Scientific Writing

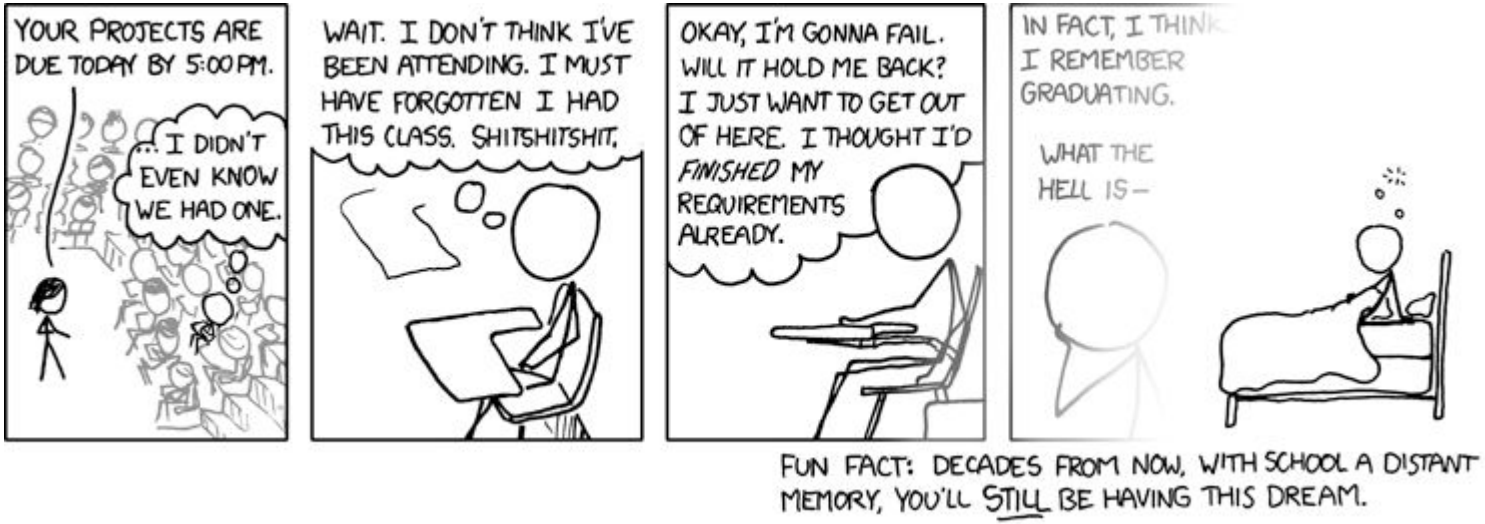
Dr. Hannah Franziska Löchel, Jan Ruhland

Overview

- Formalia
- Structure of a thesis
- References
- Figures
- Tables
- Units and equations
- Source Code / Listings
- Language and Expression

Formalia

It is your job to read the examination regulations!!



https://xkcd.com/557/

Structure of the thesis

Structure of a thesis

Scope of the thesis:

- accurate documentation of the work
- should enable the reader to understand each step and reproduce it
- even if the reader is not a specialist in the subject, it should be possible to comprehend the work based on the information provided in the thesis.

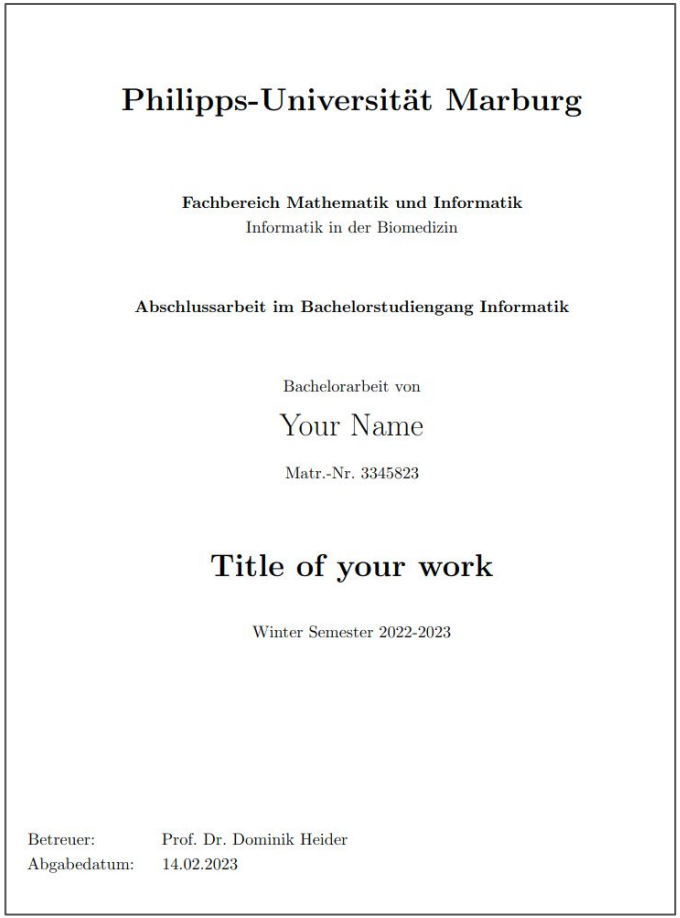
Structure of a thesis

- Cover Page
- Acknowledgement (optional)
- Declaration of Independence
- Abstract
- Zusammenfassung (depending on exam regulation, a German and English abstract may be mandatory)
- Table of Contents
- List of Abbreviations (optional)
- List of Figures (optional)
- List of Tables (optional)
- List of Listings (optional)
- →Contents of the thesis
- Appendix (optional)
- Bibliography

Ask your supervisor if a List of Abbreviations, Figures, Tables, listings or Appendix is necessary

Cover Page

- Full name of the student
- Matriculation number
- Title of the thesis
- Supervisor
- 2nd examiner (if known)
- Date of submission
- Department
- ”To obtain the academic degree of Bachelor/Master of Science



Philipps-Universität Marburg

Fachbereich Mathematik und Informatik
Informatik in der Biomedizin

Abschlussarbeit im Bachelorstudiengang Informatik

Bachelorarbeit von

Your Name

Matr.-Nr. 3345823

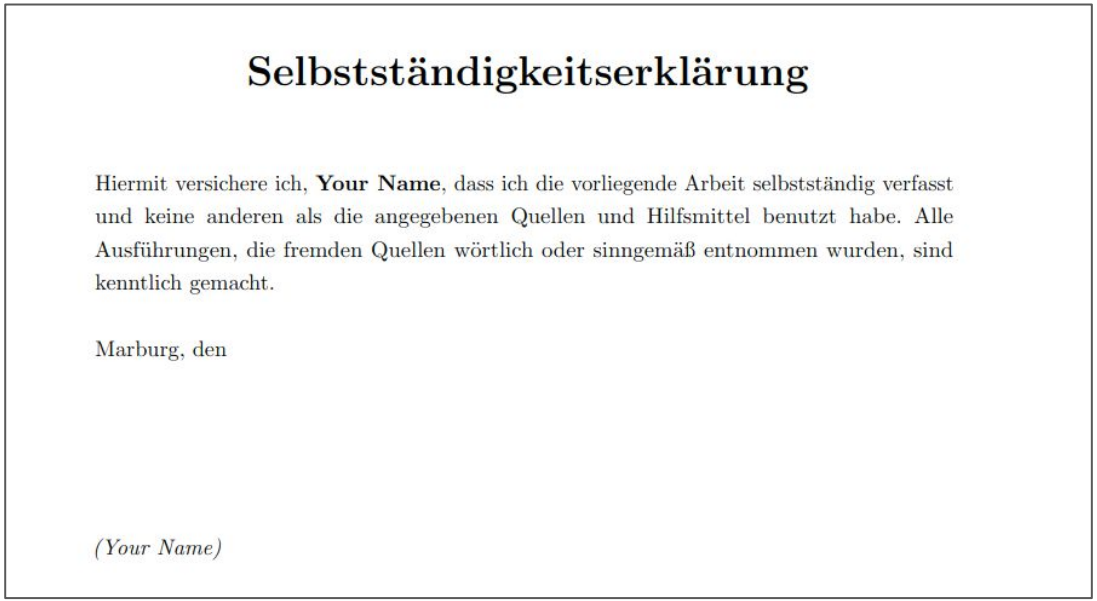
Title of your work

Winter Semester 2022-2023

Betreuer: Prof. Dr. Dominik Heider
Abgabedatum: 14.02.2023

Declaration of Independence

- Assurance that the work was done independently and only with the help of the specified aids
- Often the department provides a prefabricated wording of the declaration, which must be integrated into the work and signed by the student



Selbstständigkeitserklärung

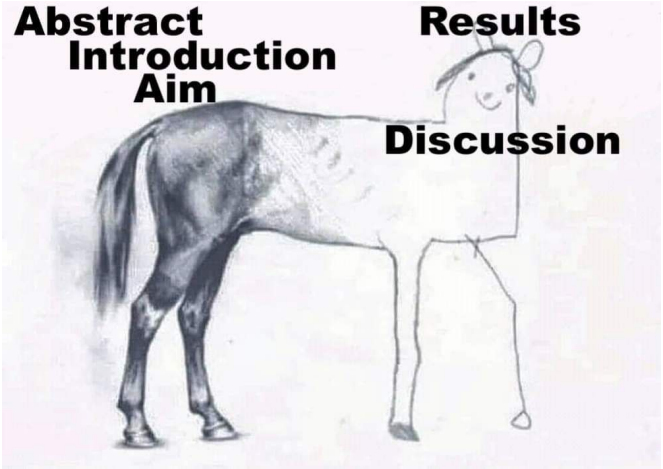
Hiermit versichere ich, **Your Name**, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Alle Ausführungen, die fremden Quellen wörtlich oder sinngemäß entnommen wurden, sind kenntlich gemacht.

Marburg, den

(Your Name)

Zusammenfassung and Abstract

- Complete summary of the work has to be included at the beginning of the thesis
- It has to be written in German (Zusammenfassung) and English(Abstract)
- The abstract describes the relevance of the task, the implementation, and essential aspects of the results
- ~0.5- 1 page



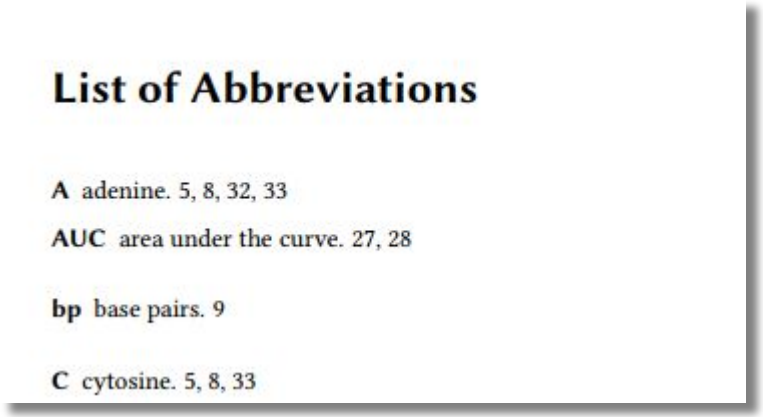
https://ifunny.co/picture/abstract-results-introduction-reno-discussion-Jt5zgymF7

Table of Content

- Overview of the chapters and subchapters of the thesis
- All indexes and the appendix are listed in the table of contents
- Content of the thesis is numbered with Arabic numerals and
- Everything else with Roman numerals
- The content of chapters and sections should be clear from the titles

List of Abbreviations/Figures/Tables Listings

- Figures, tables, source code, or abbreviations used in the thesis should be listed in an index to give a better overview
- Table of figures, tables, and source code has a similar structure to the list of content



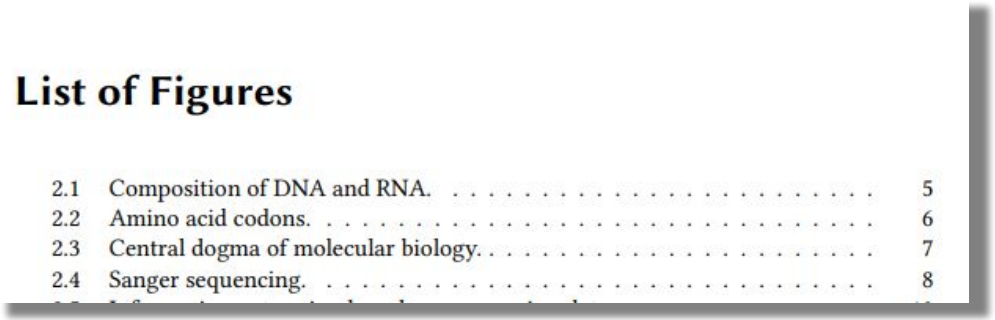
List of Abbreviations

A adenine. 5, 8, 32, 33

AUC area under the curve. 27, 28

bp base pairs. 9

C cytosine. 5, 8, 33



List of Figures

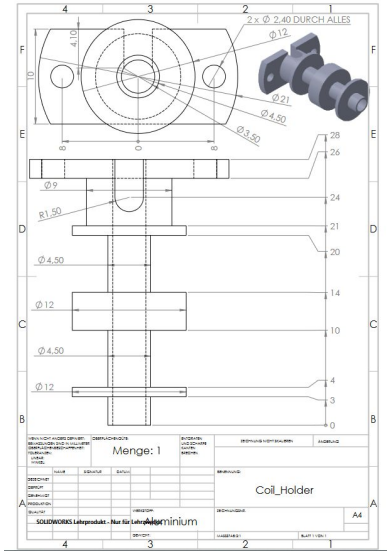
Löchel HF. Fractals in Bioinformatics: Visualization, Analysis, and Construction of Molecular Codes. University Marburg. 2021.

