011041773	0,014012516	0,023062558	0,013500926		0,014798557	0,001695155
67	2	2	4	2	2	2	2	2	4	4	0	0,003994875	0,009173499	0,020456491	0	0,005197777	0,004006666	0,025342984	0,003827102		0,011966904	0,000658022
68	2	2	4	2	2	2	2	4	2	2	0	0,016251586	0,016127614	0,037526391	0	0,012132611	0,016193934	0,029903652	0,015746956		0,018848552	0,001793494
69	2	2	4	2	2	2	2	4	2	4	0	0,004090142	0,014578038	0,013469769	0	0,006963922	0,004074561	0,010182206	0,004191031		0,014524979	0,001295234
70	2	2	4	2	2	2	2	4	4	2	0	0,013927242	0,009679919	0,033584778	0	0,009466242	0,014122813	0,052057096	0,014159722		0,024395393	0,001475125
71	2	2	4	2	2	2	4	2	2	2	0	0,010386336	0,008329237	0,023689461	0	0,006008284	0,010438001	0,018672361	0,010236221		0,020660928	0,002458685
72	2	2	4	2	2	2	4	2	2	4	0	0,002185443	0,007431928	0,013415697	0	0,006163865	0,002214061	0,015328763	0,002513539		0,012576113	0,000580249
73	2	2	4	2	2	2	4	2	4	2	0	0,019189242	0,020522619	0,022618338	0	0,007086326	0,019200504	0,016665615	0,019054904		0,025417928	0,002545 61
74	2	2	4	2	2	2	4	4	2	2	0	0,013919496	0,027294183	0,015761308	0	0,020529184	0,014215445	0,020726145	0,013778436		0,016946301	0,001072564
75	2	2	4	2	2	4	2	2	2	2	0	0,008581756	0,021756109	0,018078 61	0	0,012326067	0,008737423	0,024710825	0,008603365		0,023276294	0,002099476
76	2	2	4	2	2	4	2	2	2	4	0	0,003578797	0,012204 8	0,021001943	0	0,008330106	0,003618183	0,019461791	0,003397529		0,014144183	0,000882317
77	2	2	4	2	2	4	2	2	4	2	0	0,009656414	0,014970272	0,026140713	0	0,007380909	0,009834592	0,039740216	0,008908483		0,024775461	0,002123663
78	2	2	4	2	2	4	2	4	2	2	0	0,022539136	0,020667 67	0,016014402	0	0,011101031	0,022631916	0,033945578	0,022274752		0,028248229	0,001016116
79	2	2	4	2	2	4	4	2	2	2	0	0,013208446	0,008545479	0,036643771	0	0,010866637	0,013175092	0,030023137	0,013295077		0,016532403	0,002587803
80	2	2	4	2	4	2	2	2	2	2	0	0,012125 93	0,016775444	0,021046439	0,041516775	0,016889256	0,012201861	0,039994254	0,011840767		0,030808014	0,003225293
81	2	2	4	2	4	2	2	2	2	4	0	0,004824464	0,014052402	0,028676128	0	0,007427356	0,004835 62	0,014223031	0,004996399		0,021522937	0,000954659
82	2	2	4	2	4	2	2	2	4	2	0	0,018123217	0,023532989	0,023744 98	0	0,017743376	0,018357578	0,039255886	0,018066786		0,020965152	0,002072622
83	2	2	4	2	4	2	2	4	2	2	0	0,010988046	0,014927761	0,037805593	0,041516775	0,010754347	0,011148378	0,030454348	0,012706633		0,021313729	0,001769172
84	2	2	4	2	4	2	4	2	2	2	0	0,012384361	0,015407762	0,058677062	0	0,020441511	0,012321969	0,056292581	0,011771637		0,016190383	0,002126758
85	2	2	4	2	4	4	2	2	2	2	0	0,010113555	0,014490523	0,014804999	0	0,018479895	0,010263946	0,035459761	0,009729597		0,015918854	0,002657 96
86	2	2	4	4	2	2	2	2	2	2	0	0,007602073	0,010719585	0,016911475	0	0,017459294	0,007633988	0,015628831	0,007397377		0,017763534	0,002466392
