

Faculty of Natural and
Mathematical Sciences
Department of Informatics



7CCSMPRJ

Individual Project Submission 2015/16

Name: Sebastian Zillesen
Student Number: #1564629
Degree Programme: MSc Web Intelligenc
Project Title: Automated Small Datanalyst
Supervisor: Jeroen Keppens
Scientific Assistant: Isabel Sassoon
Word Count: 11287

Plagiarism Statement

All work submitted as part of the requirements for any examination or assessment must be expressed in your own words and incorporates your own ideas and judgements. Plagiarism is the taking and using of another persons thoughts, words, judgements, ideas, etc., as your own without any indication that they are those of another person.

Plagiarism is a serious examination offence. An allegation of plagiarism can result in action being taken under the *B3 Misconduct Regulations*.

I acknowledge that I have read and understood the above information and that the work I am submitting is my own.

Signature:

Date: August 25, 2016

Department of Informatics
King's College London
WC2R 2LS London
United Kingdom

Automated Small Datanalyst

Individual Project Submission 2015/16

Sebastian Zillesen

Student Number: #1564629

Course: MSc Web Intelligenc

Supervisor: Jeroen Keppens

Scientific Assistant: Isabel Sassoon



Thesis submitted as part of the requirements for the award of the MSc in Web
Intelligence.

7CCSMPRJ - MSc Individual Project - 2016

Abstract

Data collection has seen a dramatic increase over the last years and researchers agree exploiting the collected data to create data driven decision processes is crucial to generate valuable insights for clinical studies. This generates the requirement to assist clinicians during the design process of their studies with an intelligent support agent performing these analyses for them.

This project implements an existing approach to apply argumentation on this problem, by representing the statistical models and their assumptions as a statistical knowledge base and implementing the process into a user-friendly web application. The model selection is influenced by expressing preferences applying on different context domains and the close integration of the clinician and insights she/he can give related to the process. This will enable clinicians – even without a background in statistics or informatics – to answer their research questions in an appropriate way and to make evidence based decisions.

Keywords: *application of argumentation, automated statistical analysis, statistical model selection, argumentation theory, intelligent agent.*

Acknowledgements

The author would like to thank Isabel Sassoon for her generous and prompt support during this project and the enormous amount of time she spent in discussing her papers, our thoughts and the progress of the project. In addition, many thanks to Jeroen Keppens, who had the time to meet on a regular basis to discuss the progress of the project.

Contents

Glossary	iii
Acronyms	iv
List of Figures	v
List of Tables	vi
List of Listings	vi
1 Introduction	1
1.1 Project Goals and Objectives	1
1.2 Methodologies of the Project	3
1.3 Supplementary Resources	4
2 Background Research	5
2.1 Structure of the Background Research	5
2.2 Argumentation Theory: General Introduction	6
2.3 Extended Argumentation Framework - Working with Preferences in AF	9
2.4 Statistical Model Selection via Argumentation	14
2.4.1 Statistical Knowledge Base	15
2.4.2 Adding Preferences for different Context Domains to the Statistical Knowledge Base	16
3 Project Plan	21
4 Software Design Process	23
4.1 Use Cases, Use Case Flows and Use Case Slices	23
4.2 Test Driven Development	24
5 Specification	25
5.1 Weak Requirements	25
5.2 Actors in our System	25
5.3 Use Cases	26
5.4 Technical Specification	29
5.5 Data Flow in the System	30

6	Application Overview	33
6.1	General Process and Application Setup	33
6.2	User Interface and Core Functionalities	34
6.3	Example Walkthrough	36
7	Implementation Details	41
7.1	Entity Relationship Class Diagram	41
7.2	Extended Argumentation Framework: Algorithm	43
7.3	Argumentation Framework: Algorithm	43
7.4	R-Code and rinruby Gem	45
8	Evaluation	47
8.1	Observation and Feedback	47
8.2	Outlooks	47
9	Conclusion	49
	References	51
A	Appendix	57
A.1	Other Approaches for Preferences in Argumentation Frameworks	57
A.2	Sample Clinical Data Set	58
B	Software related Appendix	61
B.1	Code Samples	61
B.2	Installation Guide	63
B.3	Use Cases	63
C	Third Party Libraries	66
D	R-Spec Test Suite	68
E	Source Code	75

Glossary

actor Specifies a role played by a user or any other system that interacts with a use case in UML. [23](#), [25](#), [27](#)

context domain Different preferences are made of sets of mutually exclusive contexts and express for each available model a performance measurement related to this context [\[24\]](#). [2](#), [10](#), [14–19](#), [35–38](#), [43](#), [49](#)

end-to-end test Tests whether the flow of an application is behaving as expected. End-to-end tests ensure data integrity and usability of an application. [23](#), [24](#), [26](#)

integration test Individual software modules are combined and tested as a group to check whether a desired feature is implemented correctly. Usually two to four components are tested together to ensure that interfaces are developed in accordance to the specification. [23](#), [24](#), [26](#)

preference "Preference orders over models will arise from different sources in the context of statistical model selection" (such as the statistical theory underpinning each model, model intent and the clinicians preference) [\[24\]](#). [35](#)

product owner "The Scrum product owner is typically a project's key stakeholder. Part of the product owner responsibilities is to have a vision of what he or she wishes to build, and convey that vision to the Scrum team. This is key to successfully starting any agile software development project." [\[25\]](#). [3](#), [26](#)

R "A system for statistical computation and graphics. It consists of a language plus a run-time environment with graphics, a debugger, access to certain system functions, and the ability to run programs stored in script files." It is widely used for statistical analysis processes. [\[13\]](#). [25](#), [30](#), [41](#), [45–49](#)

unit test Involves testing the smallest testable parts of an application (unit). Unit tests can be run individually and independently and test one specific components feature set. Dependencies are stubbed or mocked out. [23](#), [24](#), [26](#)

use case "The use cases capture the goals of the system. To understand a use case we tell stories. The stories cover how to successfully achieve the goal, and how to handle any problems that may occur on the way. Use cases provide a way to identify and capture all the different but related stories in a simple but comprehensive way. This enables the systems requirements to be easily captured, shared and understood." [\[14\]](#). [3](#), [23–26](#), [50](#), [59](#)

use case flow "Use-Case Narrative that outlines its stories as a set of flows" [14]. 24
use case slice Use cases are build up out of multiple "use case slices (a slice being a carefully selected part of a use case) assist systematically in finding the application architecture" [14]. 23–26

Acronyms

AF argumentation framework. 2, 5–7, 17, 41, 43, 44, 48

aVAF audience specific value-based argumentation framework. 53

CI continuous integration. 24, 29

CPAF argumentation framework based on contextual preferences. 16

DRY don't repeat yourself. 34

EAF extended argumentation framework. 2, 5, 9–14, 16–19, 35–38, 41, 43, 44, 48, 49, 53

KISS keep it short and simple. 24

MVC model, view and controller. 41

PAF preference-based argumentation framework. 53

RoR Ruby on Rails. 1, 2, 24, 29, 30, 41, 45, 49

SKB statistical knowledge base. 5, 9, 15, 16, 33, 34, 41, 49

TDD test-driven development. 24, 41

UI user interface. 33–35, 38

UML Unified Modelling Language. 23

VAF value-based argumentation framework. 53

List of Figures

1	Screenshot of the used Trello board used as project management tool.	3
2	Small example argumentation framework.	6
3	Argumentation framework used for some examples.	8
4	extended argumentation framework (EAF) about the weather forecasts with preferences over arguments. Dashed attacks are cancelled out. Double-arrow-headed edges represent attacks on attacks. Green nodes represent accepted arguments in the unique preferred extension.	10
5	EAF with $\langle \mathcal{S}, \mathcal{R}, \mathcal{D} \rangle = \langle \{A, B, C\}, \{(A, B)\}, \{(C, (A, B))\} \rangle$	11
6	EAFs with acceptable and not acceptable sets S' . A_x are elements of S'	12
7	Example of a statistical knowledge base (SKB). Green coloured assumptions hold while red coloured assumptions do not hold. Green coloured models are the possible models after AS 1.	16
8	EAF if $CD1_{heavy}$ holds: The attacks (m_1, m_2) and (m_3, m_2) are cancelled out and m_3 is the only acceptable model w.r.t. to $\mathcal{S}'_1 = \{CD1_{heavy}\}$	18
9	EAF generated for three possible models with context domains $CD1_{light}$ and $CD2_{predict}$	19
10	Project plan for the whole development and research.	22
11	Use case 2.0 activities for iterative development approaches [14].	23
12	Core data flow in the system to create new analyses.	31
13	Listing of research questions: The highlighted areas might not be available, depending on the access rights of the currently logged-in user.	34
14	Overview over a model as a graph representation, in this particular case the <i>Weibull model</i> , which requires two assumptions to hold.	34
15	User interface (UI) to enter preferences between models: General information about this preference including the applicable research question (1), the context under which this preference applies can be defined as assumption (2), and the actual order of the models can be interactively arranged per drag & drop (3).	35
16	Creating a new analysis, step 1: Selection of a data set and research question.	36
17	Creating a new analysis, step 2: Clinician has to answer QueryTest - and QueryAssumptions to perform the analysis. TestAssumptions have already been evaluated.	37

18	Creating a new analysis, step 3: After answering one question (e.g. "A1: Has no non-informative censoring been in place?" with <i>No</i>), the possible models are updated and out-dated QueryAssumptions are moved from the list of queries to the list of <i>Ignored Query Assumptions</i>	38
19	Finished analysis with applied preferences on it: context domain "Censoring" evaluated to <i>heavy censoring</i>	39
20	Entity-Relationship Diagram of the major classes used in the application.	42
21	Explanation of the example data set provided by I. Sassoon.	58
22	Possible research questions arising from the example data set.	59
23	Use Case overview for clinicians.	64
24	Use Case overview for statisticians.	65
25	Use Case overview for administrators.	66
26	RSpec test suite export (Pages 1-2).	69
27	RSpec test suite export (Pages 3-4).	70
28	RSpec test suite export (Pages 5-6).	71
29	RSpec test suite export (Pages 7-8).	72
30	RSpec test suite export (Pages 9-10).	73
31	RSpec test suite export (Pages 11-12).	74

List of Tables

1	Sample performance function for model resilience to censoring (CD1). . .	17
2	Sample performance function for model intent (CD2).	17
3	Orders over models derived from the two different context domains (see Tables 1 and 2).	18
4	List of the different use cases being defined during the requirements analysis.	28
5	Use case 12-1: "Add Test-Argument into the system." An example for a detailed description of a use case.	29
6	Third-party libraries and services used in the web-application.	67

List of Listings

1	R-script to evaluate a TestAssumption on a data set to check whether the underlying data set has been mild censored or not. The performance measurement of CD1 relies on this check (see Table 1).	45
2	Ruby Code to implement Algorithm 1.	61

3	Ruby Code to implement the labelling based approach <i>FIND_PREF</i> presented in Algorithm 2.	62
4	R-script to perform a <code>QueryTestAssumption</code> on a data set to check whether the Weibull-Model is applicable or not. The script generates a plot that will be stored in <code>fileName</code> and presented to the end-user.	62

1 Introduction

Nowadays data collection is omnipresent and the buzzword *big data*¹ is referred to be the next "organizational challenge" most of the industry is or will be facing [27]. However, most of the research is done on the extraction of information from large data sets (so called Big Data Analysis). Therefore small data sets collected in day-to-day practice of professionals are often overlooked.

Clinicians and hospitals collect a lot of data on their patients, the used therapies and the outcomes. Unfortunately, this data is often not used to improve future practice. A recent systematic review has shown, that the usage of statistical analysis has improved the survival analysis slowly [1]. However, this increased usage of statistical models in theory (especially survival analysis) did not result in any noticeable evidence of assumption testing prior to the use of a model in actual clinical studies.

This project aims to implement an intelligent agent that provides advice based on statistical theory on the analysis of such data. The system depends on the design described in the related papers by Sassoon *et al.* [22, 23, 24]. In particular, the most recent publication will be used as solution to the problem of preferences over models in different context domains.

This thesis will first clarify the project aims and objectives and the used methodologies in subsections 1.1 and 1.2. This is followed by a background research in section 2 providing a review of the theoretical aspects of argumentation frameworks, their extensions and the theory underlying statistical model selection. Section 3 presents a detailed project plan and the following section 4 introduces the reader to the used design process. Furthermore, section 5 lists the specifications and a detailed list of requirements of the project. The developed application itself will be introduced to the reader in section 6. The consecutive chapter (see section 7) focuses on the actual implementation of the software as an **Ruby on Rails (RoR)** web application. The report is concluded by a critical evaluation (see section 8) of the project and the delivered web application and an overall conclusion (see section 9). Appendices A to D contain further material related to this project.

1.1 Project Goals and Objectives

The goals of this project can be divided into a list of primary and secondary. In the following lists the endings of an item ($[\dots]$) represent, whether this goal could be achieved or not. A reference to a chapter indicates, where further details can be found. Achieved

¹Big data is often described by the five V's: Volume, Variety, Velocity, Variability, Veracity [11]

goals are marked by an ✓, not achieved tasks are marked with × and partly achieved goals are marked with ∼. For a successful project progression the following primary objectives have to be reached:

- General explanation and summary of [argumentation frameworks \(AFs\)](#), [extended argumentation frameworks \(EAFs\)](#), statistical model selection and the definition of preferences between models related to [context domains](#). [[section 2](#) ✓]
- Development of an [RoR](#) web application that implements the requirements proposed in [[23](#), [24](#)] including but not limited to: [[section 7](#) ✓]
 - An approach to instantiate and solve [AFs](#) and [EAFs](#). [[section 7.2](#) and [7.3](#) ✓]
 - The ability to store, manage and reuse research questions, analysis and preferences for statistical models on different data sets. [[section 7.1](#) ✓]
 - An easy to use user interface to upload data collected during clinical studies and run analyses in an interactive way. [[section 6.3](#) ✓]
 - The ability to deal with preferences between models using [EAFs](#) while taking into account global and personal (end user) preferences. The approach proposed in [[24](#)] involving [context domains](#) will be used. [[section 6.1](#) ✓]
 - A user rights management to allow the system to be used by clinicians, statisticians and super-users (admins). [[section 7](#) ✓]
 - A small set of statistical models and their assumptions integrated in the system (provided by Sassoon in [[24](#)]). [[section 6](#) and [appendix A.2](#) ✓]
 - A comprehensive set of unit and integration tests. [[appendix D](#) ✓]
 - Hosting at a public accessible provider. [[section 5.4](#) ✓]
- Providing the user with an explanation why a statistical model should be used, and why one model might be preferred over another. [[section 6.3](#) ✓]

The secondary goals are desired to be achieved but do not influence the successful finalisation of the project. These objectives are the following:

- A documentation providing information on how to use it and an overview over the key components of the application. [[section 7.1](#) and [appendix B.2](#) ✓]
- A reusable implementation to solve standard [AFs](#) in Ruby as a [gem](#) including documentation and a comprehensive set of unit tests. [×]
- A reusable implementation to solve [EAFs](#) in Ruby as a [gem](#) including documentation and a comprehensive set of unit tests. [×]
- Extended sets of statistical models and their assumptions. [∼]
- A graphical representation of the arguments explaining the actual analysis outcome of the system. [[subsection 6.2](#) ✓]

1.2 Methodologies of the Project

This project will be developed in an agile way. To ensure that it meets the requirements described by Sassoon *et al.* ([22, 23, 24], see subsection 2.4), the main author of those papers is treated as a client or **product owner** during the requirements analysis and the testing phase. For the actual development process the Use-Case 2.0 approach by Jacobson *et al.* [14] is used as it provides a great way to communicate, specify and iterate over functional (independent) parts of the system with non-developers. Due to its descriptive nature it does not require any knowledge about the actual process to be easily understandable. However, this methodology will be explained and introduced in detail later in this thesis (see section 4).

As a project communication and management tool Trello² is utilised as it provides an easy-to-use and interactive way of dealing with cards (in our case **use cases** and tasks) and to group them. During the project planning it was decided that the development would be broken down into four release cycles (RC 1 - 4, see section 3), as this will provide a modularisation of the project and allows early feedback on it. However, tracking of the already achieved intermediate steps and the actual progress of the development process can be done efficiently with Trello. Labels and different lists visualise the status and progress of each task and **use case** (see Figure 1).

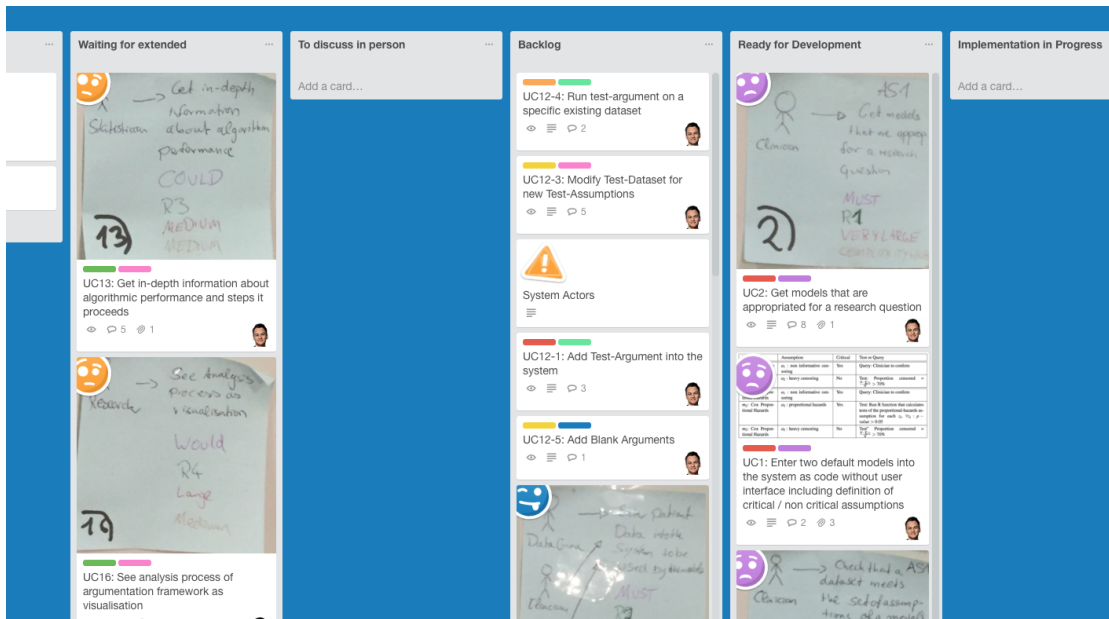


Figure 1: Screenshot of the used Trello board used as project management tool.

²<http://www.trello.com>

1.3 Supplementary Resources

During this Masters Project a web-application was developed, which is publicly available on <http://small-data-analyst.herokuapp.com>. Users can sign-up (approval of an administrator required, please reach out to the author if you have any questions related to that) and upload their own data sets. Some of the existing research questions are shared between all users of the application. However – if required – the source code is available on [Github](#)³ and an installation instruction can be found in the appendix [subsection B.2](#). A detailed list of used third-party applications is attached in [Appendix C](#).