Appendix

In rare cases, an appendix is necessary. If possible, everything should be incorporated into the actual content of the thesis.

Examples for exceptions:

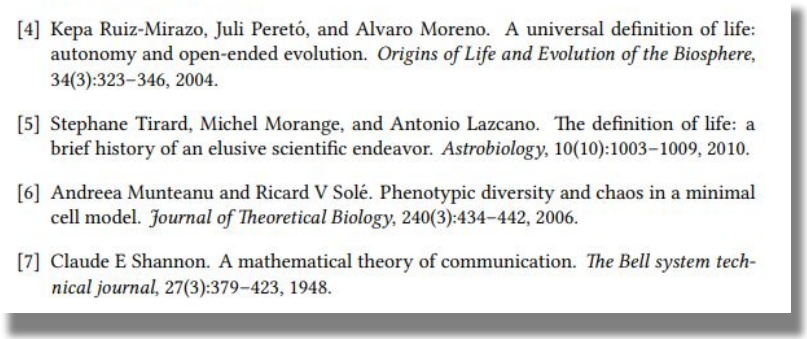
- Technical drawings
- Images of graphical interfaces
- Hardware specifications



J. B. Ruhland. First Approach on the Measurement of the Magnetocaloric Susceptibility. Technical University Munich. 2017.

Bibliography

The bibliography lists all sources used



[4] Kepa Ruiz-Mirazo, Juli Peretó, and Alvaro Moreno. A universal definition of life: autonomy and open-ended evolution. *Origins of Life and Evolution of the Biosphere*, 34(3):323–346, 2004.

[5] Stephane Tirard, Michel Morange, and Antonio Lazcano. The definition of life: a brief history of an elusive scientific endeavor. *Astrobiology*, 10(10):1003–1009, 2010.

[6] Andreea Munteanu and Ricard V Solé. Phenotypic diversity and chaos in a minimal cell model. *Journal of Theoretical Biology*, 240(3):434–442, 2006.

[7] Claude E Shannon. A mathematical theory of communication. *The Bell system technical journal*, 27(3):379–423, 1948.

Löchel HF. Fractals in Bioinformatics: Visualization, Analysis, and Construction of Molecular Codes. University Marburg. 2021.

Content of the thesis

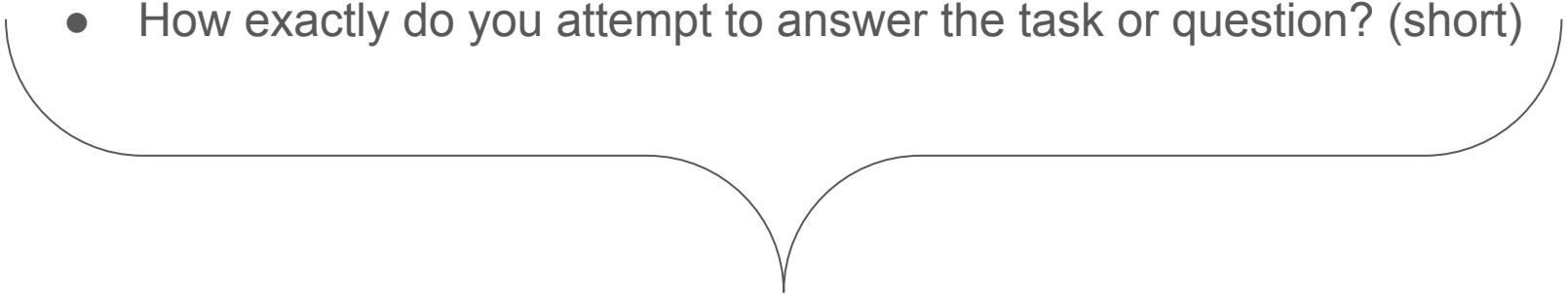
Content of the thesis

1. Introduction
2. Background
3. Material and Methods
4. Results
5. Discussion
6. Future Perspectives (optional)
7. Conclusion (optional)

1.Introduction

The introduction should answer the following questions:

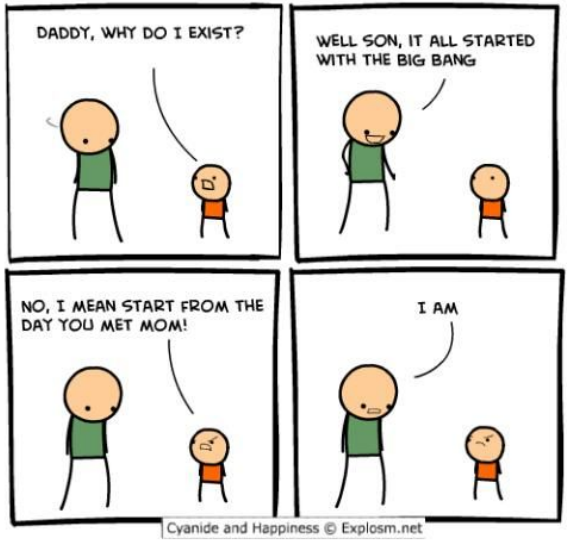
- Why is the topic relevant?
- What is the current state of the science on this topic?
- What exactly is the task or research question?
- How exactly do you attempt to answer the task or question? (short)



Motivate your work!

2. Background

- This section transports all relevant background information
- The later used methods, all formulas applied in the thesis, and the biological background are explained
- This part should contain the basic knowledge the reader needs to understand in the following sections



https://explosm.net/comics/dave-comicexist

3. Material and Methods /Implementation

This chapter should be divided into two subsections:

3.1 Material

3.2 Methods

3.1 Material

It is advisable to list all used materials in tabular form, i.e., software, packages (with version number), databases or existing data sets, and, if necessary, hardware components.

Nevertheless, the table should also be explained in the continuous text.

3.2 Methods

The methods section of the thesis should accurately describe the following:

- What has been done?
- How was proceeded to work on the task?/ How was a certain software implemented? (How is it structured?)
- Which tools were used to calculate what?
- Which software packages were used where? The reader should be able to reproduce the work in exactly the same way according to the methods section.

Results

In this section, the results are presented but not yet interpreted. The results should be described as detailed as possible in the text, using tables and figures.

Discussion

In this part, the results are discussed and interpreted. The following questions should be answered:

- Do the results meet the expectations?
  - If yes: substantiate with literature!
  - If not: also substantiates with literature that contradicts it and finds a possible explanation!
- How are the results to be interpreted?
- Can conclusions be drawn from the results, and if so, which ones?

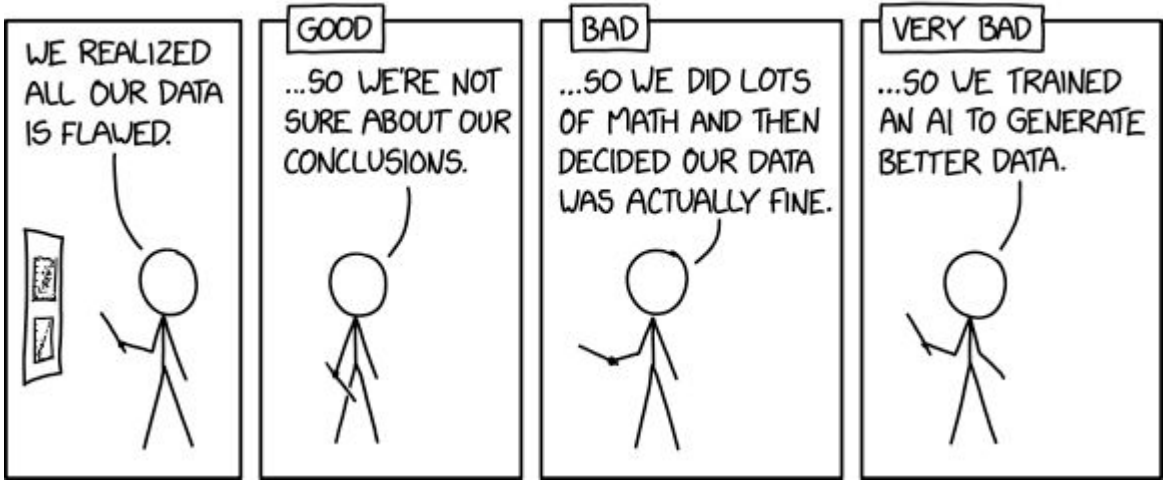
Be critical about your own work

Future Perspectives

The results may have raised new questions that need to be answered in the future. It is possible that the results do not quite meet expectations, so these questions needs to be investigated using other methods.

Conclusion

Here, the core statements of the work are summarized once again and assessed in the conclusion.



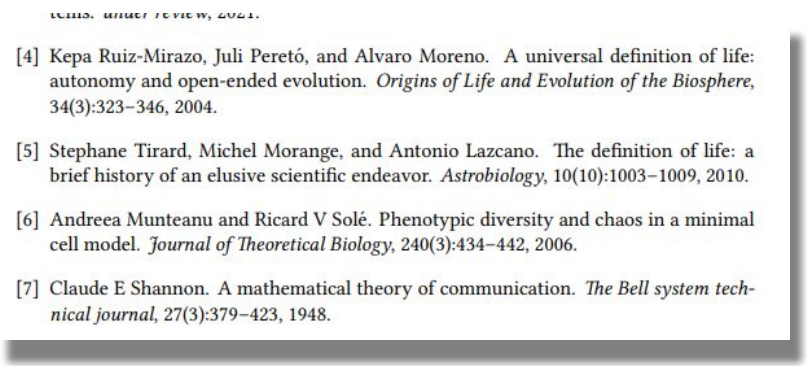
https://xkcd.com/2494/

Literature

References

The source must be indicated in the text as follows [1]. Between references and words, has to be a protected space, to avoid a new line.

Bibliography



[4] Kepa Ruiz-Mirazo, Juli Peretó, and Alvaro Moreno. A universal definition of life: autonomy and open-ended evolution. *Origins of Life and Evolution of the Biosphere*, 34(3):323–346, 2004.

[5] Stephane Tirard, Michel Morange, and Antonio Lazcano. The definition of life: a brief history of an elusive scientific endeavor. *Astrobiology*, 10(10):1003–1009, 2010.

[6] Andreea Munteanu and Ricard V Solé. Phenotypic diversity and chaos in a minimal cell model. *Journal of Theoretical Biology*, 240(3):434–442, 2006.

[7] Claude E Shannon. A mathematical theory of communication. *The Bell system technical journal*, 27(3):379–423, 1948.

- books: name [no.] author, title, edition, place of publication,publisher, year, number of pages.
- Paper: [no.] author, title, journal, year, journal number, pages.
- Online articles (Attention, not always citable!): . [No.] author, title [Online].Available:URL(retrieval date: date).

References can be numbered and listed according to the order used in the thesis or alphabetically by the last name of the first author.

When to use References

Statements which has not been worked out by the student should be supported by a reference. Figures and tables that have not been created independently or have been redrawn must be referenced accordingly. Scientific publications and textbooks are valid references, while Lecture notes and Wikipedia are not citable. Websites are only conditionally acceptable as sources.

How to find literature

https://scholar.google.de/

https://pubmed.ncbi.nlm.nih.gov/

Library

Valid Literature

Papers

Books

Plagiarism

Every statement without an reference indicates, that this is your idea and work.
Obvious plagiats are a reason to fail the thesis.