87	2	2	4	4	2	2	2	2	2	4	0	0,003428044	0,008717793	0,014273715	0	0,003282433	0,003441985	0,013414302	0,003435786		0,01092098	0,000839196
88	2	2	4	4	2	2	2	2	4	2	0	0,017900656	0,013454204	0,017488595	0	0,014730457	0,018034 39	0,040679381	0,017921036		0,016213698	0,002210778
89	2	2	4	4	2	2	2	4	2	2	0	0,006544 71	0,015072815	0,023151496	0	0,009164885	0,006553235	0,017996833	0,006613187		0,011401354	0,001529 82
90	2	2	4	4	2	2	4	2	2	2	0	0,010275175	0,015956 07	0,015866106	0	0,008571399	0,010368102	0,027656004	0,010455272		0,017223752	0,001822572
91	2	2	4	4	2	4	2	2	2	2	0	0,019640756	0,016736511	0,016292937	0	0,004942647	0,019815746	0,041530163	0,019565677		0,032504715	0,001518798
92	2	2	4	4	4	2	2	2	2	2	0	0,010205347	0,020374865	0,026683328	0,041516775	0,024170551	0,010131 56	0,035043682	0,010302693		0,0156656	0,001324735

93	2	4	2	2	2	2	2	2	2	2	0	0,033366148	0,021041403	0,842279106	0	0,01540432	0,034135849	0,028799811	0,03392699	8	0,18878531	9	0,067679445	0,004285292
94	2	4	2	2	2	2	2	2	2	4	0	0,042272668	0,015152971	0,348498549	0	0,01734440	3	0,043293588	0,035875688	0,04441344	4	0,06200731	7	0,104298641	0,001585166
95	2	4	2	2	2	2	2	2	4	2	0	0,057096776	0,023664926	0,292644531	0	0,01607856	3	0,056981561	0,043782935	0,06463316	8	0,07603649	8	0,076251871	0,003681316
96	2	4	2	2	2	2	2	2	4	4	0	0,025474313	0,019311225	0,586942433	0	0,01263834	7	0,025270077	0,014912566	0,02846087	2	0,03663092	7	0,179853107	0,000391994
97	2	4	2	2	2	2	2	4	2	2	0	0,062775127	0,022453241	0,394072838	0	0,01203847	1	0,065271536	0,025788761	0,06502596	8	0,19048465	2	0,088222937	0,002853403
98	2	4	2	2	2	2	2	4	2	4	0	0,04947127	0,012357955	0,443240588	0	0,00600757	3	0,050255441	0,018138247	0,05225696	0,04033068	2	0,132802347	0,000928257	
99	2	4	2	2	2	2	2	4	4	2	0	0,081360026	0,022291041	0,273761827	0	0,00884548	2	0,081232047	0,034845842	0,08567025	4	0,14072648	1	0,093045996	0,001559563
100	2	4	2	2	2	2	4	2	2	2	0	0,089561158	0,023417135	0,585182073	0	0,01751597	8	0,091771429	0,006221199	0,09668055	3	0,23850899	2	0,081291079	0,004709489
101	2	4	2	2	2	2	4	2	2	4	0	0,028007607	0,020263208	0,425472851	0	0,01580655	1	0,027616633	0,029796174	0,03118671	7	0,04221002	9	0,077114613	0,001021071
102	2	4	2	2	2	2	4	2	4	2	0	0,036213695	0,015624718	0,195208266	0	0,00804547	0,035367606	0,035828499	0,04091194	1	0,08144527	8	0,047258457	0,003896434	
103	2	4	2	2	2	2	4	4	2	2	0	0,040653338	0,016109449	0,372330684	0	0,01165368	7	0,039234298	0,044137618	0,04773609	2	0,15280974	2	0,050509651	0,002592224
104	2	4	2	2	2	4	2	2	2	2	0	0,085745613	0,016331857	0,423490442	0	0,02227107	4	0,087794214	0,048498665	0,09129399	5	0,25008454	1	0,0792951	0,002896432