³<https://github.com/sebastianzillessen/small-data-analyst>

2 Background Research

A lot of medical data is collected nowadays on a routine basis by clinicians and could play a critical role to support evidence based decision-making. However, the process of analysing the data and selecting the "correct" statistical model is often a demanding process for clinicians. A system helping them to select and apply appropriate models on these clinical data sets would empower them – even with minimal or no statistical knowledge – to make data driven decisions.

It has been shown in [23], that a system based on argumentation schemes and a [statistical knowledge base \(SKB\)](#) could close the gap between the clinician performing analysis and the statistician having in-depth knowledge of the underlying theory. Following this suggestion, the presented project will employ an argumentation system to analyse the [SKB](#). First, the data set and the input of the user are evaluated. Second, an [argumentation framework \(AF\)](#) is initiated based on this result to validate the usage of particular statistical models. Finally, an [extended argumentation framework \(EAF\)](#) is used to argue about preferences between these models. The relevant theoretical elements for this approach will be explained in the following sections.

2.1 Structure of the Background Research

The approach for a statistical model selection proposed in [23] is based on computational models of argumentation. Usually multiple assumptions have to be fulfilled for statistical models to be applicable on a data set. These assumptions are defeasible and may lead to multiple possible models, which requires argumentation over assumptions and models to provide a reasonable statistical model selection. An introduction to [AFs](#) is given in [subsection 2.2](#).

Non-monotonic argumentation (as proposed in [8, 15]) and monotonic (classic) logic (as proposed in [20]) are in general different approaches to deal with reasoning. However, recent research is focusing on dialogue based approaches, which are mostly based on non-monotonic argumentation [2, 7]. Nevertheless Sassoon *et al.* propose to employ preferences by using [EAFs](#) to reason about the order of applicable models and to deliver a final statistical model that should be used.

To understand this approach, an extension to the standard framework to reason over preferences is presented later in this section (see [subsection 2.3](#)).

Finally, [subsection 2.4](#) provides a summary of Sassoon *et al.* [23] on the problem of finding applicable and choosing preferred models by clinicians for a given research question. In our application we focused on the approach described by the most recent

papers [22, 24], which will be summarised as well.

2.2 Argumentation Theory: General Introduction

In the following section a general overview on **argumentation frameworks (AFs)** will be given. The notation and definitions are based on Dung's theory [8] as it is a widely used definition for **AFs** and the main sources of this thesis [16, 22, 23, 24] are based on this approach.

Definition 2.1. An argumentation framework is a tuple $AF = \langle \mathcal{S}, \mathcal{R} \rangle$ where \mathcal{S} is a set of arguments and $\mathcal{R} \subseteq \mathcal{S} \times \mathcal{S}$. \mathcal{R} is a binary "attack relation".

Notation 2.2. In this paper we use capital letters $\{A, B, \dots\}$ to denote arguments. AB or (A, B) denotes an attack from A to B ($(A, B) \in \mathcal{R}$).

Remark 2.3. An (abstract) argumentation framework can be represented as a directed graph where nodes are arguments and an arrow from a node A to a node B represents an attack from argument A against B .

Remark 2.4. Later in this thesis assumptions that need to hold for a specific model are introduced. These are as well represented as a directed graph, but here an edge from assumption A to model B denotes that the assumption A needs to hold so that B is a possible model. However, the difference will be determinable from the context.

Example 2.1. Figure 2 represents an **AF** with the definition $AF = \langle \{A, B, C, D, E\}, \{(A, B), (B, C), (B, E), (C, B), (D, C), (D, D)\} \rangle$. This framework will be used as an example for the following definitions.

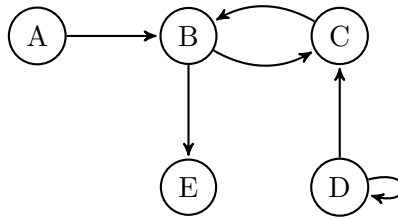


Figure 2: Small example argumentation framework.

Notation 2.5. For the following definitions let $AF = \langle \mathcal{S}, \mathcal{R} \rangle, S' \subseteq \mathcal{S}$.

Definition 2.6. A subset S' is **conflict-free** iff $\forall A, B \in S' : (A, B) \notin \mathcal{R}$ (the subset has no attacks between its arguments).

Remark 2.7. Conflict-free subsets are of interest, as these sets are not directly contradictory. In other words, a conflict-free subset of an argumentation framework does not contain any attacks between its members.

Example 2.2. Conflict-free subsets in Figure 2 are: $\emptyset, \{A\}, \{B\}, \{C, E\}, \dots$. The sets $\{D\}, \{A, B\}, \{B, C\}, \dots$ are not conflict-free.

Definition 2.8. A is **acceptable** w.r.t. a subset S' iff $\forall B \in \mathcal{S} : (B, A) \in \mathcal{R} \Rightarrow \exists C \in S' : (C, B) \in \mathcal{R}$ (an argument A is acceptable with respect to a subset S' iff for each attacker C of A there is an argument in S' that attacks this attacker of A).

Remark 2.9. If an argument A is acceptable w.r.t. a subset S' , then there exists no holding counter-argument in this AF causing the argument A not to hold.

Lemma 2.10. S' **defends** X , if and only if X is acceptable with respect to S' .

Example 2.3. In Figure 2 E is acceptable with respect to $\{A\}$. A is acceptable with respect to \emptyset .

Definition 2.11. The **characteristic function** of an argumentation framework AF (denoted as F_{AF}) is defined by the following:

$$\begin{aligned} F_{AF} : 2^{\mathcal{S}} &\rightarrow 2^{\mathcal{S}} \\ F_{AF}(S') &= \{A \mid A \text{ is acceptable with respect to } S'\} \end{aligned}$$

Notation 2.12. If it is unambiguous, we often refer to F_{AF} with F .

Definition 2.13. A conflict-free set $S' \subseteq \mathcal{S}$ is **admissible** iff S' defends all of its arguments (each argument in S' is acceptable with respect to S') or $S' \subseteq F_{AF}(S')$.

Example 2.4. In the given example in Figure 2 sets $\{A\}, \{A, E\}$ are admissible. However, $\{B\}$ is conflict-free but not admissible, since B is not acceptable with respect to $\{B\}$.

Definition 2.14. S' is a **complete extension** iff S' is admissible and each argument that is acceptable with respect to S' (which is defended by S') belongs to S' .

Remark 2.15. A complete extension is a set of arguments that defends all members and includes all arguments that can be accepted regarding these members. In Figure 2 the set $S' = \{A\}$ is acceptable, but as this set defends as well E (the only attacker B is not acceptable w.r.t. S'), this argument must be included in S' to make the set complete. Hence $S' = \{A, E\}$ is a complete extension.

Example 2.5. $X = \{A, E\}$ is a complete extension in Figure 2, since it is admissible and it defends A and E . Note that C is not defended by X since it does not attack D and D cannot be defended. Complete extensions in Figure 3 are $\{A\}$, $\{A, C\}$ and $\{A, D\}$.

Definition 2.16. A grounded extension (GE_{AF}) of an argumentation framework AF is the minimal (with respect to set inclusion) complete extension of AF . In other words GE_{AF} is the least fixed point of F_{AF} ($GE_{AF} = F_{AF}(\emptyset)$).

Remark 2.17. The grounded extension can be understood as the set of arguments an rational agent can *accept without doubts*, as it contains only the minimal acceptable arguments for an argumentation framework and does not require the agent to assume anything about any argument.

Example 2.6. The grounded extension of the example framework in Figure 2 is $\{A, E\}$. The grounded extension of the example framework in Figure 3 is $\{A\}$.

Definition 2.18. A preferred extension of an argumentation framework AF is a maximised (with respect to set inclusion) admissible set of AF .



Figure 3: Argumentation framework used for some examples.

Example 2.7. The preferred extensions in Figure 3 are $\{A, C\}$ and $\{A, D\}$.

Definition 2.19. A conflict-free set of arguments $S' \subseteq \mathcal{S}$ is called a **stable extension** iff S' directly attacks each argument that does not belong to S' .

Example 2.8. The AF in Figure 3 has the stable extension $\{A, D\}$. $\{A, C\}$ is a preferred, but not a stable extension.

Remark 2.20. The different extensions of a $AF = \langle \mathcal{S}, \mathcal{R} \rangle$ have the following relations between each other [8]:

- Each preferred extension is as well a complete extension.
- Each stable extension is as well a preferred extension.
- The grounded extension is the least (with respect to set inclusion) complete extension and therefore unique for each AF .

- Preferred extensions are the most (with respect to set inclusion) complete extensions.
- Arguments accepted in the grounded extension are sceptically accepted in the AF .
- Every AF has at least one preferred extension.
- A stable extension does not always exist.

Definition 2.21. An argument is regarded as **sceptically accepted under a semantic**, iff it is accepted in all extensions of this semantic (complete, grounded, preferred, stable). An argument is regarded as **credulously accepted under a semantic**, iff it is accepted in at least one, but not all, extensions of this semantic.

Example 2.9. In [Figure 3](#) A is sceptically accepted under each semantic. C, D are credulously accepted under a preferred semantic. D is as well accepted sceptically under the stable semantic.

Furthermore [8] introduces *well-founded argumentation frameworks* (having exactly one extension that is grounded, preferred and stable), *uncontroversial argumentation frameworks* and *coherent argumentation frameworks* (each preferred extension of an AF is stable). Regarding the problem we are looking at, defeasible argumentation is really important, as we are dealing with inconsistent [statistical knowledge bases \(SKBs\)](#). Hence these restricted argumentation frameworks will not be used in this project, therefore they are not discussed any further.

In addition to the discussed extension-based semantics, [15] introduces a so called *labelling-based approach* where there are usually three labels: IN (accepted argument), OUT (rejected argument) and $UNDEC$ (undecided whether this argument is accepted or rejected) and a labelling function $\lambda : \mathcal{S} \rightarrow \{IN, OUT, UNDEC\}$.

By defining legally labelled arguments, definitions for *conflict-free labellings*, *admissible labellings* and *complete/grounded/preferred labellings* can be derived. As there exists a bijective projection, they can be easily transferred to the extension-based semantics already introduced in this section. This labelling based approach has been used to implement the solving algorithm described in [17, 21] later in this thesis (see [subsection 7.3](#)).

2.3 Extended Argumentation Framework - Working with Preferences in AF

In this thesis the [extended argumentation framework \(EAF\)](#) approach introduced by Modgil [16] will be used, as it provides a useful meta-level on preferences between other

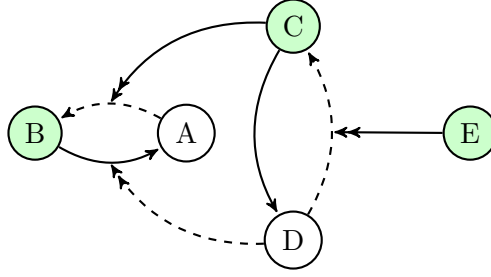


Figure 4: **EAF** about the weather forecasts with preferences over arguments. Dashed attacks are cancelled out. Double-arrow-headed edges represent attacks on attacks. Green nodes represent accepted arguments in the unique preferred extension.

arguments, by extending Dung’s framework with a new attack relation between arguments and attacks. This overcomes the issues of defining orders over preferences or value-evaluation functions (see subsection A.1) and enables us to argue about preferences, regardless of them being defeasible or conflicting [5]. In addition it provides a user-friendly way to consider preferences, which will improve the understandability of clinicians, who will be the main actors (see subsection 5.2) in the final system.

Figure 4 (taken from [16]) shows an example for an **EAF** representing the following arguments:

- *A*: "Today will be dry in London since the BBC forecasts sunshine"
- *B*: "Today will be wet in London since CNN forecasts rain"
- *C*: "But I think the BBC are more trustworthy than CNN"
- *D*: "However, statistically CNN are more accurate forecasters than the BBC"
- *E*: "Basing on a comparison on statistics is more rigorous and rational than basing a comparison on your instincts about their relative trustworthiness"

Sassoon *et al.* define the defeasible knowledge via **EAFs** [24] as it is reasonable to consider an order of importance for different preferences (e.g. preferences based on statistical knowledge should be regarded as more important than the clinicians personal preference). This can be easily achieved in **EAFs** by applying the preferences one by one related to their **context domains**. Therefore, a brief introduction (based on [16]) to the definition of **EAFs** is presented in the following section.

Definition 2.22. An **extended argumentation framework (EAF)** is a triple $\langle \mathcal{S}, \mathcal{R}, \mathcal{D} \rangle$ with \mathcal{S} being a set of arguments and:

- $\mathcal{R} \subseteq \mathcal{S} \times \mathcal{S}$: attack relation.
- $\mathcal{D} \subseteq \mathcal{S} \times \mathcal{R}$: new attack relation of arguments on attacks.

- $\{(A, (B, C)), (A', (C, B))\} \subseteq \mathcal{D} \rightarrow \{(A, A'), (A', A)\} \subseteq \mathcal{R}$ (any arguments expressing contradictory preferences must attack each other).

Remark 2.23. To be able to express a preference between C and B by an additional argument A it is required, that the arguments B and C express contradictory preferences. This can be achieved, if A defines a preference of C over B and the EAF contains $\{(B, C), (C, B)\} \in \mathcal{R}$ and $(A, (B, C)) \in \mathcal{D}$.

Let $\Delta = \langle \mathcal{S}, \mathcal{R}, \mathcal{D} \rangle$ be a EAF and $S' \subseteq \mathcal{S}$ for the following definitions.

Definition 2.24. A **defeats** $_{S'}$ B iff $(A, B) \in \mathcal{R}$ and $\nexists C \in S' : (C, (A, B)) \in \mathcal{D}$. If A **defeats** $_{S'}$ B and B does not **defeat** $_{S'}$ A then A **strictly defeats** $_{S'}$ B .

Notation 2.25. For the rest of the document $A \rightarrow^{S'} B$ denotes that A **defeats** $_{S'}$ B and $A \nrightarrow^{S'} B$ denotes that A does not **defeat** $_{S'}$ B .

By using this definition, similar properties (e.g. conflict-free and admissible sets, acceptability of an argument, sceptically/credulously accepted arguments, extensions) as in Dung's argumentation framework can be introduced and defined.

Definition 2.26. S' is **conflict free** iff $\forall A, B \in S' : (A, B) \in \mathcal{R} \Rightarrow (B, A) \notin \mathcal{R} \wedge \exists C \in S' : (C, (A, B)) \in \mathcal{D}$ (a subset is only conflict free, if for every attack within the subset there is no counter attack and the attack itself is cancelled out by an attack on this attack from an argument that is part of the subset as well).

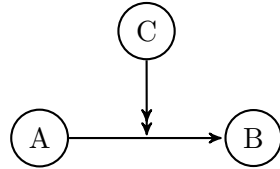


Figure 5: EAF with $\langle \mathcal{S}, \mathcal{R}, \mathcal{D} \rangle = \langle \{A, B, C\}, \{(A, B)\}, \{(C, (A, B))\} \rangle$.

Example 2.10. The set $S' = \{A, B\}$ of the EAF in Figure 5 is not conflict-free. But the set $S' = \{A, B, C\}$ is conflict-free as C attacks the attack between A and B and cancels it out.

Lemma 2.27. Let S' be a conflict-free subset of \mathcal{S} in $\langle \mathcal{S}, \mathcal{R}, \mathcal{D} \rangle$. Then for any $A, B \in S'$ A does not **defeat** $_{S'}$ B .

Definition 2.28. $R_{S'} = \{X_1 \rightarrow^{S'} Y_1, \dots, X_n \rightarrow^{S'} Y_n\}$ is called a **reinstatement set** for $C \rightarrow^{S'} B$ (C **defeats** $_{S'}$ B) iff:

- $C \rightarrow^{S'} B \in R_{S'}$,
- $\forall_{i=1}^n X_i \in S'$,
- $\forall X \rightarrow^{S'} Y \in R_{S'}, \forall Y' : (Y', (X, Y)) \in \mathcal{D}$, there is a $X' \rightarrow^{S'} Y' \in R_{S'}$.

Remark 2.29. A set of attacks is called a reinstatement set (for a particular attack $C \rightarrow^{S'} B$), if for every attack relation in $R_{S'}$, which is attacked from another argument Y' , there is a attack relation in $R_{S'}$ that attacks this argument Y' again and ensures the attack on the argument to hold. So a reinstatement set guarantees, that an attack on an argument is successfully performed, nevertheless which other attacks in the system exist. Hence these reinstatement sets can be used to define acceptability of arguments as seen in [definition 2.30](#).

The acceptability of an argument can now be formally defined based on the reinstatement set.

Definition 2.30. $A \in \mathcal{S}$ is **acceptable** w.r.t. S' iff: $\forall B : B \rightarrow^{S'} A$, there is a $C \in S' : C \rightarrow^{S'} B$ and there is a reinstatement set for $C \rightarrow^{S'} B$.

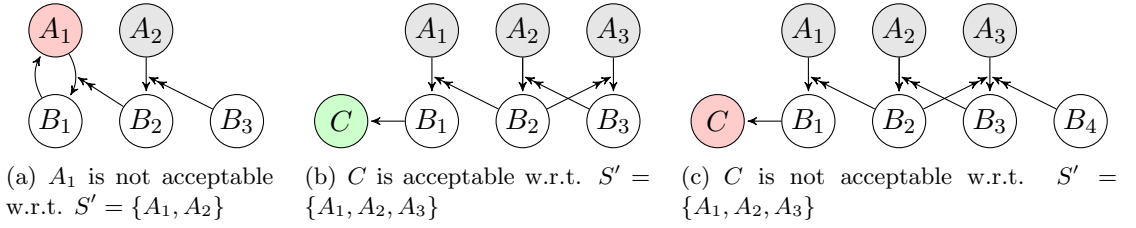


Figure 6: EAFs with acceptable and not acceptable sets S' . A_x are elements of S' .

Example 2.11. In Figure 6(a) $S' = \{A_1, A_2\}$ is not admissible since A_1 is not acceptable w.r.t. S' . In Figure 6(b) C is acceptable w.r.t. $S' = \{A_1, A_2, A_3\}$ as there is a reinstatement set $\{A_1 \rightarrow^{S'} B_1, A_2 \rightarrow^{S'} B_2, A_3 \rightarrow^{S'} B_3\}$ for $A_1 \rightarrow^{S'} B_1$. In Figure 6(c) there is an additional argument B_4 such that $B_4 \rightarrow (A_3 \rightarrow B_3)$ and no argument in S' that defeats $_{S'}$ B_4 , then no reinstatement set for $A_1 \rightarrow^{S'} B_1$ would exist, hence C is not acceptable w.r.t. S' .

Similar to Dung's theory, admissible, preferred, complete and stable extensions of an EAF can now be defined.

Definition 2.31. Let S' be a **conflict free** subset of \mathcal{S} in $\langle \mathcal{S}, \mathcal{R}, \mathcal{D} \rangle$. Then:

- S' is an **admissible** extension iff every argument in S' is acceptable w.r.t. S' .

- S' is a **preferred** extension iff S' is (w.r.t. set inclusion) a maximal admissible extension.
- S' is a **complete** extension iff each argument that is acceptable w.r.t. S' is in S' .
- S' is a **stable** extension iff $\forall B \notin S', \exists A \in S'$ such that A defeats _{s} B .

By using this definition, we can define again **sceptically**, respectively **credulously**, accepted arguments under the semantic $s \in \{\text{preferred, complete, stable}\}$ iff A is in every (at least one) s extension.