1. Introduction -> Some references
2. Background -> A lot of references
3. Material and Methods -> References for software packages, datasets
4. Results -> Some references or no references at all
5. Discussion -> A lot of references
6. Future Perspectives -> Some references or no references at all
7. Conclusion -> Some references or no references at all

Figures

How to include Figures

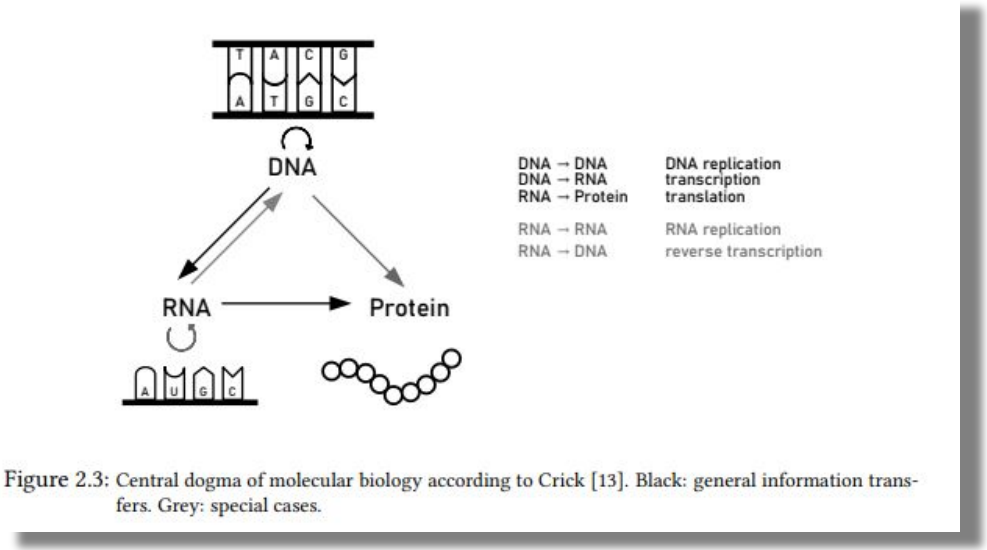


Figure 2.3: Central dogma of molecular biology according to Crick [13]. Black: general information transfers. Grey: special cases.

Löchel HF. Fractals in Bioinformatics: Visualization, Analysis, and Construction of Molecular Codes. University Marburg. 2021.

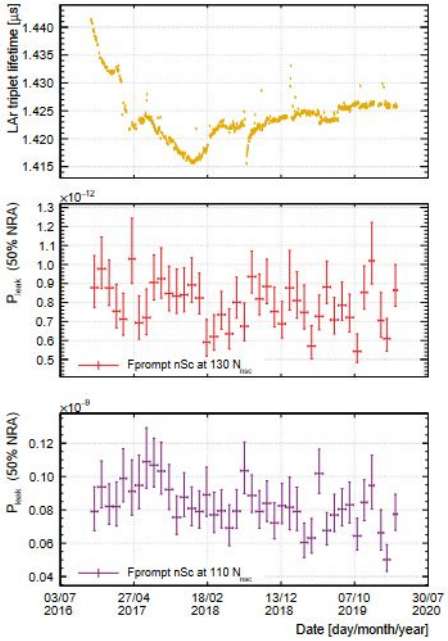


Fig. 8: The upper two figures show the time evolution of the leakage probabilities at $110\,N_{nsc}$ and $130\,N_{nsc}$ for F^{nsc}_{prompt} at 50% NRA. The triplet lifetimes in the third figure are determined by using the full mathematical model of the ^{39}Ar scintillation pulseshape [23]. The observed triplet lifetime strongly depends on the state variables of the detector, e.g. temperature, pressure, or impurities diffused from the detector into the liquid argon. A change in the triplet lifetime is expected to influence the shape of the PP distribution and hence causes the PP distribution of the whole data set to be a superposition of distributions with slightly different parameters.

The DEAP Collaboration. Pulse-shape discrimination against low-energy Ar-39 beta decays in liquid argon with 4.5 tonne-years of DEAP-3600 data. The European Physical Journal C. 2021. DOI:10.1140/epjc/s10052-021-09514-w

How to reference figures

Figure 2.3, is illustrating this principle. In black are the general ways, and in gray are the special transfers [13]. Replication, the information transfer from DNA to DNA, takes place in cell division. Transcription (DNA to RNA) and translation (RNA to Protein) are parts of protein biosynthesis. Moreover, he noticed that, in some special cases, a transfer of information takes place by RNA replication (RNA to RNA), or reverse transcription (RNA to DNA), for some virus-infected cells, and *in vitro* from DNA to protein. Thus he postulated the dogma for the unknown ways: protein to protein, protein to RNA, protein to DNA.

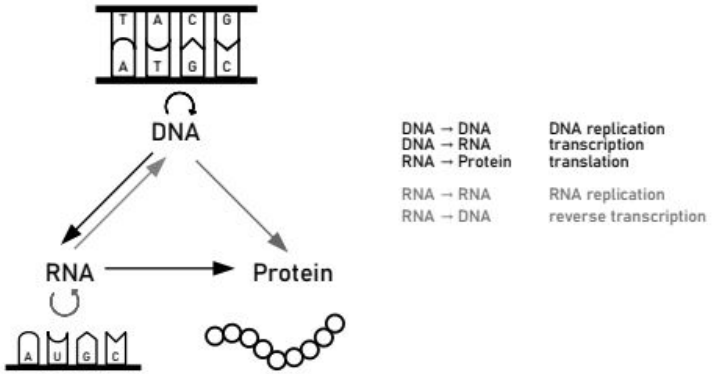


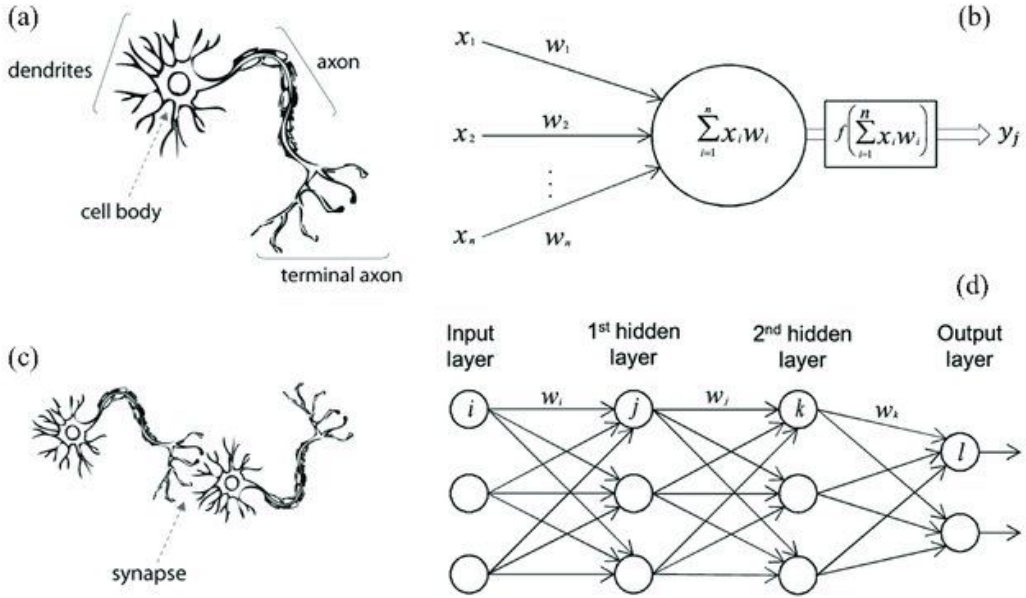
Figure 2.3: Central dogma of molecular biology according to Crick [13]. Black: general information transfers. Grey: special cases.

An image must always be explained in the text!

Löchel HF. Fractals in Bioinformatics: Visualization, Analysis, and Construction of Molecular Codes. University Marburg. 2021.

When to use figures

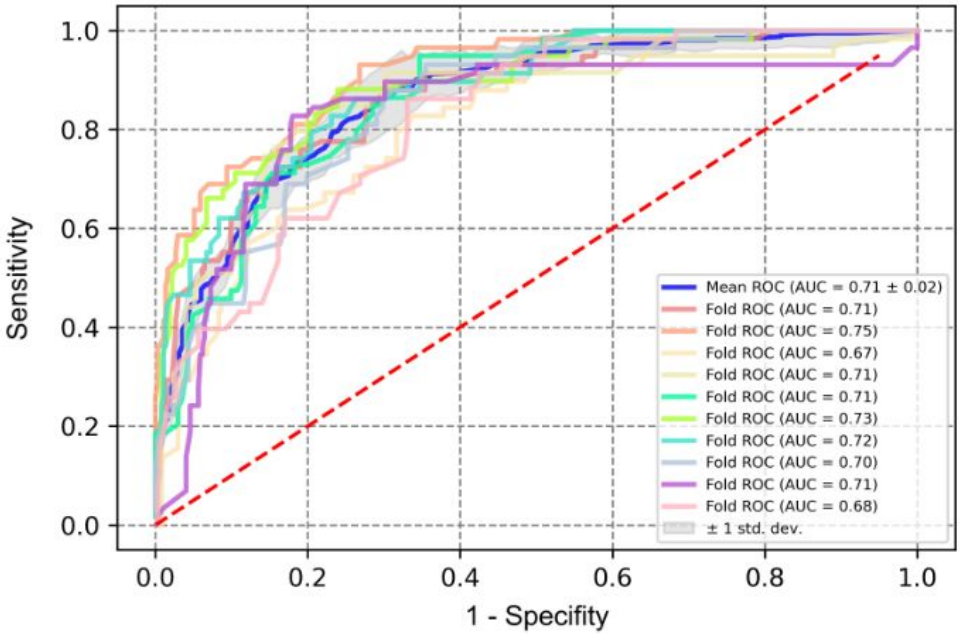
Figures can help to understand abstract concepts



Meng et al. Using a Data Driven Approach to Predict Waves Generated by Gravity Driven Mass Flows. Water. 2020. DOI:10.3390/w12020600

When to use figures

Figures might be results that you have to show



- Don’t forget to label x- and y-axis
- Make sure that the figure contributes to the understanding of your text and don’t use figures to fill space

Plagiarism of figures

DOKUMENTATION EINES PLAGIATS

Frau / Herr Matrikelnr.:

hat im Modul/in der Modulveranstaltung:

des Studiengangs:

unter der Leitung von:

folgende Studienleistung / Prüfungsleistung eingereicht:

Nach Durchsicht der vorgelegten Arbeit muss ich Ihnen leider mitteilen, dass diese aufgrund des begründeten Plagiatsvorwurfs mit ‚0 Punkten' bewertet wird!

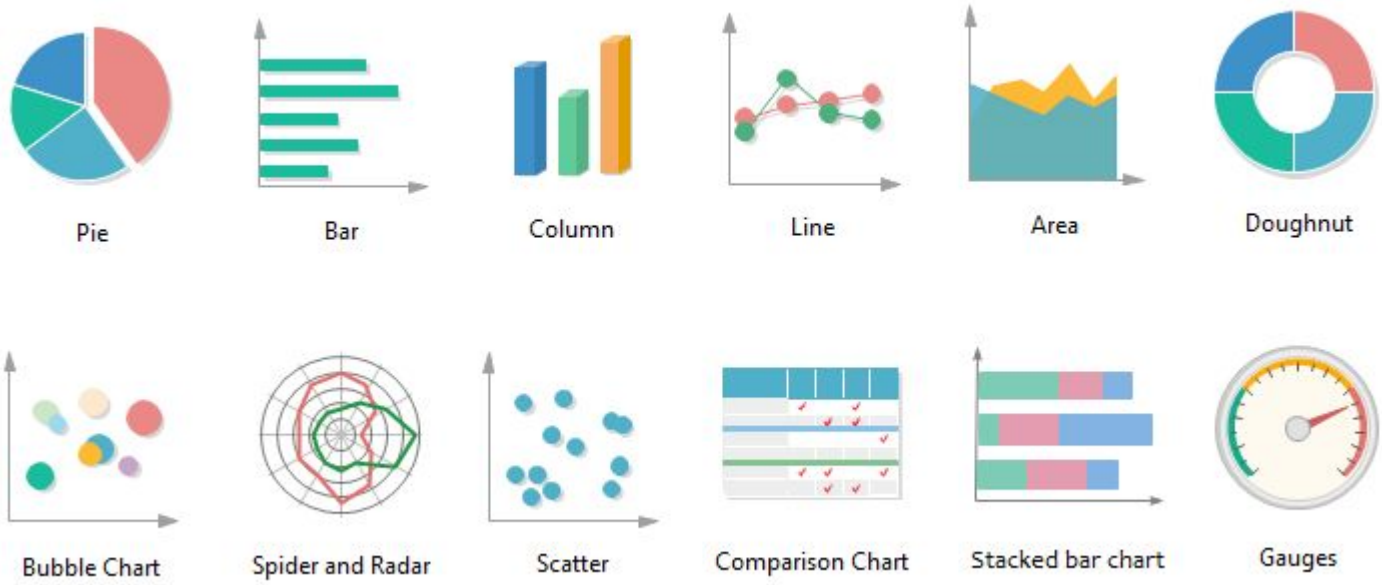
Dieser Täuschungsversuch wird im elektronischen System entsprechend verbucht und als ‚Täuschung' (TA) gekennzeichnet.