105	2	4	2	2	2	4	2	2	2	4	0	0,031634207	0,019977434	0,410108106	0	0,02014264	8	0,032719003	0,046333825	0,03287839	6	0,03932382	8	0,111573711	0,000900981
106	2	4	2	2	2	4	2	2	4	2	0	0,020147042	0,030138616	0,268065768	0	0,01108796	9	0,021702366	0,072111026	0,02352173	0,10766865	3	0,053586505	0,002478477	
107	2	4	2	2	2	4	2	4	2	2	0	0,034019126	0,021621889	0,421924874	0	0,01031368	7	0,034027949	0,012073632	0,03587380	1	0,11501306	6	0,049343135	0,001852483
108	2	4	2	2	2	4	4	2	2	2	0	0,075676744	0,028327574	0,493512533	0	0,01692356	1	0,073845181	0,037787177	0,08070480	4	0,07828818	8	0,093015502	0,002824392
109	2	4	2	2	4	2	2	2	2	2	0	0,054553842	0,029065572	0,130814096	0	0,01675891	2	0,056288179	0,043437224	0,05855822	9	0,12606039	0,067860785	0,001784654	
110	2	4	2	2	4	2	2	2	2	4	0	0,028279647	0,020815058	0,386759071	0	0,00743835	3	0,028268922	0,025802809	0,03178561	0,05301172	8	0,10265739	0,001224104	
111	2	4	2	2	4	2	2	2	4	2	0	0,013221221	0,028233601	0,244977613	0	0,01429445	4	0,015105137	0,043521382	0,01299214	0,07533939	9	0,043053077	0,002593206	
112	2	4	2	2	4	2	2	4	2	2	0	0,014693725	0,021703709	0,405634462	0,04151677	5	0,01579889	1	0,013754331	0,03600054	4	0,01903414	3	0,15145390	2	0,05298181	0,002940471
113	2	4	2	2	4	2	4	2	2	2	0	0,017010509	0,023597838	0,360909418	0	0,01726477	1	0,016270964	0,027013318	0,02145686	5	0,15478351	4	0,030773339	0,002269474
114	2	4	2	2	4	4	2	2	2	2	0	0,044075996	0,010642632	0,245473636	0	0,01608047	6	0,045044825	0,041851816	0,04903766	5	0,07103052	7	0,069891657	0,001483947
115	2	4	2	4	2	2	2	2	2	2	0	0,055561196	0,015128892	0,383346223	0	0,00761266	3	0,056228193	0,029034035	0,06291279	4	0,02989328	5	0,076329916	0,002614 05
116	2	4	2	4	2	2	2	2	2	4	0	0,025737661	0,013652546	0,912948803	0	0,01452654	2	0,025006587	0,042710539	0,03041031	0,03390652	7	0,179672745	0,000724353	
117	2	4	2	4	2	2	2	2	4	2	0	0,045689485	0,017996496	0,412421404	0	0,00608158	0,045194611	0,034073153	0,04822120	4	0,04965818	7	0,056879992	0,001680786	
118	2	4	2	4	2	2	2	4	2	2	0	0,041732878	0,022165727	0,241078692	0	0,00785959	1	0,043062949	0,048734523	0,04629384	6	0,07329548	7	0,037642899	0,002771718
119	2	4	2	4	2	2	4	2	2	2	0	0,021009643	0,019677 38	0,323447856	0	0,00434874	3	0,020864457	0,024190329	0,02261607	5	0,07638201	5	0,042767043	0,002967746
120	2	4	2	4	2	4	2	2	2	2	0	0,088420736	0,018431022	0,204814 68	0	0,01336977	4	0,089306779	0,044196026	0,08889180	2	0,18547972	4	0,073752484	0,001364144
121	2	4	2	4	4	2	2	2	2	2	0	0,044432264	0,021576545	0,418045779	0	0,01741501	7	0,042068903	0,054570493	0,05220551	1	0,12243168	8	0,05501616	0,002873631
122	2	4	4	2	2	2	2	2	2	2	0	0,019038019	0,011026 11	0,019785444	0	0,01198125	8	0,019094459	0,019122055	0,01831863	5	0,03525143	0,001596148		
123	2	4	4	2	2	2	2	2	2	4	0	0,00356363	0,008845153	0,016152867	0	0,01039413	0,003559031	0,010119289	0,00357524	9	0,09566313 6	0,000653514			