Example 2.12. The example given in Figure 4 has only the single preferred, complete and stable extension $\{B, C, E\}$. Figure 5 has the admissible sets $\{A\}, \{A, C\}, \{A, B, C\}$. $\{A, B, C\}$ is the only preferred extension that is as well stable.

Lemma 2.32. Let $\Delta = \langle \mathcal{S}, \mathcal{R}, \mathcal{D} \rangle$ be an **EAF**, S' an admissible extension of Δ and let A, A' be arguments which are acceptable w.r.t. S' . Then:

- $S'' = S' \cup \{A\}$ is admissible.
- A' is acceptable w.r.t. S'' .

Lemma 2.33. Let $\Delta = \langle \mathcal{S}, \mathcal{R}, \mathcal{D} \rangle$ be an **EAF**.

- The set of all admissible extensions of Δ form a complete partial order w.r.t. set inclusion.
- For each admissible extension E of Δ there exists a preferred extension E' such that $E \subseteq E'$.

The definition of the characteristic function for an **EAF** is similar but not equal to Dung's definition.

Definition 2.34. Let $\Delta = \langle \mathcal{S}, \mathcal{R}, \mathcal{D} \rangle$ be an **EAF**, $S' \subseteq \mathcal{S}$, and $2^{\mathcal{S}^c}$ denote the set of all conflict free subsets of \mathcal{S} . The **characteristic function** F_Δ of Δ is defined as follows:

- $F_\Delta : 2^{\mathcal{S}^c} \rightarrow 2^{\mathcal{S}}$
- $F_\Delta(S') = \{A \mid A \text{ is acceptable w.r.t. } S'\}$.

From here on we will always refer to a fixed **EAF**, hence we can simply write F rather than F_Δ . Equally to Dung's Framework, any conflict-free set $S' \subseteq \mathcal{S}$ in Δ is admissible iff $S' \subseteq F(S')$, and complete iff S' is a fixed point of F . We can apply F iteratively on an **EAF**: $F^0 = \emptyset, F^{i+1} = F(F^i)$. Note, that for **EAFs** the characteristic function F is in general **not** monotonic (e.g. C is acceptable w.r.t. $S' = \{A_1, A_2, A_3\}$ in 6(b), but is not acceptable w.r.t. the conflict-free $S'' = S' \cup \{B_2, B_3\}$).

Lemma 2.35. Let F be the characteristic function of an **EAF**, and $F^0 = \emptyset, F^{i+1} = F(F^i)$. Then $\forall i, F^i \subseteq F^{i+1}$ and F^i is conflict free.

Definition 2.36. $\Delta = \langle \mathcal{S}, \mathcal{R}, \mathcal{D} \rangle$ is a **finitary EAF** iff $\forall A \in \mathcal{S}$, the set $\{B \mid (B, A) \in \mathcal{R}\}$ is finite and $\forall (A, B) \in \mathcal{R}$, the set $\{C \mid (C, (A, B)) \in \mathcal{D}\}$ is finite.

Definition 2.37. Let Δ be a finitary **EAF** and $F^0 = \emptyset, F^{i+1} = F(F^i)$. Then $\cup_{i=0}^{\infty} (F^i)$ is the **grounded extension** $GE(\Delta)$ of Δ .

Remark 2.38. Similar to Dung's framework we can state the following relations between different extensions:

- Every **EAF** has at least one preferred extension (implied by [Lemma 2.33](#) as \emptyset is an admissible extension for every **EAF**).
- Every stable extension of an **EAF** is a preferred extension.
- The grounded extension for **EAF** is not defined over the least fix point of the characteristic function F , but can be defined for finitary **EAFs** over the union of all characteristic functions F^i .

A short overview over other approaches to deal with preferences in argumentation frameworks can be found in [subsection A.1](#).

2.4 Statistical Model Selection via Argumentation

The goal of this project is to use argumentation theory for statistical model selection in mostly clinical environments. The demand for systems supporting clinicians during analysis of this data in their day-to-day practice is extending (because of increasing availability, growing size of data sets available for clinicians, and raising awareness on evidence based decision-making). In the following section a short summary on the underlying theory presented by Sassoon *et al.* in [\[23\]](#) will be given, which will be extended by the solution to express the order of preferences in different **context domains** [\[22, 24\]](#).

During the design of clinical studies, clinicians often struggle with the selection of the right model to analyse the retrieved data as they might not be qualified to perform the statistical model selection required for their research question. The process of implementing models, specifying their requirements, providing the arguments for or against these models and specifying preferences between them should be separated from the actual design process and done by a statistician. An intelligent model selection system, which is capable of suggesting appropriate model(s), would reduce administrative and demanding workload and empower clinicians to make data driven decisions.

Sassoon *et al.* [\[23\]](#) address these problems with a defeasible knowledge base, as the (counter-) arguments for specific models might be contradicting. Furthermore the system, the clinician and the statistician might have different preferences over models

that need to be expressed in this knowledge base. Therefore they propose to split the problem into two parts (i) a (defeasible) *knowledge base* that contains the statistical model definitions, the objectives and assumptions of a model; (ii) *argumentation schemes* to guide the model selection process and to represent expressed preferences, which are reflected by *context domain* specific preference orders proposed in later work [22, 24].

The *knowledge base* is used to instantiate the *argumentation schemes* and defines how research objectives can be achieved through different statistical models considering their given assumptions. *Research objectives* are defined as different 'families' of analysis (e.g. survival analysis or categorical outcome variable analysis).

2.4.1 Statistical Knowledge Base

The *statistical knowledge base* (SKB) consists of objectives $\mathcal{O} = \{o_1, \dots, o_u\}$ (different types of research questions), models $\mathcal{M} = \{m_1, \dots, m_v\}$ and assumptions $\mathcal{A} = \{a_1, \dots, a_w\}$. Models represent statistical analysis techniques employable to answer a research question. Assumptions are conditions that ought to be met to employ a model.

Definition 2.39. Let $R_{\mathcal{O}\mathcal{M}} : \mathcal{O} \times \mathcal{M}$ be an m:n⁴-relationship such that $(o_i, m_j) \in R_{\mathcal{O}\mathcal{M}}$ implies objective o_i can be achieved by means of model m_j .

Definition 2.40. Let $R_{\mathcal{M}\mathcal{A}} : \mathcal{M} \times \mathcal{A}$ be the relation between models and their assumptions. $(m_i, a_j) \in R_{\mathcal{M}\mathcal{A}}$ implies the model m_i requires the assumption a_j to be true to be applicable. Let $\mathcal{A}(m_i) = \{a_j | (m_i, a_j) \in R_{\mathcal{M}\mathcal{A}}\}$ be the set of assumptions of m_i .

Remark 2.41. In contradiction to [23] we will not use the proposed approach to distinguish between *critical* and *non-critical* assumptions, as they have been used in the initial approach to express the preference over different possible models. Instead we gonna use the *context domain*-driven approach described in [24].

Definition 2.42. To apply a model m_i all assumptions $\mathcal{A}(m_i) = \{a_j | (m_i, a_j) \in R_{\mathcal{M}\mathcal{A}}\}$ must be met⁵.

Each *assumption* will be either specified as a specific property of the data set (assessed by applying tests on the data set) or as a characteristic of the population of interest or the way in which the data set was collected from that population (relying on the expertise of a domain expert). Sassoon *et al.* proposes a partitioning of all assumptions: \mathcal{A}_t denotes the set of *tests* (applying a test on the available data set). \mathcal{A}_q denotes

⁴Each objective can be achieved by one or more models, each model can answer one or more objectives.

⁵As described in [remark 2.41](#) we regard all assumptions as critical.

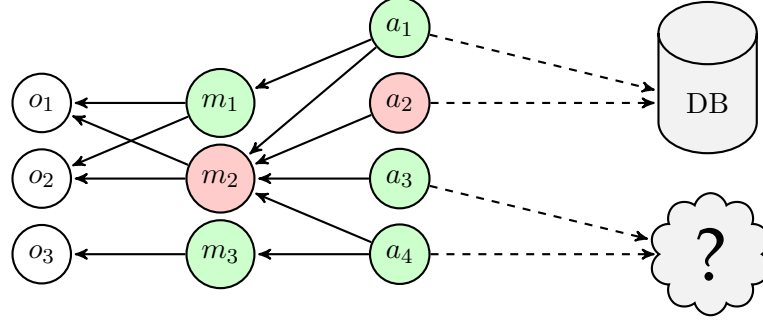


Figure 7: Example of a [SKB](#). Green coloured assumptions hold while red coloured assumptions do not hold. Green coloured models are the possible models after [AS 1](#).

AS 1 Constructed argument for a possible model [\[23\]](#).

- Model m_i achieves objective o_c .
- The data set meets the set of assumptions $\mathcal{A}'_t = \mathcal{A}_t(m_i)$.
- The research project meets the set of assumptions $\mathcal{A}'_q = \mathcal{A}_q(m_i)$.
- $\mathcal{A}(m_i) \subseteq \mathcal{A}'_t \cup \mathcal{A}'_q$.

$\Rightarrow m_i$ is a possible model for the research question o_c .

the set of *queries* (assessed by asking the clinician for an opinion) [\[23\]](#). Lets define $\mathcal{A}_t(m_i) = \{a_j | (m_i, a_j) \in \mathcal{A}_t\}$ and $\mathcal{A}_q(m_i) = \{a_j | (m_i, a_j) \in \mathcal{A}_q\}$.

[Figure 7](#) shows the structure between *objectives*, *models* and *assumptions* in a small example. The assumptions $\{a_1, a_2\} \in \mathcal{A}_t$ are based on the provided data set, $\{a_3, a_4\} \in \mathcal{A}_q$ are based on the domain expertise. Objectives $\{o_1, o_2\}$ can be achieved by the possible model m_1 , o_3 can only be achieved by m_3 .

2.4.2 Adding Preferences for different Context Domains to the Statistical Knowledge Base

To achieve an objective o_c (which has been selected by the clinician) a number of models m_i might be possible providing all their assumptions $\mathcal{A}(m_i)$ are met. The process of instantiating the arguments can be seen in [AS 1](#).

As quoted in [remark 2.41](#) we will not use the initial approach of instantiating AS2 [\[23\]](#), but the [context domain](#)-based approach described in [\[22, 24\]](#): An [EAF](#) can be employed to capture and reason with statistical and research domain knowledge that affects the relative strength of arguments and thereby implies an order over different context domains. To do so, [argumentation frameworks based on contextual preferences](#)

are used in combination with [EAFs](#) by defining a preference ordering $Pref : \mathcal{M} \times \mathcal{M}$ for a given set of models $\mathcal{M} = \{m_1, \dots, m_n\}$ by assigning performance measurements to each model for a given [context domain](#). A sample performance function for model resilience to censoring (CD1) can be seen in [Table 1](#). An additional performance function based on the intention of the model (CD2) can be seen in [Table 2](#) (both taken from [24]).

These orders over the models (see [Table 1](#) and [2](#)) can then be transferred into preference arguments (see [Table 3](#)).

Context Domain	Model	Performance measure	Order
absent	m_1 KM	unaffected	m_1, m_2, m_3
	m_2 PH	unaffected	
	m_3 X^2	unaffected	
light	m_1 KM	unaffected	$m_3 \prec m_1, m_2$
	m_2 PH	unaffected	
	m_3 X^2	affected	
heavy	m_1 KM	affected	$m_1, m_3 \prec m_2$
	m_2 PH	unaffected	
	m_3 X^2	affected	

Table 1: Sample performance function for model resilience to censoring (CD1).

Context Domain	Model	Performance measure	Order
predict	m_1 KM	avoid	$m_1, m_3 \prec m_2$
	m_2 PH	suitable	
	m_3 X^2	avoid	
explain	m_1 KM	suitable	$m_1, m_2 \prec m_3$
	m_2 PH	suitable	
	m_3 X^2	neutral	

Table 2: Sample performance function for model intent (CD2).

Sassoon *et al.* propose further to assign each [context domain](#) a priority, so that they get evaluated one after the other, until one final (if at all possible) preferred model remains. This order should be based on the global relevance of the [context domains](#): Statistical Theory \succ Intention of Analysis \succ Clinicians Preference.

After evaluating the possible models by using [AS 1](#) we will generate an [AF](#) where each of the possible model attacks every other model. Then each [context domain](#) will be initiated and adds its attacks on the existing [AF](#) and translates it into an [EAF](#). This can then be evaluated w.r.t. the argument holding in this [context domain](#) (e.g. $\mathcal{S}'_1 = \{CD1_{heavy}\}$, see [figure 8](#)). If a final decision (only one model is acceptable w.r.t.

$CD1_{light} \twoheadrightarrow (m_3 \rightarrow m_1)$	$CD2_{predict} \twoheadrightarrow (m_1 \rightarrow m_2)$
$CD1_{light} \twoheadrightarrow (m_3 \rightarrow m_2)$	$CD2_{predict} \twoheadrightarrow (m_3 \rightarrow m_2)$
$CD1_{heavy} \twoheadrightarrow (m_1 \rightarrow m_2)$	$CD2_{explain} \twoheadrightarrow (m_1 \rightarrow m_3)$
$CD1_{heavy} \twoheadrightarrow (m_3 \rightarrow m_2)$	$CD2_{explain} \twoheadrightarrow (m_2 \rightarrow m_3)$

Table 3: Orders over models derived from the two different [context domains](#) (see [Tables 1](#) and [2](#)).

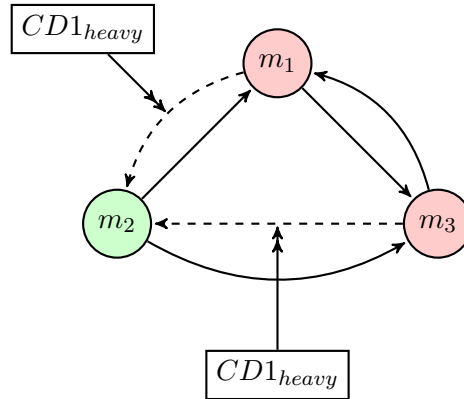


Figure 8: [EAF](#) if $CD1_{heavy}$ holds: The attacks (m_1, m_2) and (m_3, m_2) are cancelled out and m_3 is the only acceptable model w.r.t. to $\mathcal{S}'_1 = \{CD1_{heavy}\}$.

\mathcal{S}'_1) can be made, the process terminates.

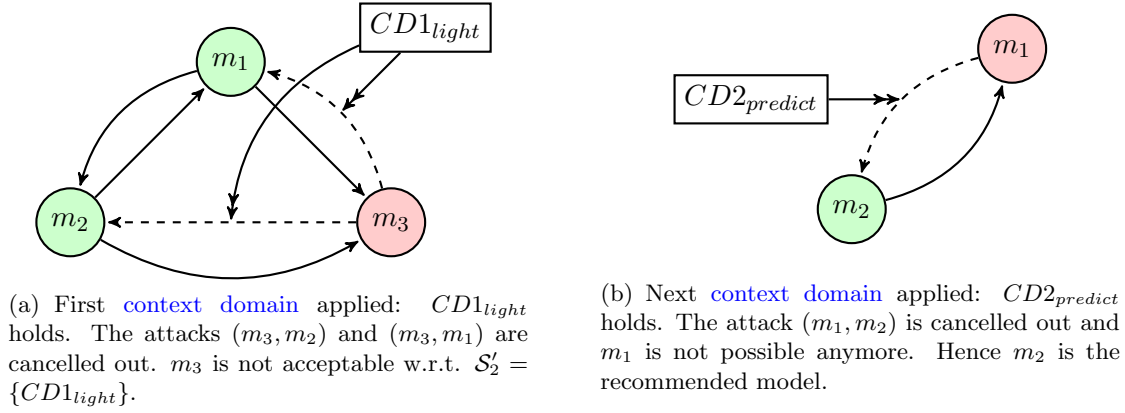


Figure 9: **EAF** generated for three possible models with **context domains** $CD1_{light}$ and $CD2_{predict}$.

Figure 9 shows another example, where we need to evaluate multiple **context domains** to get a final result. First the **EAF** will be evaluated w.r.t. the argument holding in the first **context domain** $\mathcal{S}'_1 = \{CD1_{light}\}$ as shown in Figure 9(a), then the second **context domain** $\mathcal{S}''_2 = \{CD2_{predict}\}$ is applied in Figure 9(b). This is continued until no further preferences are defined or only one – the preferred – model remains.

During this process the **context domains** that relate to statistical theory are regarded as more important than clinician preferences. This is reflected in the final web application by reserving the priorities 1 - 9 to the statistician only (1 has the highest priority).

3 Project Plan

To provide a schedule and to check whether the project is still proceeding at a good pace, a project plan has been developed after performing the requirements analysis and discussing various aspects of the project with the client (the main author of the paper summarised in [subsection 2.4](#)). This project plan can be found as a Gantt-Chart in [Figure 10](#). In general, light blue colour is used for tasks related to the actual thesis (writing, reports, research). Blue coloured tasks represent requirement, analysis and development tasks. Light green tasks are related to project reports whereas dark-green tasks stand for feedback cycles. Green milestones (diamonds) represent internal release candidates of the development iterations. Red milestones represent external milestones and due dates. The overall development process is structured in four release candidates RC1 - RC4.

This schedule was updated according to the actual progress of the project, [Figure 10](#) shows the most recent updated version. Updates on it were necessary because of various reasons:

First of all, the different project phases have changed: The writing of the final thesis has been postponed and the focus on the development has been increased during the first months. In addition as a lesson from RC 1, the duration of the different release candidate have been reduced to iterate faster. A buffer of approximately four weeks has been integrated into the planning to be able to react flexibly on problems that might occur during the development.

Other than that the feedback times have been decoupled from the remaining development process, as we decided to have short demo sessions immediately after each release candidate, therefore the feedback process could be reduced to a minimum.