Hiermit teilen wir Ihnen mit, dass jeder weitere Täuschungsversuch gem. § 17, Abs. 3 (alte AGB), § 27, Abs. 3 (neue AGB) sowie § 18, Abs. 4, 4-5 der AGB für das Lehramt an Gymnasien Sie von weiteren Prüfungsleistungen ausschließt, sodass der Prüfungsanspruch im Studiengang erlischt!

Datum Unterschrift der Prüferin/des Prüfers

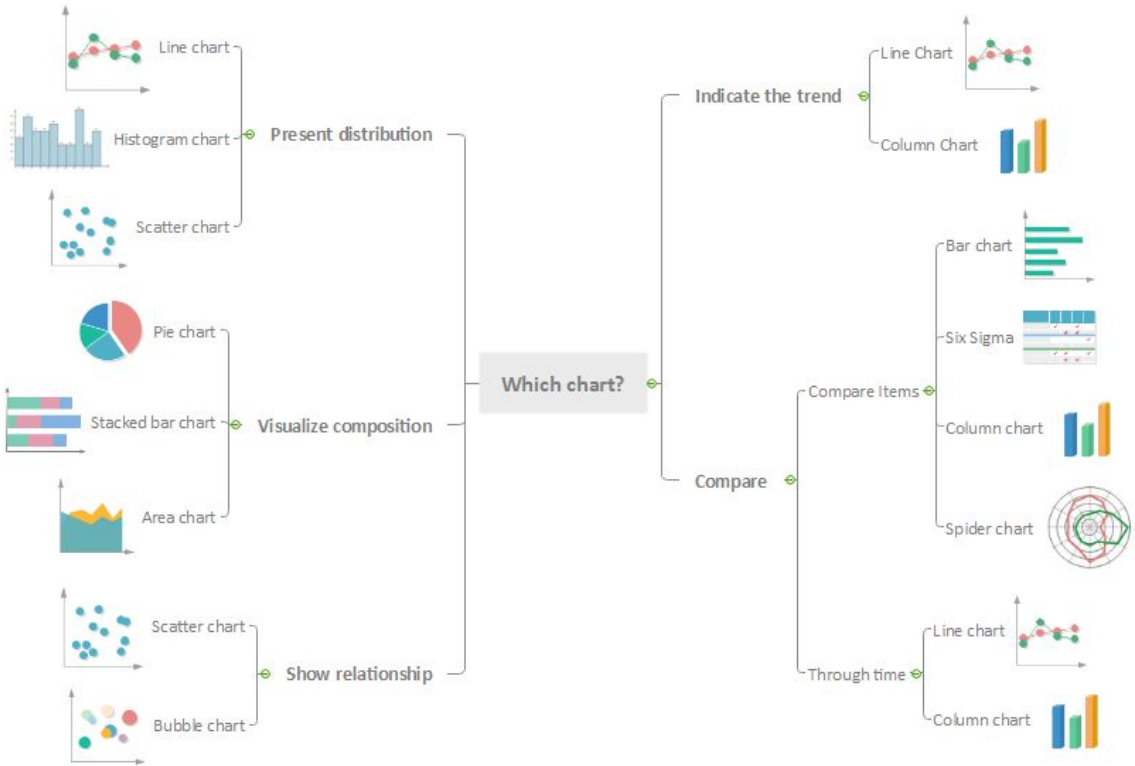
https://www.youtube.com/watch?v=8J3p7TN11Xg

Charts



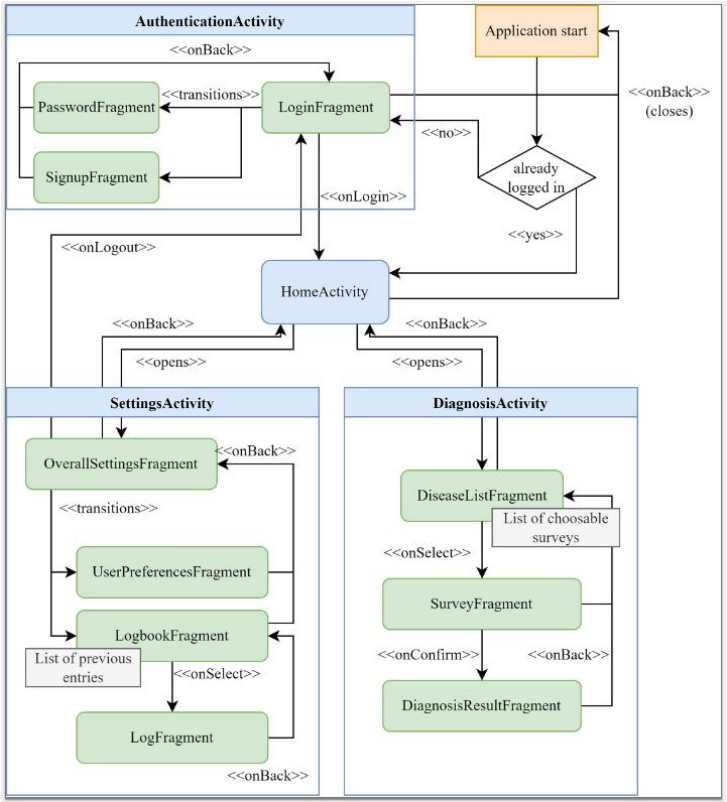
https://www.edrawsoft.com/chart/choose-right-chart.html

Charts



https://www.edrawsoft.com/chart/choose-right-chart.html

Flowcharts and UMLs



L. G. Reuter. App-basierte medizinische Datenerhebung und Evaluierung zur Unterstützung ärztlicher Diagnosen. University Marburg. 2023.

Tools for plotting figures

jupyter **Diabetes** Last Checkpoint: 19.09.2022 (autosaved) Logout

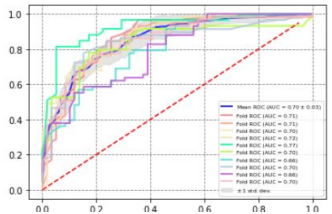
File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)

Run Code

```
mean_auc = np.mean(foldAUC)

std_tpr=np.std(tprs, axis=0)
stepperRed=np.arange(0,1,0.05)
```

In [18]:

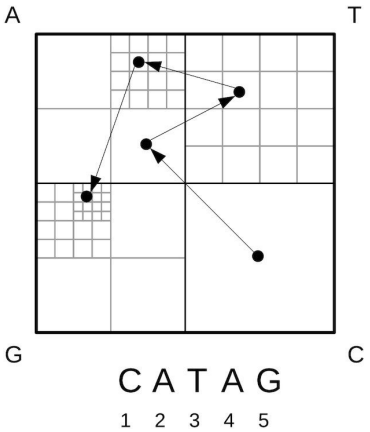
```
fig, ax = plt.subplots()

colours = ['lightcoral', 'lightsalmon', 'moccasin', 'palegoldenrod',
           'mediumspringgreen', 'greenyellow', 'turquoise', 'lightsteelblue', 'mediumorchid', 'lightpink']
ax.plot(mean_fpr, mean_tpr,
        color="b", label=r"Mean ROC (AUC = %0.2f $\pm$ %0.2f)" % (mean_auc, std_auc), lw=2, alpha=0.8)

for i,foldROC in enumerate(fprs):
    ax.plot(foldROC, tprs[i],
        color=colours[i], label=r"Fold ROC (AUC = %0.2f)" % (foldAUC[i]), lw=2, alpha=0.8)


ax.plot(stepperRed, stepperRed, color="red", ls="--")
tprs_upper = np.minimum(interpo_mean_tpr + std_tpr, 1)
tprs_lower = np.maximum(interpo_mean_tpr - std_tpr, 0)
ax.fill_between(interpo, tprs_lower, tprs_upper, color="grey", alpha=0.2, label=r"$\pm$ 1 std. dev.")


ax.grid(color='gray', linestyle='dashed')

ax.set(xlim=[-0.05, 1.05], ylim=[-0.05, 1.05])
ax.legend(loc="lower right", fontsize=6)
plt.savefig('ROCs.png', dpi=600)
```



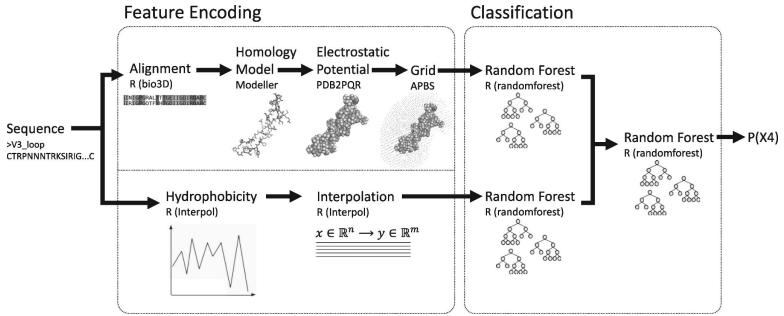
The Jupyter notebook provides a great user interface and can be used with Python, Julia, R and many more.

Tools for drawing figures

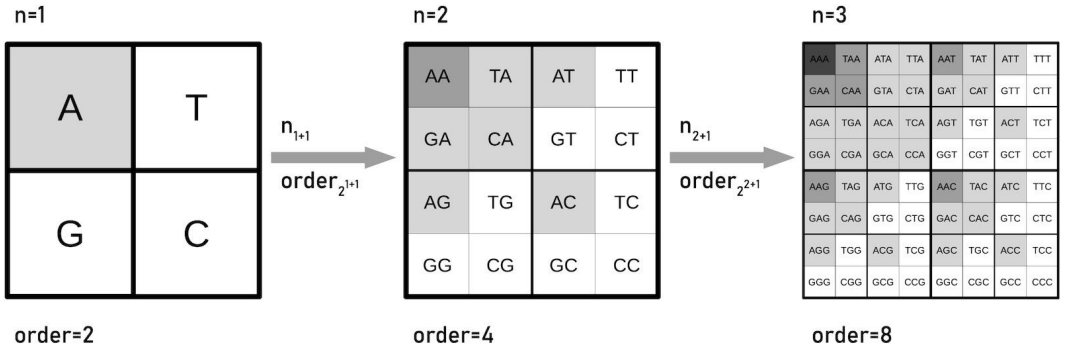
- Inkscape
- Gimp
- Libre Office Draw



Löchel, Hannah F., et al. "Fractal construction of constrained code words for DNA storage systems." *Nucleic Acids Research* 50.5 (2022): e30-e30.



Löchel, Hannah F., et al. "SCOTCH: subtype A coreceptor tropism classification in HIV-1." *Bioinformatics* 34.15 (2018): 2575-2580.



Löchel, Hannah F., et al. "Fractal construction of constrained code words for DNA storage systems." *Nucleic Acids Research* 50.5 (2022): e30-e30.

Tables

When to use tables

- Instead of bullet points
- Used material/Software
- Comparisons
- -> tables often give a better overview

How to use tables

frame. Table 2.1 shows the three NGS platforms investigated in this thesis regarding the applied technique and the read length (in base pairs (bp) or kilo bases (kb)).

Table 2.1: Next-generation sequencing technologies

Platform	**Illumina**	**Ion Torrent**	**Pacific Biosince**
Method	Sequencing by synthesis [31]	Release of H^+ by integration of dNTPs [32]	Real time sequencing by fluorescent labeled dNTPs [33]
Read length [34]	150–300 bp (paired end)	up to 40 kb (single end or circular consensus)	200–400 bp (single end)

Löchel HF. Fractals in Bioinformatics: Visualization, Analysis, and Construction of Molecular Codes. University Marburg. 2021.