124	2	4	4	2	2	2	2	2	4	2	0	0,013807 07	0,019409507	0,016594683	0	0,01191264	2	0,013861192	0,019029524	0,01353338	2	0,05231524 9	0,002065684		
125	2	4	4	2	2	2	2	4	2	2	0	0,023290941	0,021510764	0,013340834	0	0,00568624	1	0,023563452	0,035917352	0,02329672	3	0,02857664 6	0,002830085		
126	2	4	4	2	2	2	4	2	2	2	0	0,016759482	0,018787534	0,024303523	0	0,00554132	9	0,016996355	0,012740077	0,01599379	3	0,02740184 7	0,001608566		
127	2	4	4	2	2	4	2	2	2	2	0	0,015794549	0,018407266	0,022343076	0	0,00592026	6	0,016177199	0,014768615	0,01594087	3	0,03252287 6	0,002976726		
128	2	4	4	2	4	2	2	2	2	2	0	0,013116782	0,024831206	0,035232395	0	0,00542995	4	0,013199933	0,044098914	0,01294633	0,04547747 4	0,001314961			
129	2	4	4	4	2	2	2	2	2	2	0	0,011081784	0,010975466	0,014654457	0	0,00860047	2	0,011257017	0,029019057	0,01099846	7	0,03842846 8	0,002635 1		
130	4	2	2	2	2	2	2	2	2	2	0	0,03796324	0,011101677	0,559998764	0,47864058	4	0,00780486	4	0,039936545	0,03951373	2	0,03752148 6	0,18894315	0,047240764	0,004673615
131	4	2	2	2	2	2	2	2	2	4	0	0,25283428	0,008493 08	1,160266243	0,38729833	5	0,00413378	3	0,251714 89	0,02940354	0,25103419	2	0,27405187	5	0,240713131	0,048700314
132	4	2	2	2	2	2	2	2	4	2	0	0,05992496	0,014110103	0,784658768	0,59287219	8	0,01079528	8	0,060794314	0,03702306	0,06617996	0,23611753 7	0,052898096	0,004391226	
133	4	2	2	2	2	2	2	2	4	4	0	0,042566941	0,009309815	0,682578384	0,36331804	2	0,00469490	8	0,042049364	0,03588465	5	0,04452699 5	0,05368800 8	0,061834263	0,001902608
134	4	2	2	2	2	2	2	4	2	2	0	0,058193173	0,014916454	0,294985061	0,2718553	0,01238194	8	0,056696 47	0,026870306	0,06183340	5	0,13182177 3	0,06558006	0,002492372	
135	4	2	2	2	2	2	2	4	2	4	0	0,222507397	0,009088502	1,166359772	0,85933784	9	0,01336018	8	0,222395616	0,02292807	8	0,22197267 7	0,23676533 9	0,222139894	0,047345811
136	4	2	2	2	2	2	2	4	4	2	0	0,045282567	0,011032329	0,854187285	0,37757191	8	0,00734164	7	0,047547041	0,03685933	9	0,04666579 3	0,18444632	0,046841921	0,002905417
137	4	2	2	2	2	2	2	4	2	2	0	0,247489823	0,010161561	0,606880326	0,82945765	4	0,01011405	4	0,246125332	0,02239339	4	0,24552002 7	0,32531806	0,244242593	0,291361537

138	4	2	2	2	2	2	4	2	2	4	0	0,048935231	0,006109485	0,583380153	0,539775902	0,007015828	0,049412225	0,005492406	0,051781752	0,053584284	0,063749713	0,001346173
139	4	2	2	2	2	2	4	2	4	2	0	0,036292228	0,011263759	0,447999397	0,452546739	0,006925224	0,037100955	0,030371613	0,036395938	0,229219977	0,045307913	0,004398688
140	4	2	2	2	2	2	4	4	2	2	0	0,052595555	0,009041619	0,232676859	0,641179469	0,008871594	0,052082198	0,030896378	0,055024077	0,153165617	0,058412883	0,003409411
141	4	2	2	2	2	4	2	2	2	2	0	0,036571277	0,009987626	0,765528367	0,531664054	0,011000765	0,036456905	0,052472393	0,034737353	0,178008054	0,032727366	0,004159223