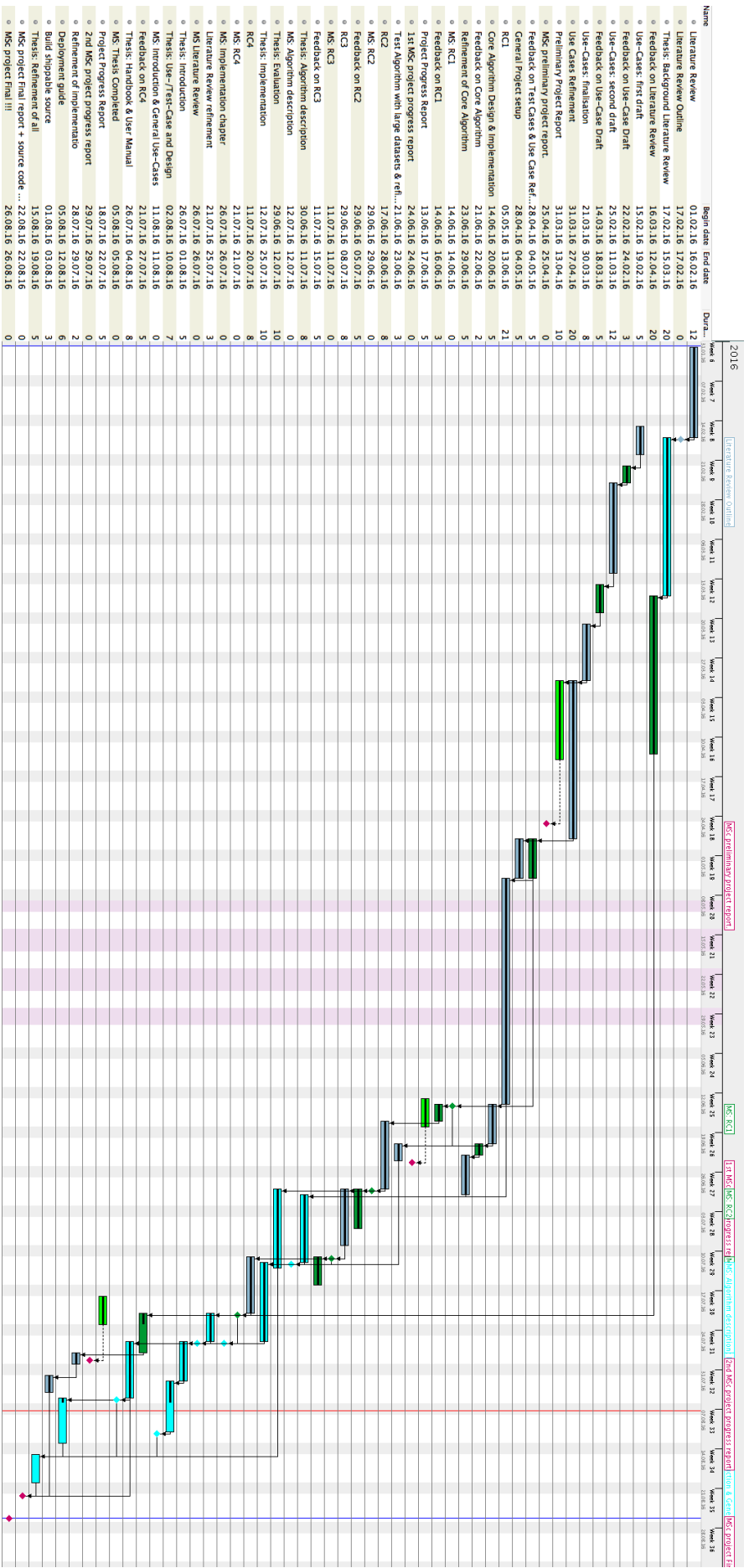


Figure 10: Project plan for the whole development and research.

4 Software Design Process

The software design process has been inspired by [Use cases 2.0](#) of Iva Jacobsen *et al.* [14]. This approach is an extension of the [Unified Modelling Language \(UML\)](#) definition for [use cases](#) and includes a more specific approach on how to deal with different layers of abstraction. This helps to guide the requirement analysis in an efficient way and enables easy communication between software engineers and other stakeholders. A brief overview of the used approach can be found in [Figure 11](#).

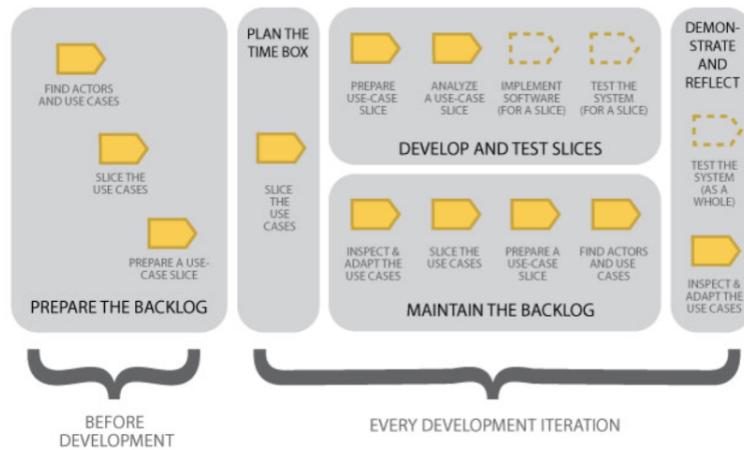


Figure 11: [Use case 2.0](#) activities for iterative development approaches [14].

4.1 Use Cases, Use Case Flows and Use Case Slices

[Use cases](#) are the core concept in Use Case 2.0. In addition to the original [UML](#) specification, [use cases](#) are described as cards including the attributes Priority, Release, Size and Complexity. The MoSCoW scheme [26] (Must, Should, Could, Won't) is used for prioritisation. After the initial requirement analysis, [use case slices](#) are generated usually including flows, tests and an estimation of the development. In this step the different [actors](#) are identified and test cases are specified. To do so, the [use case slices](#) are analysed regarding the interaction of system elements. They are then going to be implemented and tested with the previous defined test scenarios ([unit tests](#) and [integration tests](#)). Finally the whole system will be tested with [end-to-end tests](#).

[Use cases](#) capture and identify the different goals of a system by telling a story on how to achieve a specific goal. This allows an easy communication of requirements of a system [14].

Use case flows define how the interaction between actors and the system proceeds. Usually there is a standard flow, and alternative (error / exception handling) flows.

Use case slices are the final development tasks. They will be generated out of the **use case flows** and can be independently implemented as a single iteration step. They are based on flows and test cases. Further test cases are usually generated during the implementation to react on system specific scenarios and test requirements.

4.2 Test Driven Development

Test-driven development (TDD) encourages simple designs and modularity of software products [10]. Many companies therefore employ a **TDD** process. This project uses **Ruby on Rails (RoR)**, which has test driven mechanics build in its core framework (**rspec**). In combination with the introduced Use Case 2.0 approach, **TDD** can be easily used to ensure correctness between the specification of a **use case** and the actual implementation. In general **TDD** is constructed out of following steps, which are applied iterative:

1. Adding a test.
2. Running all test and seeing the recently added test fail.
3. Implementing the actual feature.
4. Running the tests again and make them pass.
5. Refactoring of the written code.

One key component in **TDD** is to **keep it short and simple (KISS)**, which results in small, testable and reusable modules in software and usually leads to better code quality. As the software will be developed in iterative steps, the progress is easily trackable and although changes are made to already existing features, the integrated tests for the previously implemented functionality ensures the continuous integrity of the overall system.

In our case **unit tests**, **integration tests** and **end-to-end tests** have been used to test the overall system. As described in subsection 5.4 a **continuous integration (CI)** system has been set up to ensure the integrity of the system. The implemented **end-to-end tests** are basically described by the standard- and alternative-flows that are attached to each **use case** as defined in subsection 5.3. The individual tests that were executed can be found in **Figure 26** in the appendix.

5 Specification

In accordance with the selected methodology of Use Case 2.0 (see [section 4](#)) no in-depth specification has been written, however a reasonable amount of [use cases](#) has been defined and iteratively refined. These were then reviewed by the client and discussed in detail. All [use cases](#) have been described on cards on Trello and [use case slices](#) have been generated out of those. They are directly reflected by the description in [\[22, 23, 24\]](#). In general each [use case](#) consists of the following attributes: Name, Desired outcome, (Main-)Actors, Flow to achieve the desired outcome, and Alternative flow (optional).

5.1 Weak Requirements

In addition to the described [use cases](#) in [Table 4](#) (derived from [subsection 1.1](#)), the following weak requirements influenced the development of the final project:

- The system should be designed as an interactive **web application** to ensure:
 - Access for multiple users.
 - Portability between different operating systems.
 - No requirement of installation of software on local computers.
 - Clinician and statistician can share the same infrastructure without the need of working at the same place.
- A decent test suite should be provided to ensure the functionality of the system.
- The system must be suitable for different user types ([actors](#)).
- The system has to provide different levels of access for different user groups.
- The system has to execute test scripts as [R](#)-scripts.
- The system should be hosted on a free accessible hosting provider.

5.2 Actors in our System

During the requirements analysis the following actors have been identified and are described as follows:

Clinicians are the main users of the system and do the actual analysis of a data set by using the predefined research questions, models and assumptions. A clinician should see all globally available research questions, models and their assumptions. In addition the preferences defined by the statistician will apply to all analyses a clinician performs. However, they will be able to enter personal preferences between models that only apply for their analyses. An overview over those [use cases](#) can be seen in [Figure 23](#).

Statisticians are able to enter statistical models into the system and to define global

applicable preferences. To do so, a statistician can enter different types of assumptions mapped to models and preferences. These might contain query assumptions, where the clinician performing an analysis has to confirm a fact, or test assumptions that are checked against the data set by running an R-script. A statistician is as well allowed to see all data sets that have been uploaded and to check which analysis have been performed (see [Figure 24](#)).

Administrators are able to create, modify and delete all available data in the system. In addition an administrator is required to approve new users and to assign them to a role. They are as well capable of inviting new users into the system. An overview over their [use cases](#) can be seen in [Figure 25](#).

5.3 Use Cases

[Table 4](#) shows an overview over the major [use cases](#) being generated during the requirements analysis. For the sake of simplicity they have not been integrated into this thesis. However, they are available on the [Trello-Board](#)⁶.

The [use cases](#) defined in [Table 4](#) were then further defined, deriving requirements have been generated as [use case slices](#) and were implemented by applying an iterative approach. Close feedback loops to the author of [\[22, 23, 24\]](#) ([product owner](#) or client) ensured the correctness of these implementations. As the software was developed in a test driven approach (see [subsection 4.2](#)), all features were implemented by designing tests first. This is reflected by the final test suite (see [Figure 26](#)), which contains a decent amount of [unit tests](#), [integration tests](#) and [end-to-end tests](#) and has an overall test coverage of $\geq 85\%$ ⁷. [Table 5](#) shows an exemplary, detailed description of one [use case](#).

⁶<https://trello.com/b/ywCkicpc>

⁷<https://codeclimate.com/github/sebastianzillessen/small-data-analyst/coverage>

ID	Actor ⁸	Description	RC	Prio ⁹	Comments
UC1	-	Enter two default models into the system as code without user interface including definition of assumptions	RC1	Must	For initial testing purposes
UC2	C	Get models that are appropriate for a research question	RC1	Must	
UC2-1	C	Store finished analyses	RC2	Should	see AS 1
UC3	S	Define assumptions for models	-	-	obsolete
UC4	C	Check that a data set meets the set of assumptions of a model and that the queries to the clinician regarding the research question's assumptions are met as well	RC1	Must	
UC5	C	Select a research question	RC1	Must	
UC6	C, S, A	Login into the system and be authenticated with your role	RC2	Should	
UC6-1	A	Create new users	RC2	Must	
UC6-2	A	Delete users	RC2	Must	
UC6-3	A	Modify users	RC2	Should	
UC7	C	Fill patient data into the system to be checked by the models	RC2	Must	
UC8	C	Inspect arguments for and against models for a research question	RC2	Must	
UC9	C	Define new personal preferences	RC3	Must	
UC9-1	S	Define new global preferences	RC3	Must	
UC10	C	Get recommended model taking into account global preferences	RC3	Should	
UC11	A	Create new users with a specific role for the system	RC3	Could	
UC12	S	Enter additional analysis models into the system in a defined language as text so that it will be regarded as an option for applicable models	RC3	Should	

⁸C: Clinician, S: Statistician, A: Admin

⁹MoSCoW priorities

UC12-1	S	Add Test-Argument into the system	RC2	Must	
UC12-2	S	Add Query-Argument into the system	RC2	Must	
UC12-3	S	Add Test-Query-Assumptions into the system	RC4	Should	added later
UC12-4	S	Run Test-Argument on a specific existing data set	RC3	Should	
UC12-5	S	Add Blank Arguments for grouping of assumptions	RC2	Could	
UC13	S	Get in-depth information about algorithmic performance	RC4	Would	rejected
UC14	C	Enter personal preferences for statistical models and/or analysis techniques	RC4	Could	
UC15	C	Inspect argumentation frameworks that are generated for/against the statistical models. Graph Export: See arguments for and against models for a research question in a nice and graphical	RC4	Would	ignored as printing of generated graphs possible
UC16	C	Inspect analysis process of argumentation framework as visualisation	RC4	Would	
UC17	S	Enter additional research questions into the system	RC3	Could	
UC17-1	S	Modify research questions	RC3	Must	
UC17-2	S	Delete research questions	RC3	Must	
UC18	S	Include R-script execution	RC1	Must	
UC19	S	Enter global preferences for statistical models and/or analysis techniques	RC3	Could	

Table 4: List of the different use cases being defined during the requirements analysis.

ID	UC12-1
Actor	Statistician or Admin
Description	Add Test-Argument into the system
Desired outcome	<ul style="list-style-type: none"> - A statistician can create new arguments - These arguments can be assigned to models - The arguments are accessible for all users in the system
Flow	<ol style="list-style-type: none"> 1.) The user is logged in as Admin/Statistician. 2.) The User clicks on "Add Test-Argument" (see Query arguments, Blank arguments). 3.) The system shows a form including "Name", "Description", "Attacking (Model or Other Argument)", "Required data set fields", "R-Code to run test" and a disabled submit button. 4.) The User enters a name. 5.) The user might enter a description. 6.) The User selects at least on Model or other Argument from the list of possible attacked objects and whether it is critical or non-critical to that object. 7.) The user enters a list of required data set fields. 8.) The user enters the R-Code to test if this argument holds or not. The R-Code must return a true/false statement at the end, <code>True</code> saying the argument holds, otherwise <code>False</code>. The R-Script will get as input a variable named <code>tabular.data</code> and must assign the variable <code>result</code> with <code>true</code> / <code>false</code> (see subsection 7.4). The attributes specified in "Required data set fields" will be checked before executing the R-script. 9.) The user is in charge of verifying his script against data sets. A functionality to do so will be given by selecting a data set in the system first. The script will then be checked against this data set. 10.) The user clicks submit. The system performs a validation of the argument. 11.) The System stores the argument in the database and redirects the user to a list of available arguments and shows a success message.
Alternatives	<ol style="list-style-type: none"> 1.a) Not logged in as Admin: No changes to arguments possible. 4.a) No name entered: show required inline message, disable submit button. 6.a) No attacked object selected: show required inline message, disable submit button. 8.a) If the user does not enter a R-Script: mark as required and do not enable submit button. 9.a) If there is a syntax error: Show syntax error and line to the user. 9.b) If the result is not boolean: Show error explaining true/false result required. 9.c) If there has been an error during execution with the test data set: show error. 9.d) After successful run: Show the data set used to test it and the result. 10.a) If the user does not select "Submit": No changes done. 11.a) If the validation fails: Redirect the user back to the form page and show validation failure. 11.b) If any error occurs, Redirect the user back to the form page and show error explanation.

Table 5: Use case 12-1: "Add Test-Argument into the system." An example for a detailed description of a use case.

5.4 Technical Specification

As the system should be accessible by multiple users and from different departments (e.g. clinicians and statisticians), it will be developed as a web-application in [Ruby on Rails \(RoR\)](#)¹⁰ and will be hosted on Heroku¹¹ as this will provide an easy-to-use and easy-to-deploy environment. In addition it allows us to host the demo application for free. To provide a [continuous integration \(CI\)](#) environment, the project code is hosted on [github](#)¹² and a Travis CI¹³ instance has been setup that runs all the implemented tests of the project.

The used data sets are anonymized, so data protection issues are reduced to a minimum and the data sets can be hosted in the cloud. [PostgreSQL](#)¹⁴ will be used as a database, as it is well integrated on Heroku and provides high scalability. Generated

¹⁰<http://rubyonrails.org>

¹¹<https://www.heroku.com>

¹²<https://github.com/sebastianzillessen/small-data-analyst>

¹³<https://travis-ci.org/sebastianzillessen/small-data-analyst>

¹⁴<http://www.postgresql.org>

plots are stored in an Amazon Web Services S3 Storage¹⁵.

The assumption tests for the statistical models will be performed in [R](#), as this is a common used language for statistical calculations and well known by statisticians. In addition many assumption-checks already exist in [R](#). These scripts (externally provided) will be executed in the [RoR](#) application with the help of third party gems. A detailed description of this process can be found in [subsection 7.4](#).

To provide a responsive and clear user interface the [Bootstrap](#)¹⁶ framework will be used to design and style the application.

5.5 Data Flow in the System

[Figure 12](#) shows the simplified data flows in the process of statistical model selection: First of all, the statistician has to create research questions (1) including their possible models (2) and the required assumptions (3) and global preferences (4). These are all stored in a knowledge base which is itself stored in a persistent database. A detailed description of all database classes can be found in [subsection 7.1](#).

A clinician can upload new clinical data retrieved from studies by using the CSV upload functionality (6). Starting a new analysis (7) retrieves the data set from the database and instantiates the knowledge base for this specific problem. The assumptions of all applicable models for the selected research question are then evaluated. If multiple models are possible, the preferences (first the globally defined ones by a statistician, later the personal preferences (5) a clinician might have) are applied in order of there importance until one last model remains. This model is then returned as recommended model. The process of the rejection of various models will be graphically visible to the user (see [subsection 6.2](#)). The process of creating and processing a new analysis is described in detail in [subsection 6.3](#).

¹⁵<https://aws.amazon.com/s3>, a simple storage service.

¹⁶<http://getbootstrap.com>

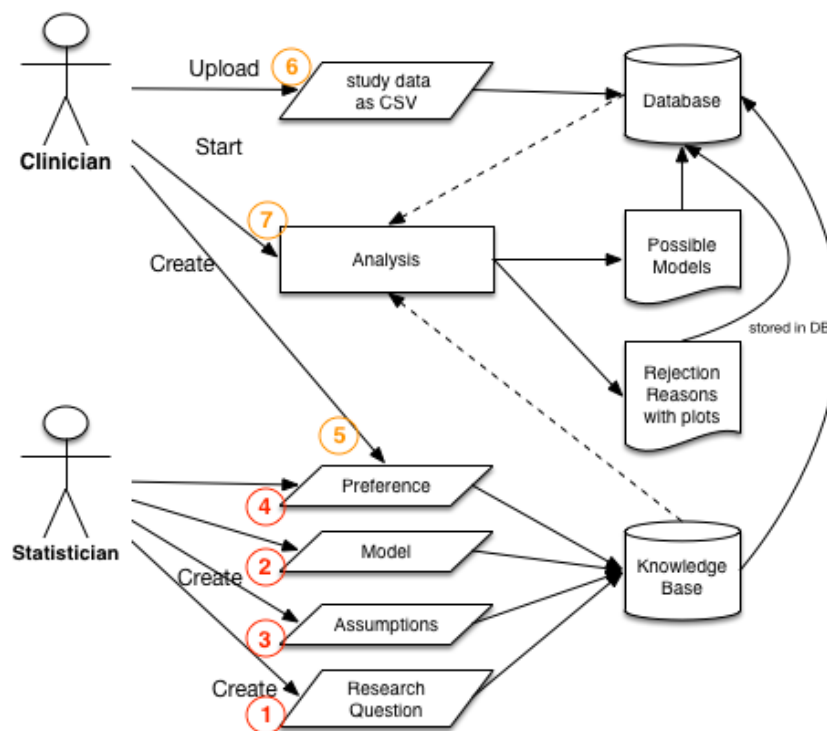


Figure 12: Core data flow in the system to create new analyses.

6 Application Overview

In the following chapter a short introduction to the actual system will be given. The general process, core functionalities and the [user interface \(UI\)](#) will be presented in [sections 6.1](#) and [6.2](#). This is followed by a walkthrough to create a new analysis.