Design of a table

- Table heading
- Reference tables in the text
- Be careful with table frames
- Cells should not contain too much text

Table 1.4. Cross-correlation Fit Details

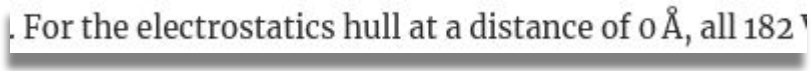
QSO Division	R Range (Mpc/h)	$\langle f \rangle$	r_0	γ	W	Separation (%)	Result Strength
1/3 Bright 2/3 Dim	[0.3,3]	$4.24 \cdot 10^{-4}$ $4.26 \cdot 10^{-4}$	6.19 4.48	1.77	96.97 52.77	96.7	1.9σ

http://jessresearch.blogspot.com/2012/07/making-awesome-latex-tables.html

Units and equations

Units

- Use SI units or units which are based on SI
- In most cases a number has a unit
- Between number and unit has to be a protected whitespace (in Latex: 10~km), to avoid linebreaks



Löchel, Hannah F., et al. "SCOTCH: subtype A coreceptor tropism classification in HIV-1." *Bioinformatics* 34.15 (2018): 2575-2580.

Equations

all variables have to be declared,

equations are numbers and not included in the text, but explained in the text

The formal definition of a fractal is that its Hausdorff's dimension exceeds the topological space [59]. In other words, the dimension D of a fractal is not represented by an integer but by a fraction. The dimension D of a fractal (or any geometrical object) A can be calculated by the box-counting theorem [63, 64]:

$$D(A) = \lim_{\epsilon \to 0} \frac{logN(A)}{log(\frac{1}{\epsilon})} \tag{2.1}$$

where: D = dimension
A = area of an object
N(A) = number of sets, covered by a grid

The underlining idea is to cover the object A with a grid. The grid is divided by the factor $\frac{1}{2}$ to the smallest diameter ϵ. The smallest number of sets, covered by a grid with the diameter close to ϵ, is denoted as $N(A)$.

Löchel HF. Fractals in Bioinformatics: Visualization, Analysis, and Construction of Molecular Codes. University Marburg. 2021.

Source code and Listings

Source Code

```
sealed class Disease(
    internal val identifier: String,
    internal val layoutID: Int
    internal val model: Class<out Model>
) : Serializable {

    abstract fun getDiseaseName(context: Context): String
    abstract fun getDiseaseDescription(context: Context): String

    //TODO add new diseases here
    internal class Diabetes(context: Context) : Disease(
        diseaseIdentifier = T2Diabetes::class.java.name,
        layoutID = R.layout.Fragment_survey_diabetes,
        model = T2DiabetesModel::class.java
    ) {

        override fun getDiseaseName(context: Context): String {
            return context.getString(R.string.disease_diabetes)
        }
        override fun getDiseaseDescription(context: Context): String {
            return context.getString(R.string.disease_diabetes_description)
        }
    }

}
```

Try to avoid using source code and if you really think it would be helpful to show source code, first talk to your supervisor!

L. G. Reuter. App-basierte medizinische Datenerhebung und Evaluierung zur Unterstützung ärztlicher Diagnosen. University Marburg. 2023.

Source Code

Explain new algorithms in pseudocode

Algorithm 1 Tiling

```
1: function TILING(A,B)
2:     res ← dimension of B
3:     for i=0; i < res; i++ do
4:         for j = 0; j < res; j++ do
5:             if B[i][j]! = 0 then
6:                 A[i][j ] ← A[i][j ] + B[i][j]
7:                 A[i+res][j ] ← A[i+res][j ] + B[i][j]
8:                 A[i][j+res] ← A[i][j+res] + B[i][j]
9:                 A[i +res][j +res] ← A[i +res][j +res] + B[i][j]
       return A
```

Löchel, Hannah F., et al. "Fractal construction of constrained code words for DNA storage systems." *Nucleic Acids Research* 50.5 (2022): e30-e30.

Language and Expression

Language and Expression

On the 1st of December 1819, the first random opportunistic symbiosis based on mutualism between four animals was reported by two individuals. According to the witnesses, a *felis catus*, a *canis lupus familiaris*, a *gallus gallus domesticus* and an equus *asinus* adopted Batesian mimicry to scare at least four *homo sapiens*. The animals climbed on top of each other in height descending order and imitated the appearance of a predator. Additionally, all four animals seemed to perform a mutually arranged vocal shriek which frightened all those present [1].

Language and Expression

On the 1st of December 1819, the **first** random opportunistic symbiosis based on mutualism between four animals was reported by **two individuals**. According to the witnesses, a *Felis catus*, a *Canis lupus familiaris*, a *Gallus gallus domesticus* and an Equus *asinus* adopted Batesian mimicry to scare at least four *Homo sapiens*. The animals climbed on top of each other in height descending order and imitated the appearance of a predator. Additionally, all four animals seemed to perform a mutually arranged vocal shriek which frightened all those present [1].

Language and Expression

Avoid expressions that can be found in fairy tales:

- Once upon a time, a long time ago
- from that time forward, always

https://de.wikipedia.org/wiki/Die_Bremer_Stadtmusikanten#/media/Datei:Stadtmusikanten_Bremen.png

Language and Expression

Avoid expressions that can be found in fairy tales:

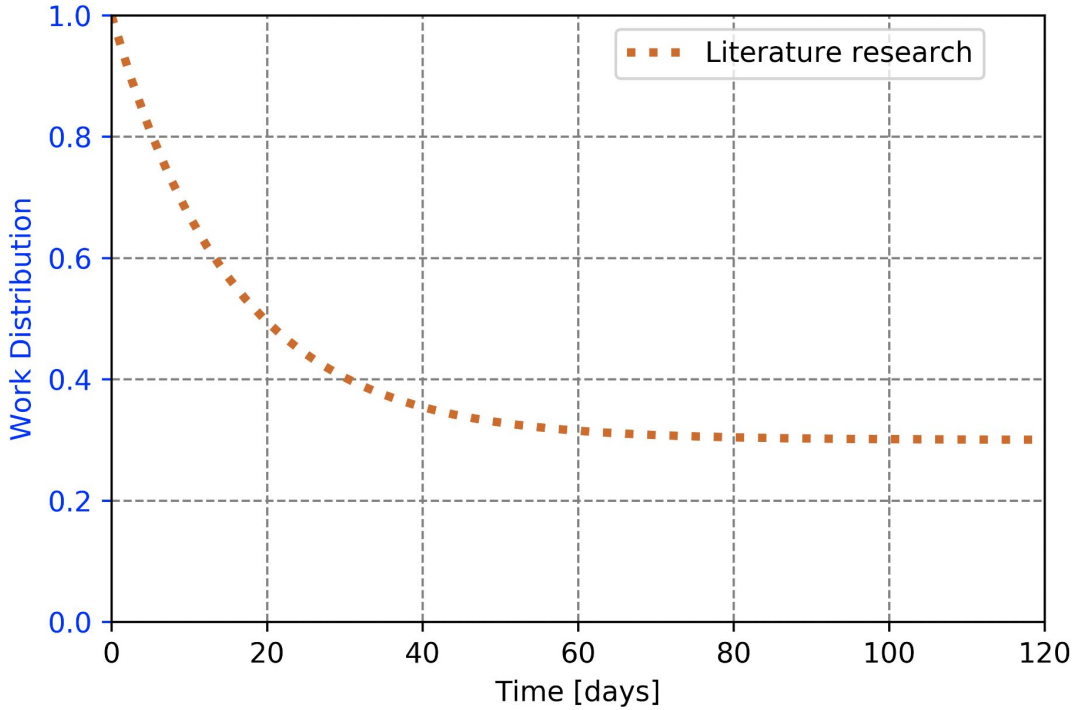
- Once upon a time, a long time ago
- from that time forward, always

Be precise and follow the cause-effect scheme!

You are responsible for the burden of proof!

https://makeameme.org/meme/evidence-proof-facts

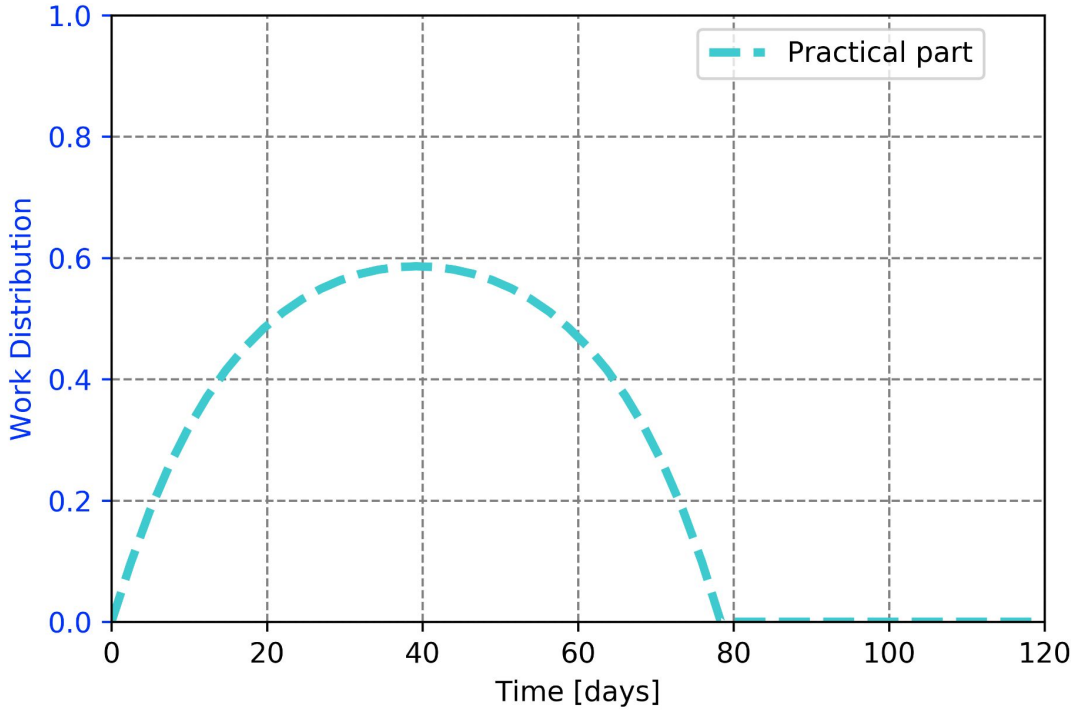
Time management



- Understand the problem and read the important literature to get ideas and anticipate the outcome of your research