142	4	2	2	2	2	4	2	2	2	4	0	0,084375505	0,008311074	0,865772584	0,659694016	0,009610369	0,084119553	0,031320207	0,082602193	0,123518799	0,096558939	0,024731141
143	4	2	2	2	2	4	2	2	4	2	0	0,046288532	0,006131362	0,126251895	0,368294565	0,006564567	0,046719858	0,042787588	0,049041457	0,133293069	0,051934069	0,002783 8
144	4	2	2	2	2	4	2	4	2	2	0	0,049492868	0,004420 26	0,179365382	0,392750949	0,008610139	0,048824 63	0,016360352	0,051956105	0,138570977	0,053933325	0,002891 77
145	4	2	2	2	2	4	4	2	2	2	0	0,034224954	0,004964183	0,369728519	0,272805454	0,005800188	0,033048039	0,023188866	0,035380364	0,146035491	0,023436043	0,002903027
146	4	2	2	2	4	2	2	2	2	2	0	0,274216823	0,010052273	0,600253 4	0,493710441	0,008146297	0,272862349	0,029815351	0,027294945	0,455486355	0,269645645	0,291190507
147	4	2	2	2	4	2	2	2	2	4	0	0,046406441	0,008175542	0,384134387	0,727796331	0,006670254	0,047174409	0,037888599	0,047697696	0,128546926	0,045660514	0,003430 66
148	4	2	2	2	4	2	2	2	4	2	0	0,024087 15	0,018042898	0,128123427	0,223427841	0,017373754	0,024696271	0,032948284	0,024945662	0,243905576	0,03516986	0,002264357
149	4	2	2	2	4	2	2	4	2	2	0	0,273037984	0,007427783	0,500960623	0,918936583	0,009205746	0,271639675	0,061537981	0,271157833	0,350585779	0,278955522	0,291695861
150	4	2	2	2	4	2	4	2	2	2	0	0,041128151	0,016040109	0,233415124	0,452476187	0,020276157	0,041158 24	0,076307645	0,047752154	0,220082189	0,0356607	0,002936518
151	4	2	2	2	4	4	2	2	2	2	0	0,021477382	0,005720752	0,257386471	0,231083838	0,008421289	0,021194149	0,064789003	0,021987431	0,096203526	0,027105857	0,002923126
152	4	2	2	4	2	2	2	2	2	2	0	0,028220536	0,003721429	0,383316091	0,571428571	0,006620456	0,027641872	0,036491183	0,025068663	0,114300923	0,045906768	0,005782531
153	4	2	2	4	2	2	2	2	2	4	0	0,061619794	0,003711104	0,976174041	0,750999334	0,002735484	0,061074355	0,020037133	0,062157079	0,051489822	0,08633155	0,012346577
154	4	2	2	4	2	2	2	2	4	2	0	0,031799405	0,013056007	0,477294872	0,737864787	0,013017642	0,033380388	0,025599916	0,034901171	0,095459094	0,048781082	0,004055284
155	4	2	2	4	2	2	2	4	2	2	0	0,016951228	0,008566641	0,896042963	0,726907877	0,008627544	0,017683444	0,020426444	0,020892236	0,081245452	0,029009238	0,002443299
156	4	2	2	4	2	2	4	2	2	2	0	0,025012578	0,018084019	1,020245016	0,645233702	0,014710891	0,025806034	0,016339476	0,026939732	0,081297952	0,015415532	0,003279846
157	4	2	2	4	2	4	2	2	2	2	0	0,040423572	0,005136535	0,340280568	0,686119255	0,006986005	0,040801564	0,040623844	0,041796794	0,061578954	0,032510384	0,005033 84
158	4	2	2	4	4	2	2	2	2	2	0	0,008454297	0,014267169	0,637720033	0,748776317	0,008783917	0,009232215	0,057015922	0,013721446	0,140540553	0,023479116	0,003211167