The application is available on <https://small-data-analyst.herokuapp.com> and the explained steps can be reproduced online. A detailed view on some key components of the software can be found in [section 7](#). To test the application the standard *ovarian* data set available in R has been used. In addition to that, I. Sassoon provided an anonymized version of a larger clinical data set (see [subsection A.2](#), [Figures 21](#) and [22](#)). The demo application contains some of the provided research questions, their models and the required assumptions in its [statistical knowledge base \(SKB\)](#).

6.1 General Process and Application Setup

The crucial role in the process of enabling clinicians to analyse their clinical data is played by the statistician: registered or approved by an administrator they are allowed to perform significant changes to the [SKB](#), which is required to provide the clinicians with a sophisticated foundation for their analyses.

To do so a **statistician** will first prepare the system involving the following steps:







1. Defining a research question (see [Figure 13](#)).
2. Assigning suitable models to a research question.
3. Providing assumptions that need to hold to make this model a possible model in a particular analysis.
4. Defining (global) preferences to express a particular order between models in certain contexts that apply if a certain assumption holds (see [Figure 15](#)).

A **clinician** can then initialise a new analysis by performing the following steps:

1. Uploading a data set that has been acquired during a clinical study.
2. Optionally expressing (private) preferences between different models.
3. Generating a new analysis related to a research question and providing the answers required by the system to generate a list of possible models.
4. Answering further questions that might arise during the evaluation of the possible models to apply context domain specific preferences on the data set.

By applying this workflow, **clinicians** will be able to make data-driven decisions that are based on the expertise entered by a **statistician**.

Small Data Analyst	Analysis	Research Questions	Models	Datasets	Advanced ▾	Log out (sebastian@zillessen.info)
--------------------	----------	--------------------	--------	----------	------------	------------------------------------

Available Research Questions			
Name	Description	Private	Actions
Survival Analysis		false	 
Is there a difference in survival (OS) by gender?		false	 
Categorisation		false	 

+ New Research Question

Figure 13: Listing of research questions: The highlighted areas might not be available, depending on the access rights of the currently logged-in user.

6.2 User Interface and Core Functionalities

The **UI** is intentionally kept clean for simplicity reasons. **Bootstrap** is used to provide a consistent look & feel and responsiveness of the application. As described in [subsection 5.2](#) we have three main users in our system, which have different abilities¹⁷ influencing the appearance of the **UI** (eg. a clinician will not be able to create, alter or delete any research questions, however the clinician will still be able to see them as depicted in [Figure 13](#)). This approach allows us to use the same **UI**-components for different users without writing them once again (**don't repeat yourself (DRY)** principle).

To provide easier access to the underlying **SKB**, research questions and models include a visual representation of their relations as a graph (see [Figure 14](#)). The system offers four different types of assumptions, which have special functionalities (explanation of each type can be found in [subsection 7.1](#)).

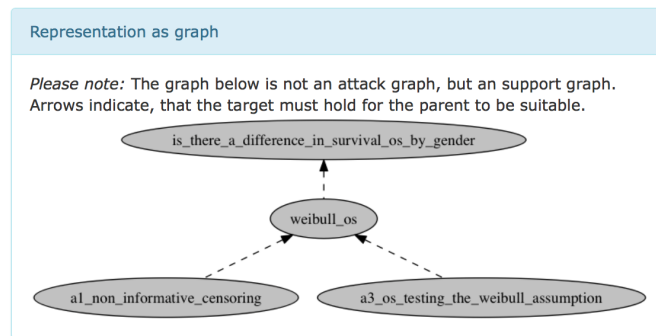


Figure 14: Overview over a model as a graph representation, in this particular case the *Weibull model*, which requires two assumptions to hold.

¹⁷Abilities are access rights, that are specified via an **Ability** class that is used by **cancan** to provide diversified access-rights to different user groups in our system.

Each of those requires different attributes, which results in different forms during the creation process of new assumptions. For a model to be possible during an analysis, all assumptions assigned to this model need to hold (evaluate to true) regardless of its type. In a second step, the defined [preferences](#) are evaluated. To do so, all possible models are added to an [extended argumentation framework \(EAF\)](#) attacking each other as described in [subsubsection 2.4.2](#) and the [context domains](#) are then added iteratively. This process can be seen in [Figure 19](#).

Editing preference

Research Question: Is there a difference in survival (OS) by gender?

Name*


Priority*

☒ Global

Globally generated preferences will apply for all users and can only be modified and created by statisticians they must have a stage value lower than 10.

Assumption*

Definition of the order of the available models*

Please express the order over the available models for this research question by dragging and dropping the below models in the correct order.
 You can add as many  as you want in your order specification.

Remaining available models that can be used to express an order

Specified order for models if the above selected assumption holds

< < <


You can remove an added  by right clicking on it.
 You can drag and drop models to order them or move them between the list by right clicking.

Figure 15: [UI](#) to enter preferences between models: General information about this preference including the applicable research question (1), the context under which this preference applies can be defined as assumption (2), and the actual order of the models can be interactively arranged per drag & drop (3).

The most complex component from an [UI](#) perspective are preferences, as they involve context domains and there required assumptions and relative orders per context domain. [Figure 15](#) shows how this has been solved: (1) allows the user to enter general details for a preference (availability to users, priority and name).

For each available context domain for this particular preference, an assumption (2) has to be chosen. A context domain will be applied, if and only if this assumption holds. The relatively order between models can be specified easily via drag & drop (3).

This order will be taken into account when analysing the **EAF** at the final step of the analysis process. If a preference is not defined over all available models for a particular research question the performance measurement for this model is undefined and it will stay untouched.

As expressed earlier before, the priority of a preference can be set and defines the order of applying different **context domains**. This ensures, that a clinician's preference will always be less important than the statistical reasons a statistician has entered into the system ensuring reliability and statistical correctness while applying the preferences on possible models.

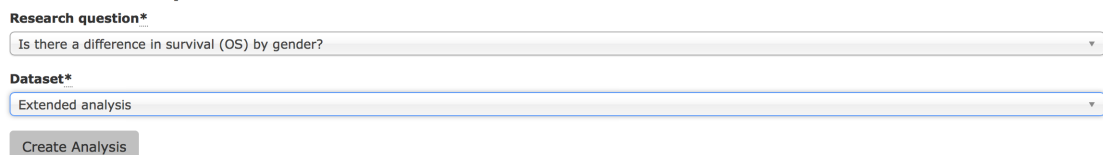
6.3 Example Walkthrough

The following subsection describes a walkthrough for a new analysis involving a sample data set provided by I. Sassoon (see [subsection A.2](#)) and a survival analysis by gender (preferences set according to [24]). These initial options are selected in [Figure 16](#). The following screen (see [Figure 17](#)) shows the actual analysis screen. All **TestAssumptions** have already been evaluated and on the left hand side of the screenshot the *Currently Possible Models* are depicted. On the right hand side the *Open Query Assumptions* are presented to the user including one **QueryTestAssumption** with a generated plot for this particular data set and one **QueryAssumption**.

In this particular case, all entered models for the given research question are possible. If one **TestAssumption** for one of those models would not hold and this model would have further **Query(Test)Assumptions**, these will not be presented to the end-user, as this model is already not possible anymore. If multiple models require the same answers to one **QueryAssumption** (e.g. for our example all three models depend on assumption A1) the clinician will only be asked to answer this question once.

As soon as the clinician answers one of the **QueryAssumptions** the analysis will be updated according to this answer and if some of the **QueryAssumptions** that are already

New analysis



The screenshot shows a web form titled "New analysis". It contains two dropdown menus. The first dropdown is labeled "Research question*" and has the text "Is there a difference in survival (OS) by gender?". The second dropdown is labeled "Dataset*" and has the text "Extended analysis". Below these dropdowns is a button labeled "Create Analysis".

Figure 16: Creating a new analysis, step 1: Selection of a data set and research question.

General information	
Dataset:	Extended analysis
Research Question:	Is there a difference in survival (OS) by gender?
Creation:	05 Aug 03:36
Created by:	sebastian@zillessen.info

Currently Possible Models
Weibull (OS)
PH (OS)
KM (OS)

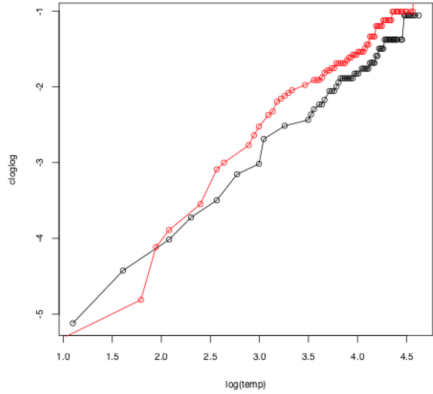
Open Query Assumptions
<p>Question: Are the produced estimated log log lines in the graph below roughly straight?</p>  <p>Yes No</p>
<p>Question: Was there no non-informative censoring?</p> <p>Yes No</p>

Figure 17: Creating a new analysis, step 2: Clinician has to answer **QueryTest**- and **QueryAssumptions** to perform the analysis. **TestAssumptions** have already been evaluated.

visible to the user get obsolete by this answer, they will be removed from the open- and moved to the ignored-list (right lower part of [Figure 18](#)). As the clinician answered A1 with *No*, the analysis results in no possible models, so no preferences are employed and the analysis is finished.

However, if all **QueryAssumptions** have been answered positively all three models remain possible. The system will now initiate the **EAF** for this particular analysis as described in [subsection 2.4](#). The evaluation of this **EAF** is performed in the same order: First, the **TestAssumptions** are evaluated. Second, if no final solution of the **EAF** could be found, the **Query(Test)Assumptions** required for the first preference are evaluated. This process is then repeated with the remaining preferences and their [context domains](#).

In this particular analysis "*Censoring*" has the performance measurement **heavy**.

Possible Models after A51 No possible models found for that analysis and the provided answers.	Recommended models No possible models found for that analysis and the provided answers.
Answered Query Assumptions A1: Non informative censoring: Was there no non-informative censoring?	Ignored Query Assumptions a3 (OS): Testing the weibull assumption: Are the produced estimated log log lines in the graph below roughly straight? Should Cox Proportional hold?: Should Cox Proportional Hold?

Extended Argumentation Framework used to evaluate preferences

Detailed view of models	Detailed view of argumentation frameworks used
-------------------------	--

Figure 18: Creating a new analysis, step 3: After answering one question (e.g. "A1: Has no non-informative censoring been in place?" with *No*), the possible models are updated and out-dated **QueryAssumptions** are moved from the list of queries to the list of *Ignored Query Assumptions*.

Hence, the models *Kaplan-Meier* and *Weibull* are wiped-out during the evaluation of the preferences per [context domain](#). The generated [EAF](#) on the right side of [Figure 19](#) shows the reason why only *Cox-Proportional hazard* remains as the last preferred model.

During the whole process the **UI** provides an interactive way of communication and as it updates its content instantly, the user gets direct feedback on his interaction. In addition, the user is not required to answer all questions in a row, the analysis will still be stored in the system and can be continued at any time.

Extended Argumentation Framework used to evaluate preferences

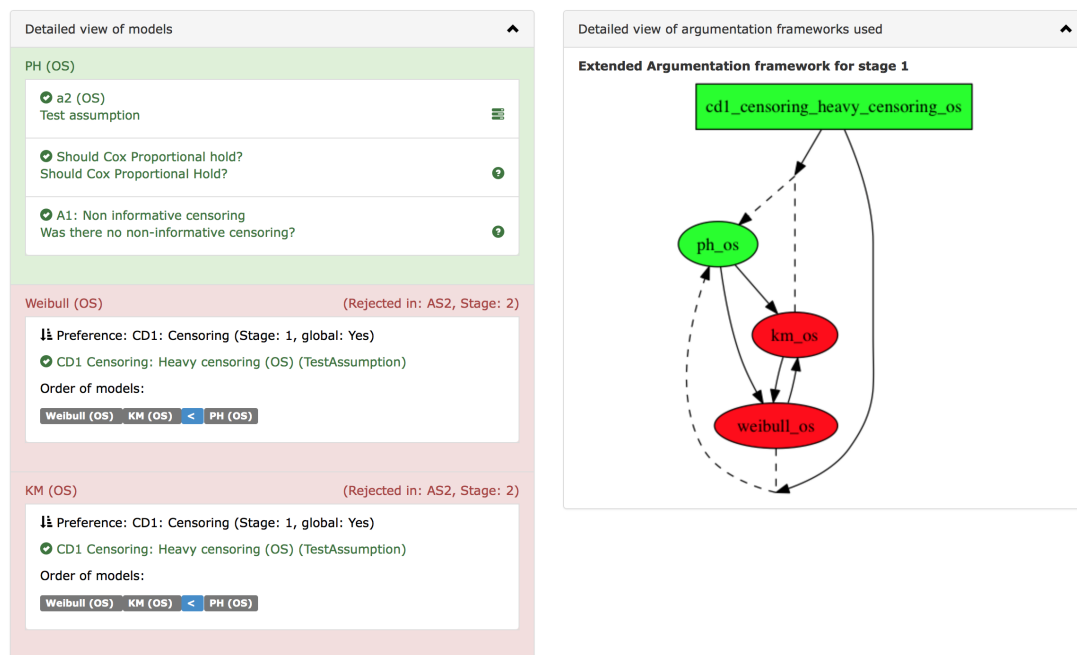


Figure 19: Finished analysis with applied preferences on it: context domain "Censoring" evaluated to *heavy censoring*.

7 Implementation Details

In the following section, some implementation details will be highlighted. An entity relationship diagram explains the core data foundation of this project in [subsection 7.1](#), which represents the [statistical knowledge base \(SKB\)](#) as described in [subsubsection 2.4.1](#). Overall the development of the intelligent agent could be accomplished with general techniques and features (such as the [model, view and controller \(MVC\)](#) architecture) provided by [Ruby on Rails \(RoR\)](#). However, the details of the components related to [extended argumentation frameworks \(EAFs\)](#) (see [subsection 7.2](#)) and [argumentation frameworks \(AFs\)](#) (see [subsection 7.3](#)) should be highlighted, as they are crucial for the successful implementation. The requirements for [R-scripts](#) entered into the application will be presented in [subsection 7.4](#).

As it has been introduced in [subsection 4.2](#), the project used a [test-driven development \(TDD\)](#) approach, which is reflected in the granularity and modularisation of the application (see [Figure 20](#)). An overview over the written tests and the code coverage can be found in [Appendix D](#).

7.1 Entity Relationship Class Diagram

The underlying database structure of the developed application is mostly influenced by the [SKB](#) and the representation of [Assumptions](#) as classes. [Figure 20](#) shows an overview over most of the involved database models used in this application. Additional classes like [Users](#), [Abilities](#), etc. are intentionally left out to reduce complexity.

The most important class is [Analysis](#), which is associated with [Dataset](#) and [ResearchQuestion](#). It has many answered or open [QueryAssumptionResults](#)¹⁸ and a list of [PossibleModels](#) (including impossible models and their [Reason](#) why they got rejected). [ResearchQuestions](#) have and belong to many [Models](#). Each of these has multiple [Assumptions](#) that need to hold for its [Model](#). These [Assumptions](#) have different specialisations:

- **TestAssumption:** An assumption that requires an R-script to be executed and to return `true` or `false`. These assumptions will be checked automatically by the system and rely only on the data set used in an analysis.
- **QueryAssumption:** A clinician has to provide a *yes* or *no* answer to a question during the process of the analysis. Only if he answers positively, this assumption holds.

¹⁸This class stores the answers of a clinician to one particular [QueryAssumption](#)

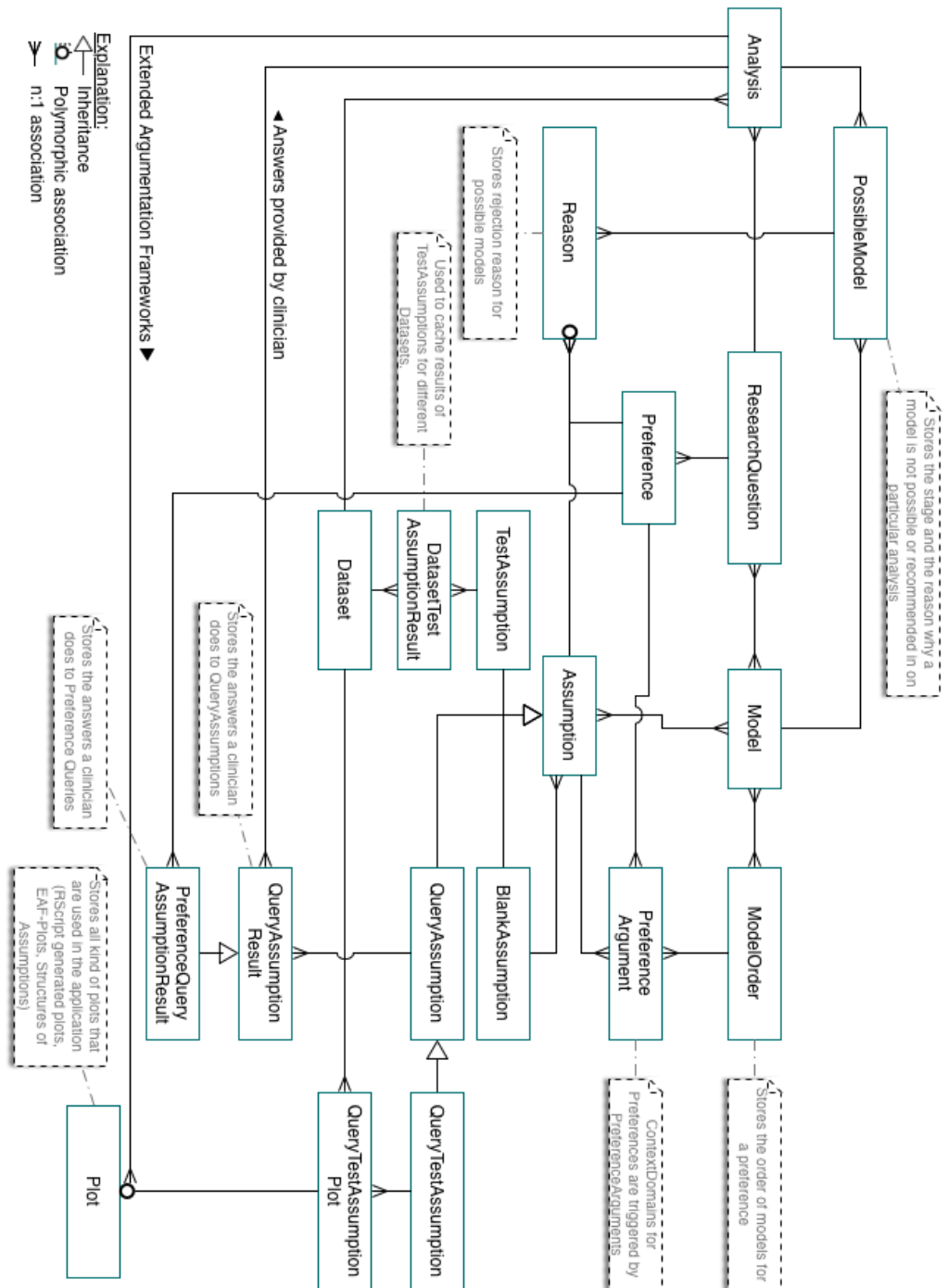


Figure 20: Entity-Relationship Diagram of the major classes used in the application.