- Keep up to date and compare your results with other literature

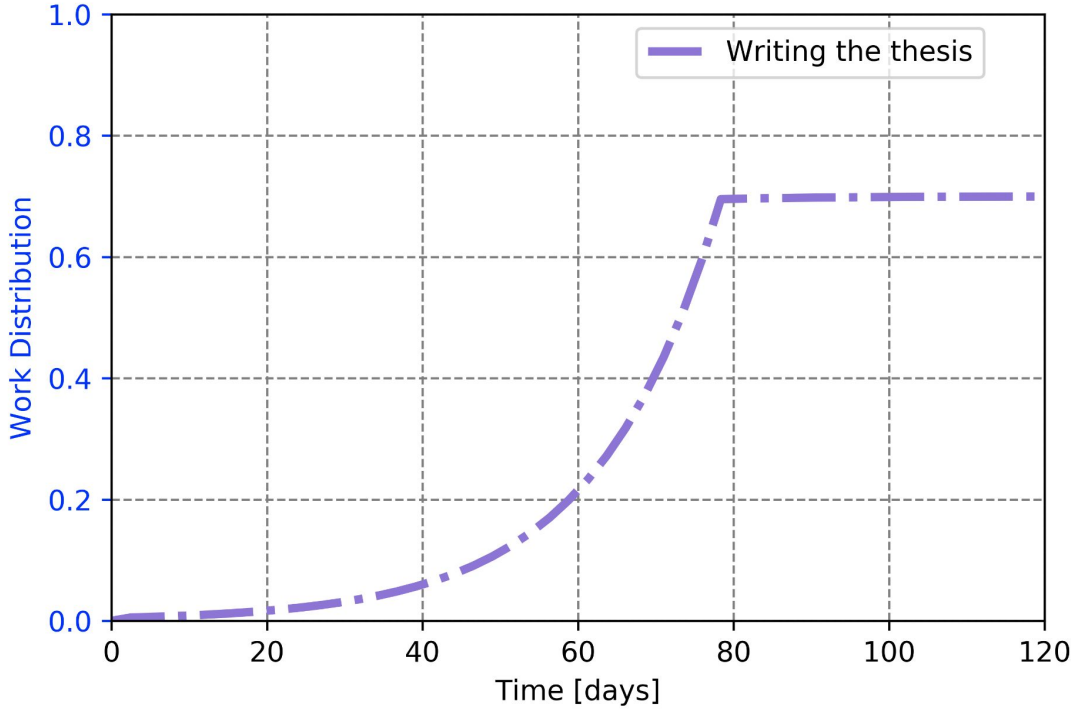
Time management



- Start solving the problem

- Keep cool if things don’t work out and meet with your supervisor regularly

Time management

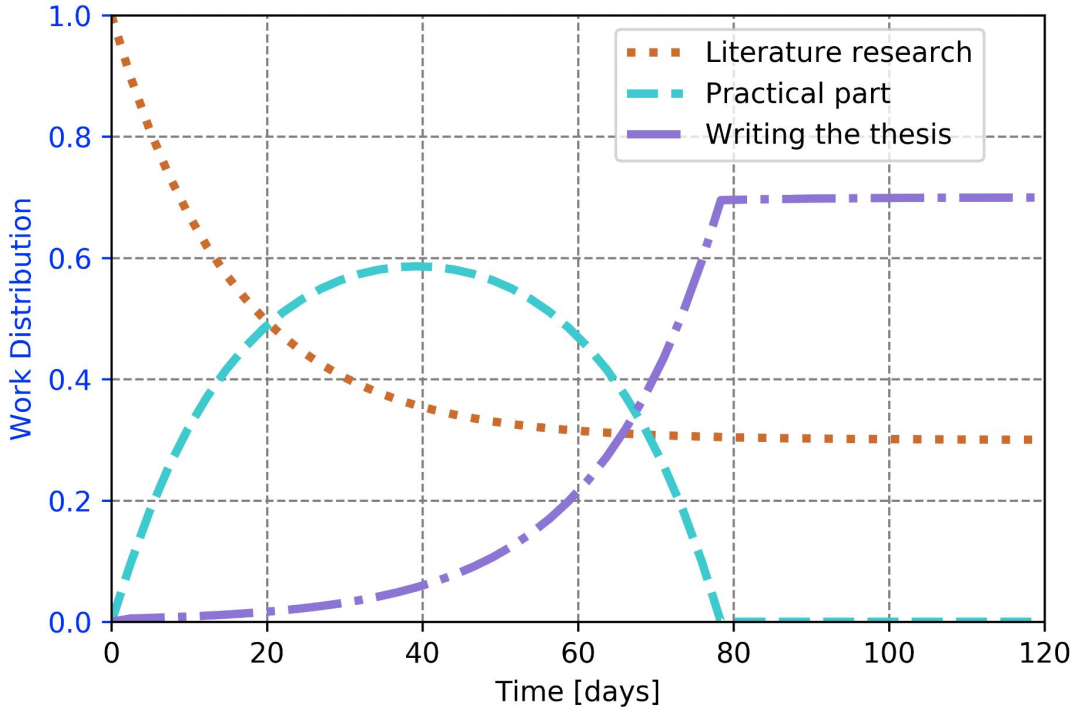


- Don’t underestimate the writing part

- If something is unclear, talk to your supervisor

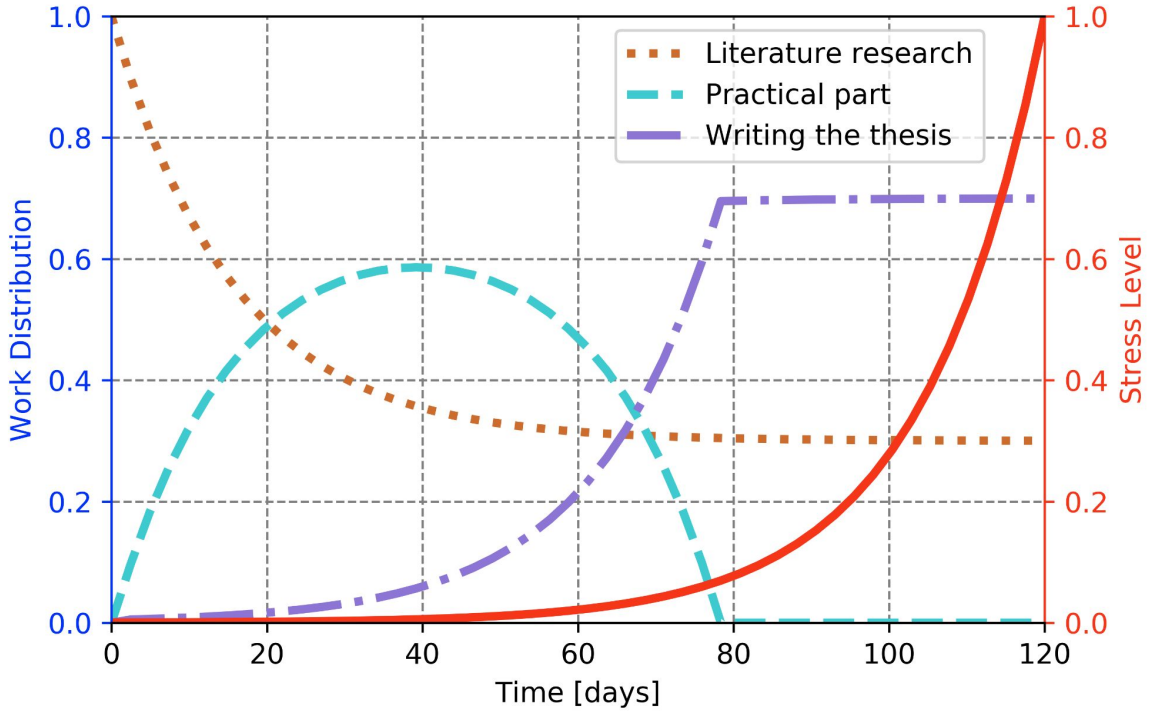
- Keep track of your deadlines

Time management

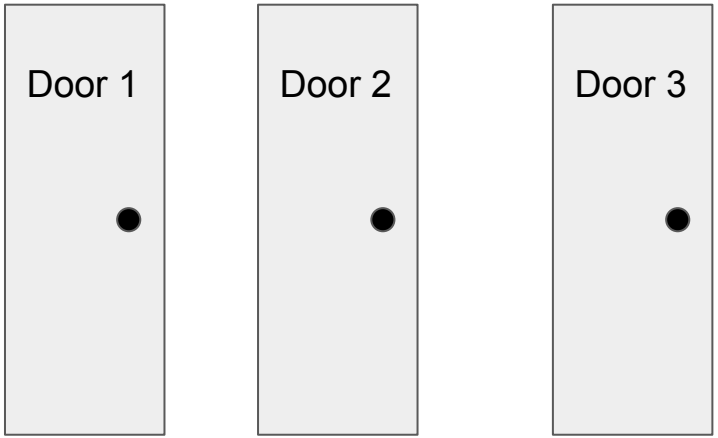


- Avoid spending too much time on problems and ask for help

Time management

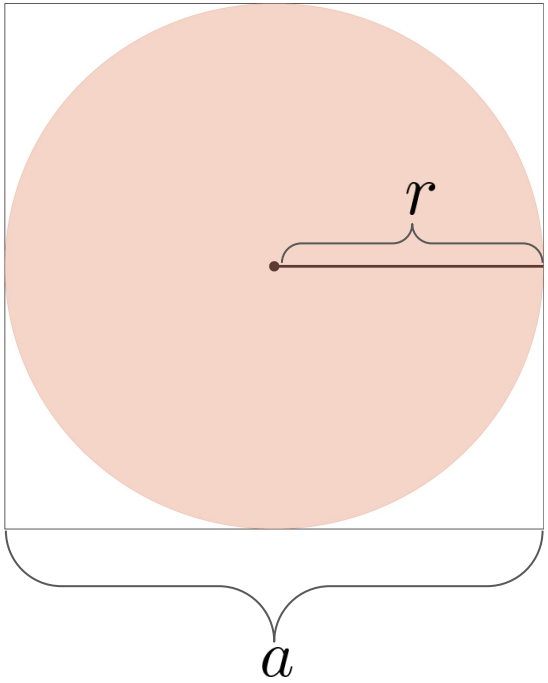


Example: Monte Carlo meets Monty Hall



https://www.ft.com/content/52eaf4be-a9ae-11e7-93c5-648314d2c72c

Example: Monte Carlo meets Monty Hall



Approximation of Pi using Monte Carlo

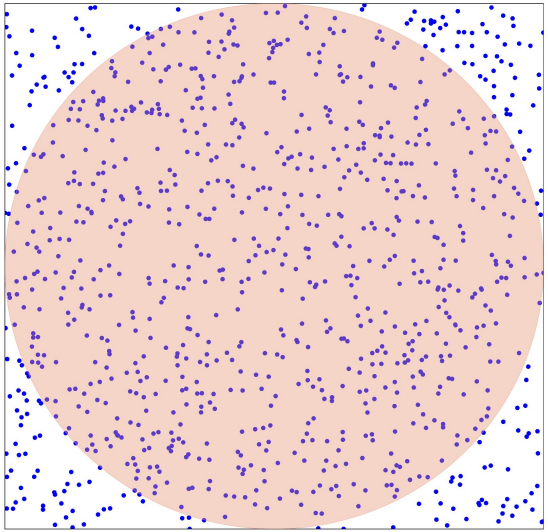
Randomly drop rice onto the square and count how many rice grains land inside the circle

$$\rightarrow F \sim \frac{\pi r^2}{a^2}$$

$$\rightarrow r = \frac{1}{2}a$$

$$\pi = 4 \cdot F$$

Example: Monte Carlo meets Monty Hall



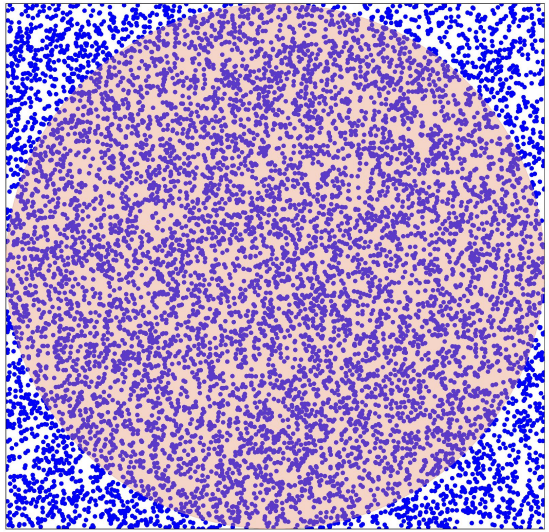
Approximation of Pi using Monte Carlo

Randomly drop rice onto the square and count how many rice grains land inside the circle

1000 grains of rice

$$\pi = 3.192$$

Example: Monte Carlo meets Monty Hall



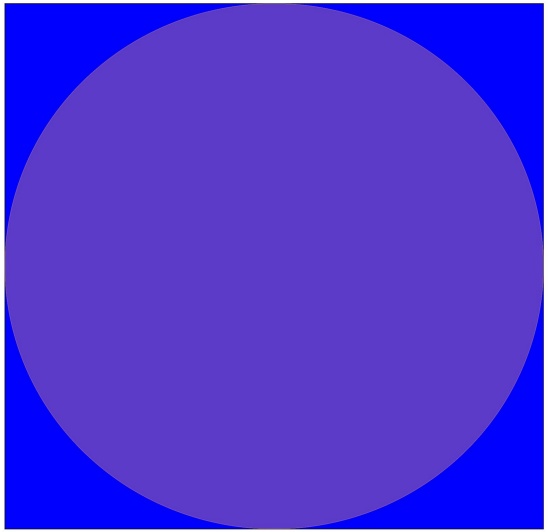
Approximation of Pi using Monte Carlo

Randomly drop rice onto the square and count how many rice grains land inside the circle

10000 grains of rice

$$\pi = 3.167$$

Example: Monte Carlo meets Monty Hall



Approximation of Pi using Monte Carlo

Randomly drop rice onto the square and count how many rice grains land inside the circle

1000000 grains of rice

$$\pi = 3.141$$

Don’t use University Marburg Logo

Philipps-Universität
Marburg

Fakultät für Mathematik und Informatik
Data Science in Biomedicine

Monte Carlo Verification of the Monty Hall Problem using Linear Congruential Generators

Machine Learning

Hannah Franziska Löchel, Jan Ruhland

April 28, 2023

Contents

1 Introduction

The Monty Hall problem is named after the host of the game show "Let's make a Deal" and was first scientifically discussed in the "Ask Marilyn" column [1]. Since the publication a lot of mathematicians have discussed the problem from different perspectives such as Bayesian interpretation or causal inference [2, 3, 4].