159	4	2	4	2	2	2	2	2	2	2	0	0,005434971	0,006978415	0,010676406	0	0,004302017	0,005488243	0,014812169	0,005726062		0,013439328	0,000496297
160	4	2	4	2	2	2	2	2	2	4	0	0,001702445	0,003374 32	0,019029371	0	0,003910406	0,001709 29	0,010434344	0,001767383		0,020396787	0,000388359
161	4	2	4	2	2	2	2	2	4	2	0	0,007843023	0,005602 69	0,023770036	0	0,010601414	0,007989493	0,010489135	0,008409143		0,009727308	0,001317299
162	4	2	4	2	2	2	2	4	2	2	0	0,006111422	0,008491417	0,011000331	0	0,003148137	0,006192349	0,020338 8	0,005985557		0,021601531	0,001598438
163	4	2	4	2	2	2	4	2	2	2	0	0,009575254	0,007771563	0,019042 84	0	0,004431108	0,009558559	0,018939 96	0,009553966		0,01400054	0,001153959
164	4	2	4	2	2	4	2	2	2	2	0	0,007194854	0,005226 73	0,010081133	0	0,006060566	0,007233371	0,011337886	0,007158317		0,015094994	0,000825452
165	4	2	4	2	4	2	2	2	2	2	0	0,009176976	0,009495217	0,021323761	0,041516775	0,008223102	0,009137585	0,027262389	0,008631461		0,021633866	0,001392 62
166	4	2	4	4	2	2	2	2	2	2	0	0,006671421	0,008412531	0,017721	0	0,005134931	0,006710996	0,017655438	0,006887596		0,021129426	0,000907165
167	4	4	2	2	2	2	2	2	2	2	0	0,118000091	0,013709772	0,552056572	0,758946638	0,009318926	0,116436 24	0,027556196	0,120357736	0,145006195	0,119251102	0,111154866
168	4	4	2	2	2	2	2	2	2	4	0	0,046361107	0,011179272	0,338176 61	0,384499675	0,013357582	0,046707236	0,034203133	0,048068194	0,026906022	0,693997589	0,001212804
169	4	4	2	2	2	2	2	2	4	2	0	0,042195939	0,005886408	0,421047966	0,510449989	0,008702913	0,042482887	0,036141418	0,043756393	0,128922836	0,067115328	0,005446544
170	4	4	2	2	2	2	2	4	2	2	0	0,038373752	0,006087836	0,623792018	0,445286166	0,004982516	0,041786259	0,028553548	0,038042955	0,220823934	0,062183551	0,003806701
171	4	4	2	2	2	2	4	2	2	2	0	0,048824603	0,008461082	0,802234162	0,387298335	0,013067595	0,047908955	0,020371345	0,053117755	0,127209559	0,051659905	0,002807723
172	4	4	2	2	2	4	2	2	2	2	0	0,038123046	0,006432585	0,259400858	0,260584015	0,012454602	0,037695424	0,035630683	0,043205966	0,088535116	0,054963887	0,002136 86
173	4	4	2	2	4	2	2	2	2	2	0	0,281624356	0,015963485	0,580725307	0,626613655	0,010696462	0,280711633	0,038513289	0,280050298	0,354680476	0,314509495	0,289193876
174	4	4	2	4	2	2	2	2	2	2	0	0,030916842	0,009880461	0,509336199	0,542977448	0,009361629	0,031067299	0,016584954	0,031382357	0,052727347	0,082397072	0,002635915
175	4	4	4	2	2	2	2	2	2	2	0	0,008151812	0,008034581	0,021071025	0	0,006895263	0,008139 66	0,013216 16	0,007951216		0,022723263	0,001856461
												0,040776419	**0,016884 31**	**0,342335 487**	**0,116210368**	**0,012296714**	**0,040981 072**	**0,031956364**	**0,042731166**	**0,114321736**	**0,056041052**	**0,010144 538**

Table 10 Standard Deviations as Fraction of Mean (Georg & Lau, 2016)