- **QueryTestAssumption**: An R-script generates a plot based on the data set, which is then presented to the user who has to confirm, that the plot shows some required features.
- **BlankAssumption**: An assumption that represents a grouping ability for other assumptions. All assigned assumptions (regardless of their type) must hold.

Preferences have different **PreferenceArguments** that represent a specific **context domain**, which are assigned to a particular **ModelOrder**. This allows the representation of **context domains**, their order and the expressed preferences as described in [subsection 2.4.1](#) in a generic way without limiting the triggers for a **context domain** to any particular **Assumption** type.

The class **DatasetTestAssumptionResult** stores for each **TestAssumption** and each **Dataset** a pre-calculated result to enhance the speed of the evaluation during new **Analyses**. **PreferenceQueryAssumptionResults** and **QueryAssumptionResults** store the answers of clinicians to **(Test)QueryAssumptions** for one particular **Analysis**.

7.2 Extended Argumentation Framework: Algorithm

An algorithm to solve **EAFs** was presented in [9] (see [algorithm 1](#)) and is used to solve acceptability w.r.t. a subset $S' \subseteq \mathcal{X}$ of an argument $x \in \mathcal{X}$ in an **EAF**. The presented algorithm defines an **EAF** as a triple $\langle \mathcal{X}, \mathcal{A}, \mathcal{D} \rangle$ ([subsection 2.3](#) introduced it as $\langle \mathcal{S}, \mathcal{R}, \mathcal{D} \rangle$). This algorithm transforms an **EAF** into an **AF** using a colouring-based approach. To do so, it eliminates the attack-on-attacks relation \mathcal{D} by removing elements from \mathcal{A} according to the provided subset S' . In addition, [algorithm 2](#) (see [section 7.3](#)) will be applied to check whether x is acceptable w.r.t. S' in the resulting AF $\langle \mathcal{X}', \mathcal{A}' \rangle$ ¹⁹.

7.3 Argumentation Framework: Algorithm

The algorithm presented in [algorithm 2](#) is based on an labelling approach proposed in [17, 21]. It is supposed to calculate the preferred extensions of an **AF**. **Cand** holds the possible candidates for a preferred labelling as defined in [definition 2.18](#) and is initialised with **Cand** = \emptyset . The Ruby implementation of this algorithm can be found in [listing 3](#). The used class **Labelling** can be found in the provided source code.

¹⁹As the **EAF** gets reduced to an **AF** the following is true: $\mathcal{X}' \subseteq \mathcal{X}, \mathcal{A}' \subseteq \mathcal{A}$.

Algorithm 1 Deciding [EAF](#) Acceptability of $x \in \mathcal{X}$ w.r.t. $S' \subseteq \mathcal{X}$ in $\langle \mathcal{X}, \mathcal{A}, \mathcal{D} \rangle$.

Input: $\langle \mathcal{X}, \mathcal{A}, \mathcal{D} \rangle, S \subseteq \mathcal{X}; x \in \mathcal{X};$
 $\mathcal{A} := \mathcal{A} \setminus \{ \langle y, z \rangle : \neg(y \rightarrow^S z) \};$
 Colour each attack $y \rightarrow^S x$ **BLUE**;
repeat
 for $y \in \mathcal{X}$ s.t. $y \rightarrow^S x$ or $\langle y, \langle u, v \rangle \rangle$ is **BLUE** **do**
 Colour $z \rightarrow^S y$ **RED** for each $z \in S$ s.t. $z \rightarrow^S y$;
 end for
 for $z \rightarrow^S y$ coloured **RED** **do**
 Colour each attack $\langle v, \langle z, y \rangle \rangle \in \mathcal{D}$ **BLUE**
 end for
until No change in attack colours
repeat
 if $\exists y \in \mathcal{X}$ s.t. $\langle y, \langle v, w \rangle \rangle$ is **BLUE** and there is no $u \rightarrow^S y$
 coloured **RED** **then**
 $\mathcal{A} := \mathcal{A} \setminus \{ \langle v, w \rangle \};$
 $\mathcal{D} := \mathcal{D} \setminus \{ \langle y, \langle v, w \rangle \rangle \};$
 end if
 if $\exists z \in S$ s.t. ($\langle z, y \rangle$ is **RED** with $\langle y, \langle u, v \rangle \rangle$ **BLUE**) and
 there is no $\langle p, \langle z, y \rangle \rangle \in \mathcal{D}$ **then**
 $\mathcal{D} := \mathcal{D} \setminus \{ \langle y, \langle u, v \rangle \rangle \};$
 end if
until No change in \mathcal{D}
 Report whether x is acceptable w.r.t. S in the AF $\langle \mathcal{X}, \mathcal{A} \rangle$

Algorithm 2 Algorithm to compute preferred labellings $\lambda \in \text{Cand}$ for [AFs](#) [21].

procedure FIND_PREF(λ)
 if there exists $\lambda' \in \text{Cand}$, such that $\text{in}(\lambda) \subset (\lambda')$ **then return**
 if λ does not have an argument illegally labelled **in** **then**
 Remove from Cand all λ' such that $\text{in}(\lambda') \subset \text{in}(\lambda)$
 $\text{Cand} = \text{Cand} \cup \{ \lambda \}$
 return
 else $\triangleright \lambda$ has args illegally labelled **in**
 if λ has a super-illegaly **in** argument x **then**
 FIND_PREF($\text{step}(\lambda, x)$)
 else
 for each x that is illegally **in** in λ **do**
 FIND_PREF($\text{step}(\lambda, x)$)

Listing 1 R-script to evaluate a `TestAssumption` on a data set to check whether the underlying data set has been mild censored or not. The performance measurement of CD1 relies on this check (see [Table 1](#)).

```
#no censoring = 0, mild censoring < 0.7, heavy censoring >= 0.7
censoring.prop <- (nrow(tabular_data) - sum(tabular_data$os.censor)) /
  ↳ nrow(tabular_data)
result <- (censoring.prop < 0.7)
```

7.4 R-Code and rinruby Gem

To test if a `TestAssumption` holds or not, statisticians have to upload R-scripts that verify these assumptions on the selected data set. This is done by executing the script in [R](#) and returning the result back to the [RoR](#) application. Therefore the scripts have to fulfil certain criteria to be able to communicate correctly with the rest of the agent. The provided scripts are executed using regular R-Syntax. In addition to the by default loaded libraries, the `survival` library will be installed on the executing systems, as it is often used during analyses related to statistical model selection.

Accessing the data set of the analysis as an input in the R-script is essential. The application supports only uploads of CSV files. Thereby the selected format of providing the data sets to the scripts was chosen to be a tabular representation as [R](#) has an excellent support of loading CSV files.

Each script that is executed will have access to the variable `tabular_data`, which has been initialised with `tabular_data=read.csv(file='#{filename}')`. This type of assumption must return its result (whether the assumption holds on this data set or not) in the boolean variable named `result`. The provided example in [listing 1](#) shows, how this `tabular_data` and the `result` variable are used together to perform an assumption check on the data set.

A `QueryTestAssumption` will receive the additional variable `fileName`, which is initialised with an absolute filename to a temporary file. This must be used to store a generated plot during the analysis, as the hosted application does not have read and write access to all folders. For this type of assumption it is necessary to return a valid PNG at the provided file location. Please note, that due to some rendering issues, the Heroku hosted installation only supports PNG generation in R-scripts when the `type` is set to `cairo` (see [listing 4](#)).

To execute the R-scripts the gem `rinruby`²⁰ has been used. As the original gem did

²⁰<https://github.com/clbustos/rinruby>

not provide all required functionalities and features it has been extended and forked at <https://github.com/sebastianzillessen/rinruby>.

In addition to this, separate so called buildpacks have been used to enable [R](#)²¹ and the generation of plots²² on Heroku.

²¹<https://github.com/virtualstaticvoid/heroku-buildpack-r>

²²<https://github.com/weibeld/heroku-buildpack-graphviz.git>

8 Evaluation

The overall project meets the in [subsection 1.1](#) defined primary objectives and aligns with the specified weak requirements in [subsection 5.1](#). Most of the secondary objectives could be achieved. The following section evaluates the developed application and gives a short outlook on future improvements.

8.1 Observation and Feedback

The analyses of the provided data sets (as described in [subsection A.2](#)) are executed with good performance on even larger data sets (≈ 400 rows \times 15 columns). The actual data analysis is done in [R](#), which has been used in statistical environments for many years and proved to be efficient. The results for these scripts are pre-calculated in the background as soon as the data sets are available. Hence, the actual analyses of the different research questions can be performed quickly and is stored for future reference.

The final version of the developed application was tested and evaluated by Isabel Sassoon, scientific assistant of this thesis and product owner during the development process. As a statistician with practical knowledge in data mining and consulting clinicians, she could deliver the following feedback on the application (personal communication, August 2016):

In general the developed app fulfils the original defined requirements. In some aspects it provides even more functionality, such as the implementation of contextual factors and argumentation graphs. The overall layout is the same for the expert and the novice user with some of the advanced options being not available for the novice user profile.

Research questions can be private or shared, which supports clinicians that are analysing the same data to compare and discuss the best approaches. However, this functionality could be extended in the future to provide the ability to publish research questions to groups of collaborators.

8.2 Outlooks

The developed application should be considered as a functional proof of concept and could be extended and improved in various ways. A short list of outlooks is partial based on the feedback of Isabel Sassoon (personal communication, August 2016) follows:

For a prospective trial roll out to clinicians a pre-population of the application with data (including all the relevant research questions, models, assumptions and preferences) would be required, as this will provide deeper feedback on the app from an end user

perspective. In addition to this roll out, a trial with statisticians would provide feedback on the underlying mechanisms of the process. This would as well give the chance to verify the system as a statistician could assess whether the models the system reports are the same as he would recommend. The results of both roll outs could provide a comprehensive analysis of the app and the used methodology. The comparison against a range of criteria could then help to shape any future development plans.

Moreover, an output functionality that covers the steps taken from the research question selected, to the data used, the assumptions tested and the results of the assumptions testing as a summary document could provide better explanation of the process and be used to develop a statistical analysis plan document (SAP).

Furthermore, as a very demanding client, customisation of the layout and the phrasing of texts might be desired to reflect individual workflows. This could be done during the client specific analysis and be part of a detailed design document.

Especially the process of adding new assumptions and models to research questions can be further improved. For now assumptions rely heavily on the underlying structure of the data set. To perform two analysis regarding the research questions *"Is there a difference between the survival time for a treatment group in accordance with their **age** / **weight**"* we would need to add the related test assumptions twice into the system.

This is due to the fact, that the column names in the original data sets should not be changed without documentation as this would conflict the properties of data-provenance [12]. As the only changing aspect between the two research questions are the column names **age** and **weight**, it would be simpler to map a column name of the data set to a specific variable name. This variable could then be used by an assumption during the evaluation of its associated R-script. As a result, the number of required assumptions in the system would reduce drastically by preventing duplication. For each analysis, the clinician would then have to chose, which columns of the data set should be mapped to which variable required by the underlying assumptions. This feature would improve the developed application by adding flexibility and data-provenance to the system.

An additional functional extension might include the ability to fully document a data set provenance and the context in which it was acquired. By implementing an additional anonymisation functionality the agent could then as well be used as data-source for other projects related to the evaluation of clinical data sets and statistical models.

Furthermore, the developed solvers for [argumentation frameworks \(AFs\)](#) and [extended argumentation frameworks \(EAFs\)](#) have not been implemented in a generic way. Extracting these into separate reusable **gems** could add benefits to other projects related to the theoretical approaches.

9 Conclusion

The goal of this project was to implement an already existing theoretical approach described in [22, 23, 24] by developing an intelligent agent, which is capable suggesting appropriate models to the end-user (usually a clinician). During the analysis of a particular data set different assumptions of these models must be evaluated to decide which are applicable. Furthermore, the application should present the relevant arguments for and against the various possible models for a specific research question. To resolve the problem of defeasible and contradictory preferences between these models, different initialisations of multiple [extended argumentation frameworks \(EAFs\)](#) based on [context domains](#) have been used as described in [22, 24].

The developed web-application provides an intuitive and easy-to-use approach to perform statistical model selection based on properties of the underlying data, statistical theory, as well as global and personal preferences. This allows statisticians with a deeper knowledge of statistical theory to provide their expertise to clinicians with insights about the acquisition and external properties of a data set.

As a statistician has to enter the [statistical knowledge base \(SKB\)](#) only once the workload for clinicians is reduced to a minimum. This allows them to focus on the actual analysis of the underlying data and equips them with a low-barrier tool, that they might be more likely to use over a longer period of time, which leads ultimately to a continuous data-driven decision process. As clinicians might not always be qualified to select the appropriated statistical models during the design stage of a study [23] a support-agent is essential for a data-driven process.

Only well known and acknowledged techniques have been used to evaluate the (extended) argumentation frameworks, which provides a well-founded root for the overall decision process. In addition graphical representations of the used frameworks have been added to the web-application to provide easy access to the argumentation techniques, especially if they are not familiar to the end-user.

As described in [subsection 8.2](#), the application could be improved by adding the functionality to provide data-provenance by providing "dynamic" [R](#)-scripts (scripts that are binding to a mapping of database columns instead of names).

As the underlying work of I. Sassoon [24] is still in progress, it is likely that further strategies regarding the selection of possible models and evaluating their preferences over one and another will be developed. As the application²³ is written in the well-known framework [Ruby on Rails \(RoR\)](#) and includes a sophisticated test suite, changes and

²³Published under the [MIT license](#).

extensions are easily possible.

Although the requirements for the project changed slightly (see [subsubsection 2.4.2](#)) the agile software development process and the defined [use cases](#) allowed to react flexible on these changes and provided a good source for discussions with the supervisor and the scientific assistant of this project, which lead to no significant delays during the development process.

References

- [1] V. Abraira, A. Muriel, J. Emparanza, J. Pijoan, A. Royuela, M. N. Plana, A. Cano, I. Urreta, and J. Zamora, “Reporting quality of survival analyses in medical journals still needs improvement. a minimal requirements proposal,” *Journal of Clinical Epidemiology*, vol. 66, no. 12, pp. 1340–1346, July 2016.
- [2] L. Amgoud, N. Maudet, and S. Parsons, “Modelling dialogues using argumentation,” in *MultiAgent Systems, 2000. Proceedings. Fourth International Conference on*, 2000, pp. 31–38.
- [3] L. Amgoud and C. Cayrol, “On the acceptability of arguments in preference-based argumentation,” in *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence*. Morgan Kaufmann Publishers Inc., 1998, pp. 1–7.
- [4] L. Amgoud and C. Cayrol, “A reasoning model based on the production of acceptable arguments,” *Annals of Mathematics and Artificial Intelligence*, vol. 34, no. 1, pp. 197–215, 2002.
- [5] L. Amgoud, S. Parsons, and L. Perrussel, “An argumentation framework based on contextual preferences,” *Proceedings of the 3rd International Conference on Formal and Applied Practical Reasoning (FAPR’00)*, pp. 59–67, 2000.
- [6] T. J. M. Bench-Capon, “Persuasion in practical argument using value-based argumentation frameworks,” *Journal of Logic and Computation*, vol. 13, no. 3, pp. 429–448, 2003.
- [7] W. Douglas and E. C. W. Krabbe, *Commitment in Dialogue: Basic Concepts of Interpersonal Reasoning*. State University of New York Press, 1995.
- [8] P. M. Dung, “On the acceptability of arguments and its fundamental role in non-monotonic reasoning, logic programming and n-person games,” *Artificial Intelligence*, vol. 77, no. 2, pp. 321 – 357, 1995.
- [9] P. E. Dunne, S. Modgil, and T. Bench-Capon, “Computation in extended argumentation frameworks,” in *Proceedings of the 2010 Conference on ECAI 2010: 19th European Conference on Artificial Intelligence*. Amsterdam, The Netherlands, The Netherlands: IOS Press, 2010, pp. 119–124.
- [10] S. Fraser, K. Beck, B. Caputo, T. Mackinnon, J. Newkirk, and C. Poole, “Test driven development (tdd),” in *Proceedings of the 4th International Conference on*

- Extreme Programming and Agile Processes in Software Engineering*, ser. XP'03. Berlin, Heidelberg: Springer-Verlag, 2003, pp. 459–462.
- [11] M. Hilbert, “Big data for development: A review of promises and challenges,” *Dev Policy Rev*, vol. 34, no. 1, pp. 135–174, dec 2015.
 - [12] D. Hills, R. R. Downs, R. Duerr, J. Goldstein, M. Parsons, and H. Ramapriyan, “The importance of data set provenance for science,” *Eos*, vol. 96, dec 2015.
 - [13] K. Hornik. (2016, July) R faq. Last access: Aug 8, 2016. [Online]. Available: <https://cran.r-project.org/doc/FAQ/R-FAQ.html#What-is-R.003f>
 - [14] I. Jacobson, I. Spence, and K. Bittner, *USE-CASE 2.0 The Guide to Succeeding with Use Cases*. Ivar Jacobson International SA., Dezember 2011.
 - [15] B. Liao, *Efficient Computation of Argumentation Semantics*, 1st ed. Academic Press, 2014.
 - [16] S. Modgil, “Reasoning about preferences in argumentation frameworks,” *Artificial Intelligence*, vol. 173, no. 910, pp. 901 – 934, 2009.
 - [17] S. Modgil and M. Caminada, *Proof Theories and Algorithms for Abstract Argumentation Frameworks*. Boston, MA: Springer US, 2009, pp. 105–129. [Online]. Available: http://dx.doi.org/10.1007/978-0-387-98197-0_6
 - [18] J. L. Pollock, “Defeasible reasoning,” *Cognitive science*, vol. 11, no. 4, pp. 481–518, 1987.
 - [19] H. Prakken and G. Sartor, “Argument-based extended logic programming with defeasible priorities,” *Journal of Applied Non-Classical Logics*, vol. 7, no. 1-2, pp. 25–75, 1997.
 - [20] R. Reiter, “A logic for default reasoning,” *Artificial Intelligence*, vol. 13, no. 12, pp. 81 – 132, 1980, special Issue on Non-Monotonic Logic.
 - [21] O. Rodrigues, “Artificial intelligence: Common-sense reasoning. computation of argumentation semantics.” Nov 2015, lecture slides, King’s College London [unpublished].
 - [22] I. Sassoon, J. Keppens, and P. McBurney, “Incorporating preferences and argumentation schemes for statistical model selection,” [unpublished].

- [23] I. Sassoon, J. Keppens, and P. McBurney, “Towards argumentation for statistical model selection.” in *COMMA*, 2014, pp. 67–74.
- [24] I. Sassoon, J. Keppens, and P. McBurney, “Preferences in argumentation for statistical model selection,” in *COMMA*, 2016, [under submission].
- [25] M. G. Software. Scrum product owner: The agile product owner’s role. Last access: Aug 8, 2016. [Online]. Available: <https://www.mountangoatsoftware.com/agile/scrum/product-owner>
- [26] K. Walters. (2009, Jan) Prioritization using moscow. Last access: Aug 7, 2016. [Online]. Available: <http://www.allaboutagile.com/prioritization-using-moscow/>
- [27] R. Wegener and V. Sinha. (2013, Sep) The value of big data: How analytics differentiates winners. Last access: Aug 8, 2016. [Online]. Available: <http://www.bain.com/publications/articles/the-value-of-big-data.aspx>

Declaration

I declare that this thesis is the solely effort of the author. I did not use any other sources and references than the listed ones. I have marked all contained direct or indirect statements from other sources as such.