The problem describes the interaction between the player and the game master, resulting in the player's dilemma. The player is given the choice to open one of three doors. Behind one door is the win and behind the other two doors are losses. After the player has chosen one door, the game master will show one door with a loss behind it which isn't the door chosen by the player. The player is now faced with the dilemma of switching the door or staying with the door he had chosen in the beginning. The problem can be solved using conditional probabilities on the three events: the player's door chosen by the game master, and the winning door. Theoretical calculations show that the player doubles the chances of winning by switching the door [2, 3, 4].

The interpretation of probabilities always depends on the underlying paradigm. The use of the Bayesian or frequentist paradigm is philosophical in nature, yet, depending on the problem, one method might be favored [5]. The first solution to the Monty Hall problem used the Bayesian approach and in this report, a Monte Carlo simulation of the player's dilemma is used to verify the Bayesian results.

Monte Carlo simulations describe the repetitions of experiments to infer statistical properties of a system [6]. The first attempt at a computational Monte Carlo simulation was attributed to the theoretical physicist Enrico Fermi in 1930. The computing power at that time limited early attempts at Monte Carlo simulations. Due to the increase of commercially available high-performance computing services, Monte Carlo simulations are standard practice in modern research [7].

The aim of the thesis is to implement a Monte Carlo simulation of the Monty Hall problem in Python based on rudimentary operations and Python modules. The theoretical results should be derivable by only the simulation and hence show a modern approach to solving complex statistical problems.

Motivate Monty Hall

Explaining Monty Hall

Transition to probabilistic interpretation

Explaining Monte Carlo simulations as an alternative to theoretical calculations

Explaining goal of the thesis

2 Background

One of the first methods to generate pseudorandom numbers is given by equation (2.1) [8]. The equation describes a recursive function defined on the interval $[0,m]$. a, c, $m \in \mathbb{N}$ are parameters that define the period of the recursive function as well as the number of repeating patterns.

$$x_i = (a \cdot x_{i-1} + c) \mod m \tag{2.1}$$

The parameters a, c and m have to be chosen to obtain a full period with a minimal number of repeating patterns. A full period can only be achieved if the following conditions are met [9]:

i) m and c are coprime

ii) each prime divisor p of m divides $(a-1)$

iii) if 4 divides m, then 4 divides $(a-1)$

Repeating patterns are unavoidable but the frequency can be reduced by additional conditions for the parameters a, c and m [10]. The parameters given in table 2.1 fulfill these conditions and are used in other software applications [10].

Table 2.1: The table shows the values for the parameters a, c and m of the linear congruential generator taken from [10].

a	c	m
1664525	1013904223	2^{32}

The theoretical foundation of the Monte Carlo simulation lies within the law of large numbers. The theorem states that the normed sum of independently and identically distributed random variables X_i converges to the sample mean as described in equation (2.2) [11].

$$\mathbb{E}[X] = \lim_{N\to\infty} \frac{1}{N} \sum_{i=1}^{N} X_i \tag{2.2}$$

Defining congruential linear generator

Defining law of large numbers

3 Materials and Methods

The Monte Carlo simulation was developed in the programming language Python. Due to the rudimentary implementation, the only additional module was Matplotlib for plotting and the corresponding versions of the software components are listed in Table 3.1.

Table 3.1: Software components used for the implementation of the Monte Carlo simulation of the Monty Hall Problem.

Software	Version
Python	3.7.6
Matplotlib	3.1.3

Software modules

For the Monte Carlo simulation we needed three different random numbers representing the events P (door picked by player), W (winning door), and G (door revealed by the game master). Each event was defined by equation (2.1) with a different seed value. The parameter values are provided in table 2.1 and the seed values are listed in Table 3.2.

Table 3.2: The seed values for the different random number generators which represent the events P, W and G.

Event	seed
Player (P)	476756
Win (W)	1076
Game master (G)	90213

First, we drew the two random numbers for the events P and W. The interval $[0, 2^{32}-1]$ was subdivided into three disjoint and equidistant intervals. The three intervals represented the three doors, e.g. if a generated random number lands in the first interval, then the first door will be picked. If the doors for P and W are the same, then we have to generate the third random number for event G. For event G, the whole interval was subdivided into two disjoint and equidistant intervals corresponding to the remaining two doors. In the simulation, we also considered the case, that the player chooses one of the two remaining doors randomly. The player's random choice was implemented analogously to the event that the game master has to pick one of two doors.

Implementation

3 Materials and Methods

The Monte Carlo simulation was developed in the programming language Python. Due to the rudimentary implementation, the only additional module was Matplotlib for plotting and the corresponding versions of the software components are listed in Table 3.1.

Table 3.1: Software components used for the implementation of the Monte Carlo simulation of the Monty Hall Problem.

Software	Version
Python	3.7.6
Matplotlib	3.1.3

For the Monte Carlo simulation we needed three different random numbers representing the events P (door picked by player), W (winning door), and G (door revealed by the game master). Each event was defined by equation (2.1) with a different seed value. The parameter values are provided in table 2.1 and the seed values are listed in Table 3.2.

Table 3.2: The seed values for the different random number generators which represent the events P, W and G.

Event	seed
Player (P)	476756
Win (W)	1076
Game master (G)	90213

First, we drew the two random numbers for the events P and W. The interval $[0, 2^{32}-1]$ was subdivided into three disjoint and equidistant intervals. The three intervals represented the three doors, e.g. if a generated random number lands in the first interval, then the first door will be picked. If the doors for P and W are the same, then we have to generate the third random number for event G. For event G, the whole interval was subdivided into two disjoint and equidistant intervals corresponding to the remaining two doors. In the simulation, we also considered the case, that the player chooses one of the two remaining doors randomly. The player's random choice was implemented analogously to the event that the game master has to pick one of two doors.

Implementation of the simulation

3 Materials and Methods

The Monte Carlo simulation was developed in the programming language Python. Due to the rudimentary implementation, the only additional module was Matplotlib for plotting and the corresponding versions of the software components are listed in Table 3.1.

Table 3.1: Software components used for the implementation of the Monte Carlo simulation of the Monty Hall Problem.

Software	Version
Python	3.7.6
Matplotlib	3.1.3

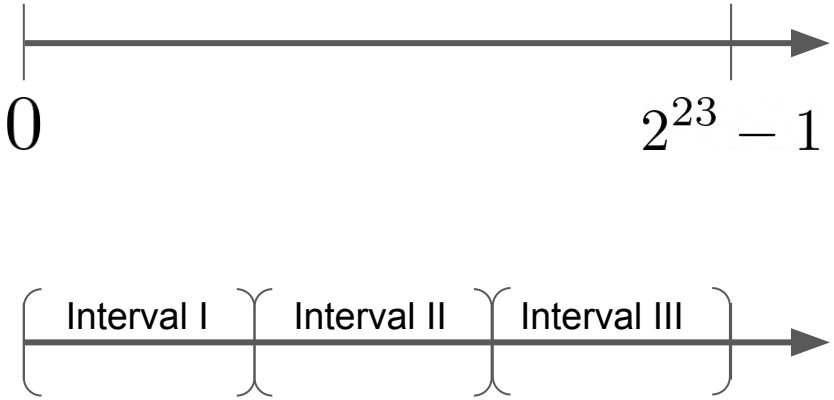
For the Monte Carlo simulation we needed three different random numbers representing the events P (door picked by player), W (winning door), and G (door revealed by the game master). Each event was defined by equation (2.1) with a different seed value. The parameter values are provided in table 2.1 and the seed values are listed in Table 3.2.

Table 3.2: The seed values for the different random number generators which represent the events P, W and G.

Event	seed
Player (P)	476756
Win (W)	1076
Game master (G)	90213

First, we drew the two random numbers for the events P and W. The interval $[0, 2^{32}-1]$ was subdivided into three disjoint and equidistant intervals. The three intervals represented the three doors, e.g. if a generated random number lands in the first interval, then the first door will be picked. If the doors for P and W are the same, then we have to generate the third random number for event G. For event G, the whole interval was subdivided into two disjoint and equidistant intervals corresponding to the remaining two doors. In the simulation, we also considered the case, that the player chooses one of the two remaining doors randomly. The player's random choice was implemented analogously to the event that the game master has to pick one of two doors.

Implementation of the simulation



3 Materials and Methods

The Monte Carlo simulation was developed in the programming language Python. Due to the rudimentary implementation, the only additional module was Matplotlib for plotting and the corresponding versions of the software components are listed in Table 3.1.

Table 3.1: Software components used for the implementation of the Monte Carlo simulation of the Monty Hall Problem.

Software	Version
Python	3.7.6
Matplotlib	3.1.3

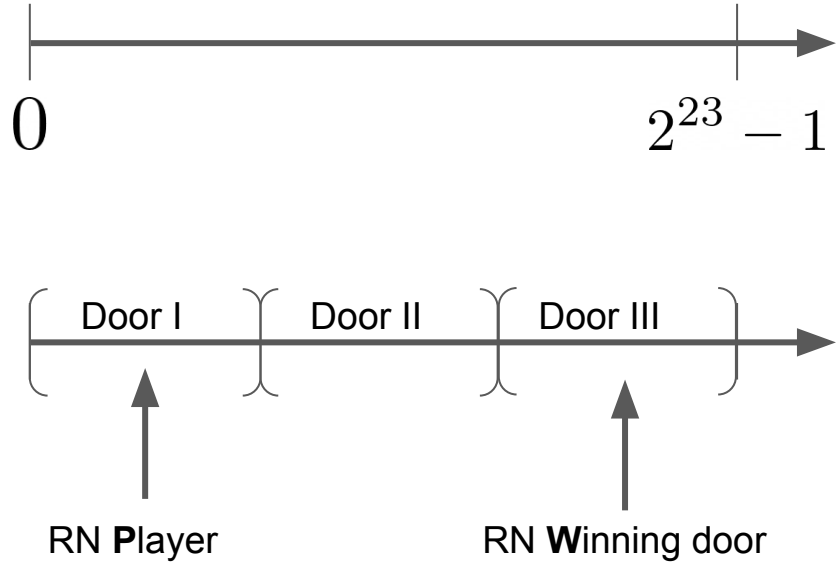
For the Monte Carlo simulation we needed three different random numbers representing the events P (door picked by player), W (winning door), and G (door revealed by the game master). Each event was defined by equation (2.1) with a different seed value. The parameter values are provided in table 2.1 and the seed values are listed in Table 3.2.

Table 3.2: The seed values for the different random number generators which represent the events P, W and G.

Event	seed
Player (P)	476756
Win (W)	1076
Game master (G)	90213

First, we drew the two random numbers for the events P and W. The interval $[0, 2^{32}-1]$ was subdivided into three disjoint and equidistant intervals. The three intervals represented the three doors, e.g. if a generated random number lands in the first interval, then the first door will be picked. If the doors for P and W are the same, then we have to generate the third random number for event G. For event G, the whole interval was subdivided into two disjoint and equidistant intervals corresponding to the remaining two doors. In the simulation, we also considered the case, that the player chooses one of the two remaining doors randomly. The player's random choice was implemented analogously to the event that the game master has to pick one of two doors.

Implementation of the simulation



3 Materials and Methods

The Monte Carlo simulation was developed in the programming language Python. Due to the rudimentary implementation, the only additional module was Matplotlib for plotting and the corresponding versions of the software components are listed in Table 3.1.

Table 3.1: Software components used for the implementation of the Monte Carlo simulation of the Monty Hall Problem.

Software	Version
Python	3.7.6
Matplotlib	3.1.3

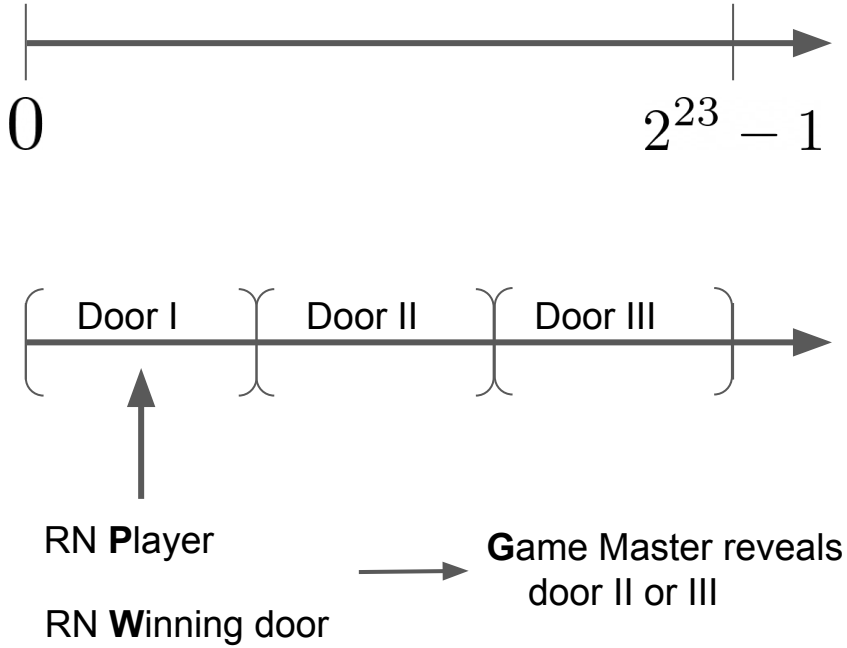
For the Monte Carlo simulation we needed three different random numbers representing the events P (door picked by player), W (winning door), and G (door revealed by the game master). Each event was defined by equation (2.1) with a different seed value. The parameter values are provided in table 2.1 and the seed values are listed in Table 3.2.