Neither this work nor significant parts of it were part of another review process. I did not publish this work partially or completely yet. The electronic copy is consistent with all submitted copies.

Signature and date:

A Appendix

A.1 Other Approaches for Preferences in Argumentation Frameworks

Other approaches to deal with preferences in argumentation frameworks have been proposed by [4, 3, 6, 18, 19]. A [preference-based argumentation framework \(PAF\)](#) defined in [3] is a triple $\langle \mathcal{S}, \mathcal{R}, Pref \rangle$ where $Pref$ is a (partial or complete) order on $\mathcal{S} \times \mathcal{S}$. The difference between this approach and the one we use is mostly the requirement of a strict ordering that has to be associated with $Pref$ and must be explicitly defined.

Modgil *et al.* introduce in [16] the concept of a special class of [hierarchical extended argumentation frameworks \(EAFs\)](#), which are defined by the existence of a partition of Δ into multiple regular Dung argumentation frameworks. Due to this restriction it is possible to define a least fix point of the characteristic function F , hence defining the grounded extension $GE(\Delta)$ in the same way, as it has been defined in Dung's framework. However, this is a too narrow restriction for our purposes, therefore this will not be described here.

[Value-based argumentation frameworks \(VAFs\)](#) as proposed in [6] define an argumentation framework as a 5-tuple: $\langle \mathcal{S}, \mathcal{R}, V, val, P \rangle$ (\mathcal{S} : Arguments, \mathcal{R} : Attack relation, V : nonempty set of values, $val(\cdot) : \mathcal{S} \rightarrow V$: value mapping function, P : set of audiences $\{a_1, \dots, a_n\}$ where each audience names a total ordering $>_{a_i}$ on $V \times V$). By referring to one specific audience we retrieve a [audience specific value-based argumentation framework \(aVAF\)](#). The set of audiences P is introduced to be able to make use of preferences between values in V , so we might have as many audiences as there are orderings on V . The new definition of an argument that defeats another argument takes into account the audience a and the $val(\cdot)$ of both arguments to define successful attacks. This approach requires a value mapping function $val(\cdot)$ and doesn't argue with preferences in a natural way.

However, [16] proves that an [aVAF](#) can be transferred to an *equivalent* [EAF](#) by representing the [aVAF](#) with three layers: The outer layer expressing the audience, the second layer expressing the pairwise orderings on values in V , and the inner layer based on the actual arguments and attacks in the [aVAF](#).

Defeasible reasoning and preferences and their impact on argumentation frameworks are formalised as logical formalism by [18, 19] in the underlying logical formalism which will be used to instantiate a regular Dung framework.

A.2 Sample Clinical Data Set

Figures 21 and 22 have both been provided by I. Sassoon in a private email conversation.

Argumentation for Statistical Model Selection

I.Sassoon

Sample clinical data for use in MSc project.

The data is analysis_data.csv and it contains 27 attributes and 415 cases. The following columns are in the data:

Name	Type	Comments
id	identifier	
age	interval	
Gender	binary	
Performance.Status	nominal	
Site.in.mouth	nominal	
Lymphoscintigraphy.type	nominal	
Path.depth	interval	
T.Size	nominal	
Peri.neural.spread	binary	
Differentiation	nominal	
No.of.positive.nodes	interval	
No.of.negative.nodes	interval	
Sentinel.node.status	binary	
Recurrence.status	binary	
Type.of.recurrence	nominal	
Treatment.of.recurrence	nominal	
Metastasis.Type	nominal	
Current.status	nominal	
os	time to event	Overall Survival
os.censor	censoring indicator	
dfs	time to event	Disease Free Survival
dfs.censor	censoring indicator	
dss	time to event	Disease Specific Survival
dss.censor	censoring indicator	
false.negative	binary	
rec.3.yr	nominal	
fu.3yr	binary	

Figure 21: Explanation of the example data set provided by I. Sassoon.

Argumentation for Statistical Model Selection I.Sassoon

Potential Research Questions:

Is there a difference in survival (OS) by gender?
Is there a difference in survival (DFS) by sentinel node status?

These are based on the objective "time to event".

Additional research objectives:

This is another type of objective for the SKB. It is relevant or cases where the target variable in of type interval. A research question of this type would be:

Is there a difference in ages (Interval) by Gender (nominal or binary)?

Figure 22: Possible research questions arising from the example data set.

B Software related Appendix

B.1 Code Samples

Listing 2 Ruby Code to implement [Algorithm 1](#).

```

def to_dung_framework(subset, x, fw)
  raise ArgumentError, "#{subset.map(&:to_s).join(",")} is not a subset of #{fw.arguments.map(&:to_s).join(",")}" unless
    ↪ subset.subseteq?(fw.arguments)
  attacks = fw.attacks.reject { |a| !a.succeeds_wrt(subset, fw) }
  attacks_on_attacks = fw.attacks_on_attacks.clone
5  colors = {}
  attacks.select { |a| a.target == x }.each { |a| colors[a] = BLUE }
  change_in_attack_color = false
  begin
    change_in_attack_color = false
10  # lines 5-7 in original code
    fw.arguments.select { |y|
      fw.attacks(y).select { |e| e.target == x && e.succeeds_wrt(subset, fw) }.any? ||
      fw.attacks_on_attacks(y).select { |yuv| colors[yuv] == BLUE }.any?
    }.each do |y|
15    subset.map { |z| fw.attacks(z).select { |a| a.target == y }.flatten.each do |edge|
      if (colors[edge] != RED)
        colors[edge] = RED
        change_in_attack_color = true
      end
20    end
    end
    # lines 8-10 in original code
    colors.select { |e, v| v == RED }.each do |e, v|
      fw.attacks_on_attacks.select { |a| a.target == e }.each do |aoa|
25    if (colors[aoa] != BLUE)
      colors[aoa] = BLUE
      change_in_attack_color = true
    end
    end
30  end
  end while change_in_attack_color
  change_in_d = false
  begin
    change_in_d = false
35  # line 13 in original code
    colors.select { |e, v| v == BLUE }.select { |e, v| colors.select { |e2, v| v==RED && e2.target == e.source &&
    ↪ e2.succeeds_wrt(subset, fw) }.empty? }.each do |edge, v|
      # edge is <y, <u,v>>
      attacks -= [edge.target]
      if attacks_on_attacks.delete(edge)
40    change_in_d = true
      end
    end
    # line 17 in original code
    subset.each do |z|
45    # (<z,y> is RED with <y,<u,v>> is BLUE)
      fw.attacks(z).each do |zy|
        y = zy.target
        if (colors[zy] == RED)
          fw.attacks_on_attacks(y).select { |yuv| colors[yuv] == BLUE }.each do |yuv|
50    # and there is no <p,<z,y>> in D}
          if (attacks_on_attacks.select { |aoa| aoa.target == zy }.empty?)
            if (attacks_on_attacks.delete(yuv))
              change_in_d = true
            end
55    end
          end
        end
      end
    end
60  end while change_in_d

  # remove from attacks on attacks all edges that point on a not existing terminal edge
  remove_all_non_terminal_edges(attacks, attacks_on_attacks)
  [attacks, attacks_on_attacks]
65 end

```

Listing 3 Ruby Code to implement the labelling based approach *FIND_PREF* presented in [Algorithm 2](#).

```

def find_pref(labelling, cand = [])
  # check if candidate labelling exists where current labelling is a subset of
  # return cand if cand.select { |l| labelling.arg_in.subseteq?(l.arg_in) }.any?
  # if labelling does not have an argument illegally labelled in then Remove
  #   ↪ from Cand all subsets of the current labelling
5  if (labelling.illegally_labelled_ins.empty?)
    cand.reject! { |l| l.arg_in.subseteq?(labelling.arg_in) }
    cand << labelling
  else
    if x = labelling.super_illegally_labelled_in.first
10    cand = find_pref(labelling.step(x), cand)
    else
      labelling.illegally_labelled_ins.each do |x|
        cand = find_pref(labelling.step(x), cand)
      end
15  end
end
cand
end

```

Listing 4 R-script to perform a `QueryTestAssumption` on a data set to check whether the Weibull-Model is applicable or not. The script generates a plot that will be stored in `fileName` and presented to the end-user.

```

library("survival")
m=1
tabular_data.survfit<-survfit(Surv(tabular_data$futime,
  ↪ tabular_data$fustat==1)~tabular_data$rx,data=tabular_data)
n=tabular_data.survfit$strata[1]
5 temp<-tabular_data.survfit$time[m:n]
  cloglog=log(-log(tabular_data.survfit$surv[m:n]))
  png(file = fileName, type="cairo")
  plot(log(temp), cloglog, type = "o")
  m=n+1
10 n=n+tabular_data.survfit$strata[2]
  temp=tabular_data.survfit$time[m:n]
  cloglog=log(-log(tabular_data.survfit$surv[m:n]))
  lines(log(temp),cloglog,type="o",col=2)
  dev.off()

```

B.2 Installation Guide

The following installation guide has been tested on a clean debian OS:

1. Install postgresql and R with: `sudo apt-get install postgresql r-base`
2. Install rvm following [this guide](#).
3. Clone the repository:
`git clone https://github.com/sebastianzillessen/small-data-analyst.git`
4. `cd small-data-analyst`
5. Install the required ruby version as prompted by rvm: `rvm install ruby-2.2.4`
6. Install bundler: `gem install bundler foreman`
7. Install all required gems: `bundle install`
8. Set the AWS credentials in the file `.env`:

```
PORT=3000
AWS_ACCESS_KEY_ID=XXXXXXXXXXXXX
AWS_SECRET_ACCESS_KEY=YYYYYYYYYYYYYYYYYYY
S3_BUCKET_NAME=ZZZZZZZZZZZZZZ
```

9. Set the database credentials if required in `config/database.yml`.
10. Initialise the database:
`rake db:create && rake db:setup && rake db:migrate`
11. Start the server with `foreman start`
12. Create an Admin user:

```
$ rails console
> u = User.create(email: "test1@test.de", password: "fooPassword",
approved: true, role: :admin)
> u.confirm!
```

13. Navigate to `http://localhost:3000` and play around with the application.

B.3 Use Cases

Figures [23](#), [24](#) and [25](#) provide an overview over the implemented [use cases](#) in the application grouped by their main actors. A complete list of all [use cases](#) can be found for further references under <https://trello.com/b/ywCkicpc/msc-small-data-analyst>.

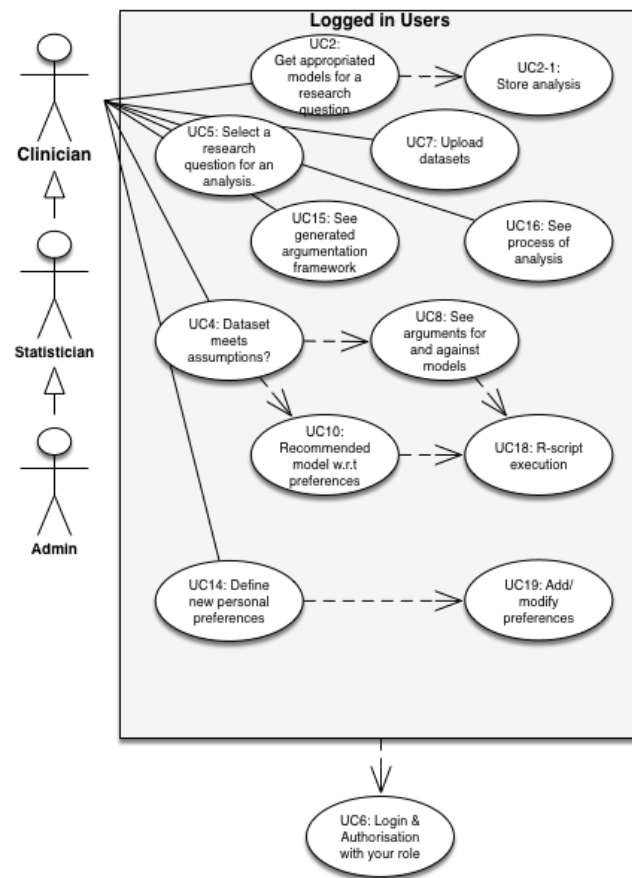


Figure 23: Use Case overview for clinicians.

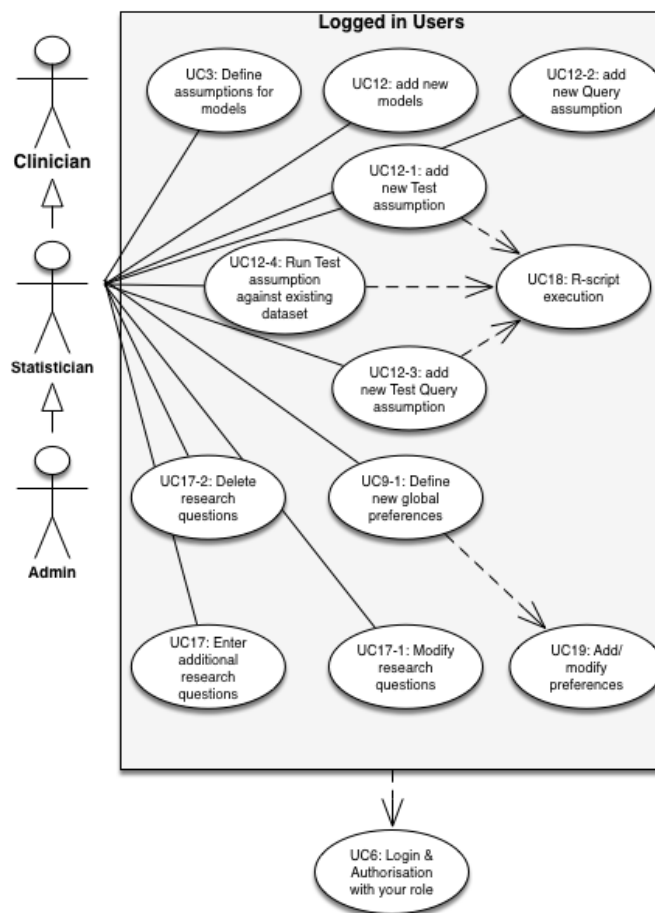


Figure 24: Use Case overview for statisticians.

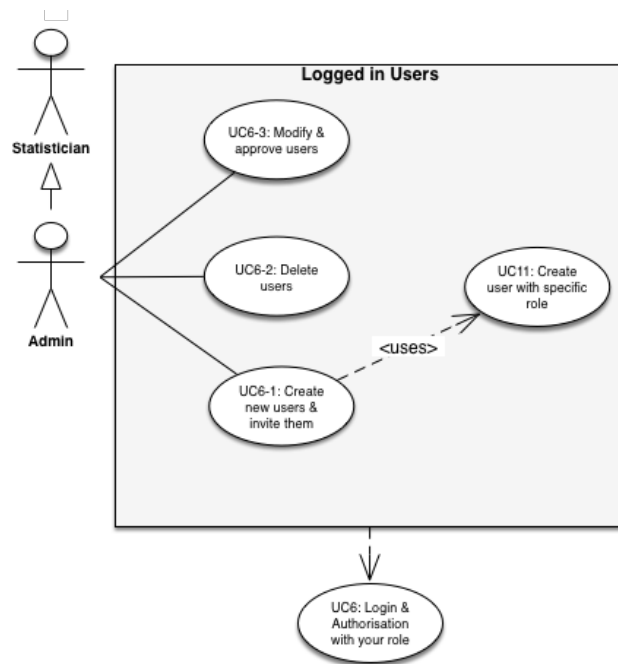


Figure 25: Use Case overview for administrators.

C Third Party Libraries

Table 6 contains all the third party libraries that have been used in the project. These are all public available and free of use. Additional information can be found for each listed **gem** under <http://rubygems.org>.

Library	ver.	Library	ver.	Library	ver.
actionmailer	(4.2.6)	faker	(1.6.3)	rails_12factor	(0.0.3)
actionpack	(4.2.6)	foreman	(0.82.0)	rails_serve_static_assets	(0.0.5)
actionview	(4.2.6)	formtastic	(3.1.4)	rails_stdout_logging	(0.0.5)
activejob	(4.2.6)	formtastic-bootstrap	(3.1.1)	railties	(4.2.6)
activemodel	(4.2.6)	globalid	(0.3.6)	rake	(11.2.2)
activerecord	(4.2.6)	haml	(4.0.7)	rdoc	(4.2.2)
activesupport	(4.2.6)	haml-rails	(0.9.0)	ref	(2.0.0)
addressable	(2.4.0)	heroku-api	(0.4.2)	responders	(2.2.0)
arel	(6.0.3)	html2haml	(2.0.0)	rootapp-rinruby	(3.1.2) ²⁴
aws-sdk	(2.4.2)	i18n	(0.7.0)	rspec-core	(3.4.4)
aws-sdk-core	(2.4.2)	jbuilder	(2.5.0)	rspec-expectations	(3.4.0)
aws-sdk-resources	(2.4.2)	jmespath	(1.3.1)	rspec-mocks	(3.4.1)
bcrypt	(3.1.11)	jquery-rails	(4.1.1)	rspec-rails	(3.4.2)
binding_of_caller	(0.7.2)	jquery-turbolinks	(2.1.0)	rspec-retry	(0.4.5)
builder	(3.2.2)	jquery-ui-rails	(5.0.5)	rspec-support	(3.4.1)
bullet	(5.1.1)	json	(1.8.3)	ruby-graphviz	(1.2.2)
bundler	(1.12.5)	launchy	(2.4.3)	ruby_parser	(3.8.2)
byebug	(9.0.5)	less	(2.6.0)	rush	(0.6.8)
cancancan	(1.15.0)	less-rails	(2.7.1)	sass	(3.4.22)
capbara	(2.7.1)	libv8	(3.16.*)	sass-rails	(5.0.4)
capbara-screenshot	(1.0.13)	loofah	(2.0.3)	sdoc	(0.4.1)
choice	(0.2.0)	mail	(2.6.4)	session	(3.2.0)
cliver	(0.3.2)	method_source	(0.8.2)	sexp_processor	(4.7.0)
cocoon	(1.2.9)	mime-types	(3.1)	shoulda-matchers	(3.1.1)
codeclimate-test-reporter	(0.6.0)	mime-types-data	(3.2016*)	simplecov	(0.12.0)
coderay	(1.1.1)	mini_portile2	(2.1.0)	simplecov-html	(0.10.0)
coffee-rails	(4.1.1)	minitest	(5.9.0)	slop	(3.6.0)
coffee-script	(2.4.1)	multi_json	(1.12.1)	sprockets	(3.6.2)
coffee-script-source	(1.10.0)	nokogiri	(1.6.8)	sprockets-rails	(3.0.4)
commonjs	(0.2.7)	orm_adapter	(0.5.0)	teaspoon	(1.1.5)
concurrent-ruby	(1.0.2)	pg	(0.18.4)	teaspoon-jasmine	(2.3.4)
data_migrate	(2.1.0)	pkg-config	(1.1.7)	therubyracer	(0.12.2)
database_cleaner	(1.5.3)	poltergeist	(1.10.0)	thor	(0.19.1)
debug_inspector	(0.0.2)	pry	(0.10.3)	thread_safe	(0.3.5)
delayed_job	(4.1.2)	puma	(3.4.0)	tilt	(2.0.5)
delayed_job_active_record	(4.1.1)	rack	(1.6.4)	turbolinks	(2.5.3)
devise	(4.1.1)	rack-mini-profiler	(0.10.1)	twitter-bootstrap-rails	(3.2.2)
diff-lcs	(1.2.5)	rack-test	(0.6.3)	tzinfo	(1.2.2)
plec docile	(1.1.5)	rails	(4.2.6)	uglifier	(3.0.0)
dotenv	(2.1.1)	rails-assets-chosen	(1.6.1)	uniform_notifier	(1.10.0)
dotenv-rails	(2.1.1)	rails-assets-jquery	(3.0.0)	warden	(1.2.6)
erubis	(2.7.0)	rails-deprecated_sanitizer	(1.0.3)	web-console	(2.3.0)
excon	(0.51.0)	rails-dom-testing	(1.0.7)	websocket-driver	(0.6.4)
execjs	(2.7.0)	rails-erd	(1.4.7)	websocket-extensions	(0.1.2)
factory_girl	(4.7.0)	rails-html-sanitizer	(1.0.3)	workless	(1.2.3)
factory_girl_rails	(4.7.0)				

Table 6: Third-party libraries and services used in the web-application.