Table 3.2: The seed values for the different random number generators which represent the events P, W and G.

Event	seed
Player (P)	476756
Win (W)	1076
Game master (G)	90213

First, we drew the two random numbers for the events P and W. The interval $[0, 2^{32}-1]$ was subdivided into three disjoint and equidistant intervals. The three intervals represented the three doors, e.g. if a generated random number lands in the first interval, then the first door will be picked. If the doors for P and W are the same, then we have to generate the third random number for event G. For event G, the whole interval was subdivided into two disjoint and equidistant intervals corresponding to the remaining two doors. In the simulation, we also considered the case, that the player chooses one of the two remaining doors randomly. The player's random choice was implemented analogously to the event that the game master has to pick one of two doors.

Implementation of the simulation



3 Materials and Methods

The Monte Carlo simulation was developed in the programming language Python. Due to the rudimentary implementation, the only additional module was Matplotlib for plotting and the corresponding versions of the software components are listed in Table 3.1.

Table 3.1: Software components used for the implementation of the Monte Carlo simulation of the Monty Hall Problem.

Software	Version
Python	3.7.6
Matplotlib	3.1.3

For the Monte Carlo simulation we needed three different random numbers representing the events P (door picked by player), W (winning door), and G (door revealed by the game master). Each event was defined by equation (2.1) with a different seed value. The parameter values are provided in table 2.1 and the seed values are listed in Table 3.2.

Table 3.2: The seed values for the different random number generators which represent the events P, W and G.

Event	seed
Player (P)	476756
Win (W)	1076
Game master (G)	90213

First, we drew the two random numbers for the events P and W. The interval $[0, 2^{32}-1]$ was subdivided into three disjoint and equidistant intervals. The three intervals represented the three doors, e.g. if a generated random number lands in the first interval, then the first door will be picked. If the doors for P and W are the same, then we have to generate the third random number for event G. For event G, the whole interval was subdivided into two disjoint and equidistant intervals corresponding to the remaining two doors. In the simulation, we also considered the case, that the player chooses one of the two remaining doors randomly. The player's random choice was implemented analogously to the event that the game master has to pick one of two doors.

Implementation of the simulation

Strategies:

I. Don't change the door

II. Change the door

III. Random picking

4 Results

For the evaluation of the simulation, we simulated 10000 games and calculated the fraction of wins for every subinterval. Figure 4.1 shows the win rates with respect to the number of games for the three strategies: i) staying with the initial decision, ii) changing the door and iii) randomly choosing one of the two remaining doors. In the figure, a high fluctuation of the win rates can be observed for a small number of games. Moreover, the simulations suggest a convergence of the win rates for a sufficiently large number of games. For this case, the Monte Carlo simulation predicts the probability of winning for strategy i) to be 33.4%, for strategy ii) 66.6%, and for strategy iii) 50.0%. This is in accordance with theoretical considerations [2, 3, 4].

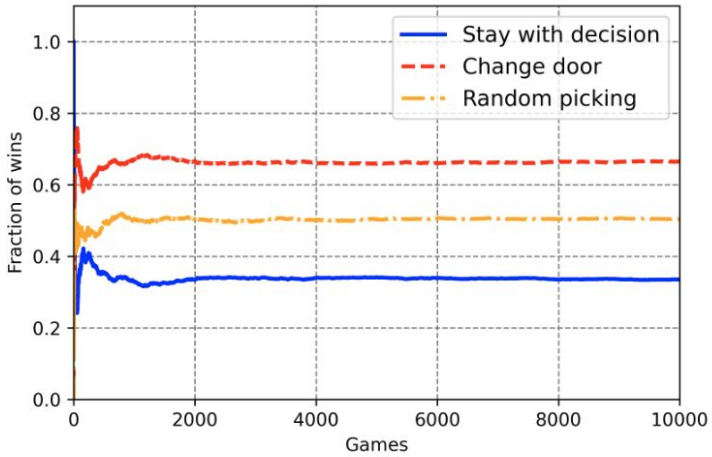


Figure 4.1: The figure shows the resulting win rates of the Monte Carlo simulation of the Monty Hall problem for 10000 games. The blue line shows the win rate if the player doesn't change the door after the game master revealed a loss door. The red line corresponds to the decision to change the door and the orange line to a random choice.

Describing/Explaining the results of the simulations

4 Results

For the evaluation of the simulation, we simulated 10000 game
of wins for every subinterval. Figure 4.1 shows the win rates
of games for the three strategies: i) staying with the initi
door and iii) randomly choosing one of the two remaining
fluctuation of the win rates can be observed for a small num
simulations suggest a convergence of the win rates for a suffici
For this case, the Monte Carlo simulation predicts the probal
i) to be 33.4%, for strategy ii) 66.6%, and for strategy iii) 5
with theoretical considerations [2, 3, 4].

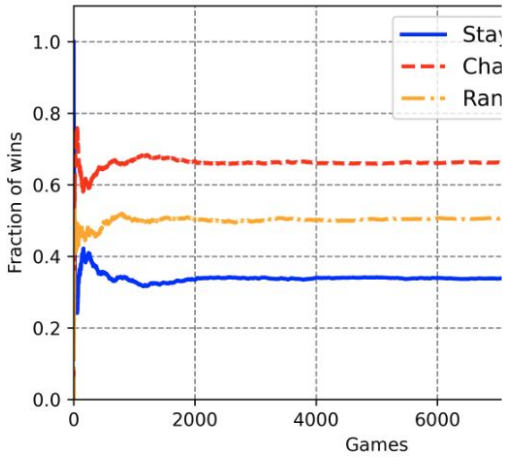
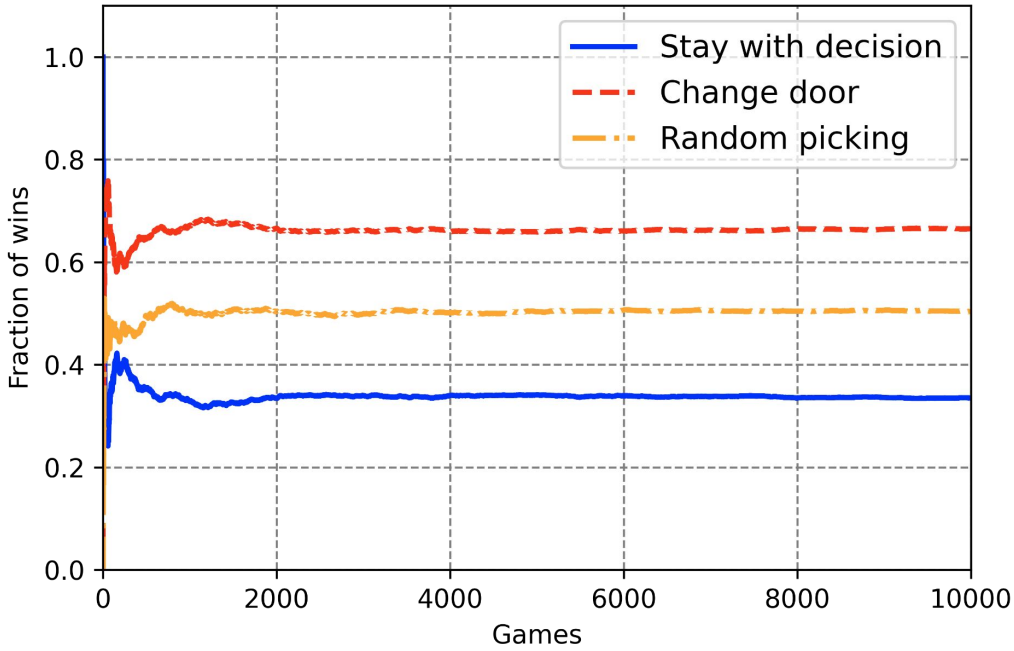


Figure 4.1: The figure shows the resulting win rates of the M
Monty Hall problem for 10000 games. The blue li
player doesn't change the door after the game mas
red line corresponds to the decision to change the
a random choice.



5 Discussion

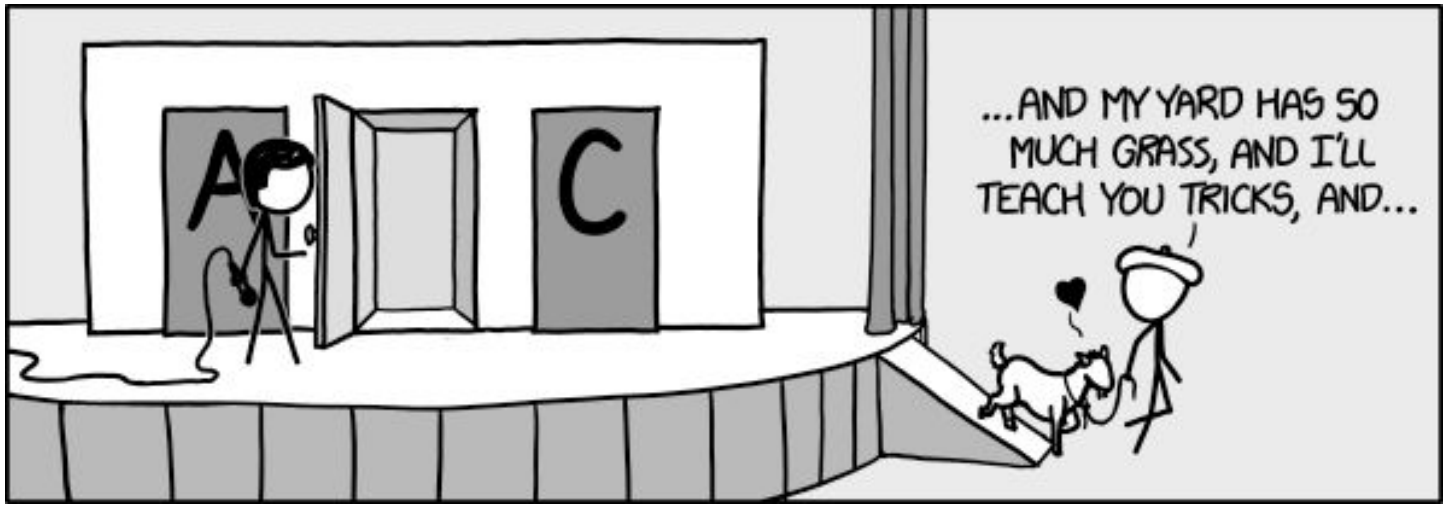
In this thesis, a Monte Carlo simulation of the Monty Hall problem was implemented in Python using a linear congruential generator. The implementation was used to simulate 10000 games and compare the probability of winning to the theoretically derived values. The observed properties of the simulation, shown in figure 4.1, can be explained by the law of large numbers given by equation (2.2). For the explicit calculation, we have to consider the probability of winning $\mathbb{P}(X = 1)$ and losing $\mathbb{P}(X = 0)$, i.e. choosing the right door given our applied strategy.

$$\mathbb{E}[X] = \lim_{N\to\infty} \frac{1}{N} \sum_{i=1}^{N} X_i = 1 \cdot \mathbb{P}(X = 1) + 0 \cdot \mathbb{P}(X = 0) = \mathbb{P}(X = 1) \tag{5.1}$$

Equation (5.1) shows explicitly that the simulated win rates converge to the expected fraction of wins and hence to the probability of winning for the three strategies. This shows that both the theoretical consideration as well as the simulation lead to the same result. Despite having the same result, the adjustment of the theoretical framework to experimental alternations is more complicated than for the Monte Carlo method.

Summary

Explaining result



https://xkcd.com/1282/