D R-Spec Test Suite

Figure 26-31 provide an impression on the developed `rspec` tests. The whole report can be as well found on <https://github.com/sebastianzillessen/small-data-analyst/blob/master/report.html>.

The test suite, which is made out of ≈ 350 specs, results in a resonable test coverage of $\geq 85\%$ (see <https://codeclimate.com/github/sebastianzillessen/small-data-analyst/coverage> for in depth code climate details and coverage report).

Spec Test Results: Small Data Analyst

475 examples, 0 failures, 108 pending
Finished in 75.35433 seconds

AnalysesController	
GET #index	1.03840s
GET #show	
assigns the requested analysis as @analysis	0.06769s
GET #new	
assigns a new analysis as @analysis	0.03314s
POST #create	
with valid params	
creates a new Analysis	0.07126s
assigns a newly created analysis as @analysis	0.06645s
redirects to the created analysis	0.06629s
with invalid params	
assigns a newly created but unsaved analysis as @analysis	0.06513s
re-renders the 'new' template	0.06160s
DELETE #destroy	
destroys the requested analysis	0.07218s
redirects to the analyses list	0.07314s
AssumptionsController	
GET #new	0.03922s
assigns a new assumption as @assumption	
POST #create	
PUT #update	
DatasetsController	
GET #index	0.06108s
assigns all datasets as @datasets	
GET #show	0.04186s
assigns the requested dataset as @dataset	
GET #new	
assigns a new dataset as @dataset	0.04092s
GET #edit	
assigns the requested dataset as @dataset	0.04188s
POST #create	
with valid params	
creates a new Dataset	0.04739s
assigns a newly created dataset as @dataset	0.04108s
redirects to the created dataset	0.04547s
PUT #update	
DELETE #destroy	
destroys the requested dataset	0.03825s
redirects to the datasets list	0.03734s
HomeController	
GET #index	
returns http success	0.02424s
ModelsController	

GET #new	0.03247s
assigns a new model as @model	
POST #create	
PUT #update	
PreferencesController	
GET #new	0.05320s
assigns a new preference as @preference	
POST #create	
PUT #update	
QueryAssumptionResultsController	
PUT #update	
with invalid params	
assigns the query_assumption_result as @query_assumption_result	0.04155s
re-renders the 'edit' template	0.03453s
ResearchQuestionsController	
GET #new	0.03022s
assigns a new research_question as @research_question	
POST #create	
PUT #update	
AdminUsersController	
GET #new	0.06652s
assigns a new user as @user	
POST #create	
PUT #update	
creating a new analysis	0.82213s
should be logged in	
creating a new analysis	
with js support	
new analysis	
should find field input fields	2.44002s
should select attributes and create a new analysis	2.33714s
should not allow an empty analysis to be created	1.38211s
show analysis	
with 2 possible models at the beginning and as 3 not holding	
should contain possible models	1.68764s
should have 1 open query assumptions and 2 possible models	1.52727s
"No" on first query	2.68048s
Yes on first query and only one remainign (so no second answer)	3.56799s
with all 3 models beeing possible at the beginning	
Yes on first query, then CD2_explain on AS2	4.77919s
Yes on first query, then CD2_predict on AS2	5.65261s
creating a new analysis	
show analysis with a test query plot	
should show 1 possible model	3.46575s
should have 1 open query assumptions and 1 possible models	2.91121s
"No" on first query	6.07802s
"YES" on first query	5.21178s
creating a new preference	
should show research question selection first	0.88005s
creating a new research question	

Figure 26: RSpec test suite export (Pages 1-2).

should require a name	0.278945
should require a name phantoms	0.476175
should be of version 2.0.0 or higher	0.076535
As2Int	
should have 3 possible models	0.312675
should have 3 possible models	0.24475
should return all model rules as contradictions	0.323015
ExtendedArgumentationFramework:DungSolver	
should return two preferred extension	0.007305
should contain the preferred extension a1,a3	0.014495
should contain the preferred extension a1,a4	0.009765
ExtendedArgumentationFramework:Edge	
#successes_wrt	
should return the subject	0.008015
should not be accepted w.r.t {a3}	0.003655
should be accepted w.r.t {}	0.004025
ExtendedArgumentationFramework:KFramework	
#arguments	
should respond to #arguments	0.007875
should contain exactly #<ExtendedArgumentationFramework:Edge 0x007f0f0442a30 @source=#<ExtendedArgumentationFramework:Argument 0x007f0f0942328 @name="a_1">, and #<ExtendedArgumentationFramework:Argument 0x007f0f0942328 @name="a_2"> and #<ExtendedArgumentationFramework:Argument 0x007f0f09422d8 @name="a_3">>	0.006495
#attacks	
should respond to #attacks	0.005735
should contain exactly #<ExtendedArgumentationFramework:Edge 0x007f0f0442a30 @source=#<ExtendedArgumentationFramework:Argument 0x007f0f0442b20 @name="a_1">, @target=#<ExtendedArgumentationFramework:Argument 0x007f0f0442b20 @name="a_2">>	0.003865
#attacks on attacks	
should respond to #attacks_on_attacks	0.005105
should contain exactly #<ExtendedArgumentationFramework:Edge 0x007f0f073b640 @source=#<ExtendedArgumentationFramework:Argument 0x007f0f073b708 @name="a_3">, @target=#<ExtendedArgumentationFramework:Argument 0x007f0f073b708 @name="a_1">, @target=#<ExtendedArgumentationFramework:Argument 0x007f0f073b708 @name="a_2">>	0.004395
ExtendedArgumentationFramework:Argument 0x007f0f073b758 @name="a_2">>>	0.004455
should only return the attacks_on_attacks of a given source	
#edges	
should return attacks and attacks_on_attacks	0.006345
#size	
should have 3 arguments	0.006775
should match 3 arguments	0.006135
should have 2 attack	0.005515
should have one attack on attacks	0.006945
should have one attack on attacks	0.004985
should throw an error if no counterattack available	0.010665
Framework	
should have 9 arguments	0.006525
should have 7 edges	0.006035
should have 5 attacks on attacks	0.004825
should return one preferred extension	0.003265
ExtendedArgumentationFramework:Labeling	
#set	

for arguments	
should be inlabeled	0.004165
should change label to out	0.002635
should change label to undec	0.004935
With all in labelling	
a should be legally labelled in	0.005005
a should not be illegally labelled in	0.004595
b should be illegally labelled in	0.004535
b should not be legally labelled in	0.004635
should not be admissible	0.004635
With correct labelling	
a should be legally labelled in	0.005435
a should not be legally labelled out	0.003415
a should not be legally labelled undec	0.004315
b should be legally labelled out	0.005105
b should not be legally labelled in	0.003575
should be admissible	0.004805
b should not be legally labelled undec	0.004025
#clone	
should equal	0.002325
should not change properties	0.002275
more complex framework	
#clone	
should equal	0.004025
With all in labelling	
a should be legally labelled in	0.006325
a should not be illegally labelled in	0.004975
b should be in in	0.004785
b should not be legally in	0.005845
b should be illegally labelled in	0.004575
should label attack on attack in	0.004525
should not label attack on attack out	0.005515
should not label attack on attack undec	0.004605
should not label attack in	0.004455
should not label attack undec	0.005225
should label attack out	0.005905
should not be admissible	0.004015
With correct labelling	
a should be legally labelled in	0.002925
a should not be legally labelled out	0.004685
a should not be legally labelled undec	0.006115
b should be legally labelled in	0.005875
b should not be legally labelled out	0.004385
should be admissible	0.005325
b should not be legally labelled undec	0.004705
ExtendedArgumentationFramework:Solver	
with acceptability check	
should equal 6	0.003265
should equal 5	0.003195

Figure 27: RSpec test suite export (Pages 3-4).

should equal 11	0.00371s
z1to5 should be an array of arguments	0.00446s
should be acceptable wrt. [z1...z5]	0.00847s
should not be acceptable wrt. [z1]	0.00487s
with 3 models each attacking each other	
with CD1_heavy in place	
should return true for m2	0.00786s
should return false for m1 and m3	0.00607s
should return true for m1	0.00573s
should return false for m3	0.00715s
with CD1_mild in place	
should return not decidable for acceptable for any m_1 and m2	0.01050s
should return not acceptable for m3	0.00711s
with CD2_predict in place	
should return true for m2	0.00671s
should return false for m1 and m3	0.00685s
with CD2_explain in place	
should return true for m2	0.00315s
should return false for m1 and m3	0.00360s
chaining CD1_mild and CD2_predict	
should return m2 as only possible model	0.00506s
should return m2 as only possible model on second level	0.00581s
should declare m3 as not possible model	0.00398s
should declare m1 as not possible model	0.00510s
chaining CD2_predict and test	
should return m2 as only possible model on first level	0.00321s
should return m2 as only possible model	0.00603s
chaining CD1_mild only	
should return m1 as undecided model	0.01309s
should return m2 as under model	0.00956s
should return m3 as non possible model	0.00681s
chaining CD1_mild and TEST	
should return m1 as only possible model	0.00802s
should declare m3 as not possible model	0.00656s
should declare m2 as not possible model	0.01252s
AdminMailer	
new_user_waiting_for_approval	
renders the headers	0.08694s
renders the body	0.09846s
Analysis	
should respond to #private	0.08224s
should respond to #in_progress	0.07192s
should be truthy	0.00422s
should respond to #dataset	0.10139s
should respond to #research_question	0.09840s
should respond to #possible_models	0.07294s
should respond to #query_assumption_results	0.04998s
should validate that 'research_question' cannot be empty/falsy	0.05995s
should validate that 'dataset' cannot be empty/falsy	0.05535s

#start	
should eq 3	1.69263s
should respond to #start	1.61852s
should trigger start	0.05433s
should create new QueryAssumptionResults	0.94951s
should create only one QueryAssumptionResults	0.93267s
should create a QueryAssumptionResults for a1	0.91365s
should assign 2 possible models for now	1.22900s
Assumption	
assumptions in general	
should validate that 'name' cannot be empty/falsy	0.00424s
should not respond to #assumptions	0.00140s
should respond to #required_by	0.00109s
should respond to #models	0.00101s
should respond to #preference_arguments	0.00130s
assumptions and assumptions	
should be empty	0.01047s
small argumentation framework	
argument with no critical attacker should be critical true	0.05933s
argument with a critical false assumption attacker should be critical false	0.26939s
argument with a query critical false assumption attacker should be true as they should be ignored	0.05811s
should return the query_assumption	0.06214s
should return the query_assumption of a sub critical blank assumption	0.06362s
should not return the query_assumption of a sub non-critical blank assumption	0.05948s
should not return the query_assumption of a sub critical blank assumption that has a critical evaluating to false	0.07223s
argument with a critical assumption attacker should be critical true	0.30700s
single argument should be critical true	
should respond to #evaluate	0.00790s
should be truthy	0.04477s
#get_all_parents	
should return empty for single assumption	0.00853s
should return only parent	0.02235s
should return itself if there is a circle	0.03972s
should be invalid with a circle	0.02237s
should be valid without a circle	0.02191s
BlankAssumption	
general BlankAssumptions	
should respond to #assumptions	0.00195s
should respond to #assumptions	0.00153s
assumptions and assumptions	
should be empty	0.01152s
should be empty	0.00891s
QueryAssumption	
general QueryAssumption	
should not respond to #assumptions	0.00237s
should respond to #required_by	0.00209s
assumptions and assumptions	
should be empty	0.00759s
TestAssumption	

Figure 28: RSpec test suite export (Pages 5-6).

general TestAssumption	
should respond to #required_by	0.001855
should not respond to #assumptions	0.001715
assumptions and assumptions	
should be empty	0.010015
Dataset	
should respond to #user	0.014735
should respond to #columns	0.015235
should respond to #data, file	0.015505
should respond to #data	0.015775
parsing a file	
should parse the columns	0.016375
should be valid	0.015275
should have the right columns	0.017165
should have the right rows	0.016105
should call update_dataset_test_assumption_results	0.017555
should call update on the dataset_result	0.284035
DatasetTestAssumptionResult	
should respond to #dataset	0.248775
should respond to #test_assumption	0.246415
validation	
should allow nil for result	0.263475
test assumption reference	
should be possible to add TestAssumptions	0.270565
should not be possible to add Assumptions	0.253845
with test assumption	
should return the test assumption	0.246405
should trigger update method on change of dataset	0.024535
#update	
should trigger evaluate on test_assumption	0.250475
ModelOrder	
should be valid	0.126005
should respond to #preference_argument	0.119905
should respond to #models	0.096215
should respond to #index	0.093565
should be able to delete	0.101105
Model	
should respond to [name, :description]	0.010905
should respond to #analyses	0.011165
should respond to #research_questions	0.011925
should respond to #assumptions	0.011435
PreferenceArgument	
should be valid	0.077985
should respond to #model_orders	0.080935
should respond to #models	0.070765
should respond to #models_grouped	0.077275
should respond to #assumption	0.085365
should respond to #preference	0.120765
should be able to delete	0.104055

Preference	
should be valid	0.196905
should respond to #research_question	0.176165
should respond to #stage	0.155845
should respond to #user	0.162185
should respond to #name	0.159745
should respond to #int_name	0.143255
should respond to #preference_arguments	0.137105
should respond to #global	0.142835
should be able to delete	0.166595
with cdi	
should equal 2	0.188825
should return the right rules	0.200395
should return the right rules for one assumption only	0.203855
should return the right rules for one assumption only stage 1-10 registered for statisticians only	0.207305
a statistician	
should be allowed to set stage to 1	0.184415
should not be allowed to set stage to 10	0.195435
a clinician	
should not be allowed to set stage to 1	0.185635
should not be allowed to set stage to 10	0.160735
#global	
with global = true	
a statistician	
should be allowed to set global	0.162865
should not be allowed to set stage to 10	0.163865
a clinician	
should not be allowed to set it	0.191445
QueryAssumptionResult	
should respond to #dataset	0.041325
should respond to #query_assumption	0.038165
should respond to #analysis	0.035615
validation	
should allow nil for result	0.036155
query_assumption reference	
should be possible to add QueryAssumptions	0.044255
should not be possible to add Assumptions	0.039795
only one per analysis and assumption	
should be invalid to have a second query_assumption	0.078735
ResearchQuestion	
should respond to #analyses	0.001815
should respond to #name	0.001525
should respond to #description	0.002175
should respond to #preferences	0.002025
should validate that name is case-sensitively unique	0.015085
should be able to delete	0.005435
TestAssumption	
should be valid	0.014815

Figure 29: RSpec test suite export (Pages 7-8).

should trigger update on dataset_test_assumption_result	0.259815
check if the columns in the dataset are present	
should succeed	0.01721s
should not succeed if the columns are not present	0.01294s
should not succeed if there are columns in both but the assumption requires one more	0.01452s
should succeed if there are columns in both	0.01237s
Users	
GET /users	
works! (now write some real specs)	0.12589s
Analyses	
GET /analyses	
works! (now write some real specs)	0.10592s
#new /analyses	
should be success	0.10711s
should render form	0.11397s
should ask for a research question	0.10525s
should ask for a dataset	0.10653s
Assumptions	
GET /assumptions	
works! (now write some real specs)	0.09838s
Datasets	
GET /datasets	
works! (now write some real specs)	0.09746s
Models	
GET /models	
works! (now write some real specs)	0.09399s
Preferences	
GET /preferences	
works! (now write some real specs)	0.08996s
ResearchQuestions	
GET /research_questions	
works! (now write some real specs)	0.08527s
AnalysesController	
routing	
routes to #index	0.00210s
routes to #new	0.00207s
routes to #show	0.00186s
routes to #edit	0.00184s
routes to #create	0.00211s
routes to #update via PUT	0.00158s
routes to #update via PATCH	0.00136s
routes to #destroy	0.00130s
AssumptionsController	
routing	
routes to #index	0.00151s
routes to #new	0.00205s
routes to #show	0.00163s
routes to #edit	0.00213s
routes to #create	0.00168s
routes to #update via PUT	0.00153s

routes to #update via PATCH	0.00167s
routes to #destroy	0.00160s
DatasetsController	
routing	
routes to #index	0.00122s
routes to #new	0.00135s
routes to #show	0.00130s
routes to #edit	0.00130s
routes to #create	0.00111s
routes to #update via PUT	0.00121s
routes to #update via PATCH	0.00161s
routes to #destroy	0.00166s
ModelsController	
routing	
routes to #index	0.00198s
routes to #new	0.00196s
routes to #show	0.00135s
routes to #edit	0.00125s
routes to #create	0.00110s
routes to #update via PUT	0.00107s
routes to #update via PATCH	0.00109s
routes to #destroy	0.00108s
PreferencesController	
routing	
routes to #index	0.00112s
routes to #new	0.00108s
routes to #show	0.00104s
routes to #edit	0.00109s
routes to #create	0.00166s
routes to #update via PUT	0.00153s
routes to #update via PATCH	0.00133s
routes to #destroy	0.00175s
QueryAssumptionResultsController	
routing	
routes to #index	0.00202s
routes to #new	0.00184s
routes to #show	0.00142s
routes to #edit	0.00162s
routes to #create	0.00149s
routes to #update via PUT	0.00153s
routes to #update via PATCH	0.00166s
routes to #destroy	0.00158s
ResearchQuestionsController	
routing	
routes to #index	0.00131s
routes to #new	0.00148s
routes to #show	0.00124s
routes to #edit	0.00136s
routes to #create	0.00134s

Figure 30: RSpec test suite export (Pages 9-10).

	routes to #update via PUT	0.00187s
	routes to #update via PATCH	0.00128s
	routes to #destroy	0.00119s
Admin::UsersController		
routing		
	routes to #index	0.00143s
	routes to #new	0.00145s
	routes to #show	0.00112s
	routes to #edit	0.00114s
	routes to #create	0.00106s
	routes to #update via PUT	0.00151s
	routes to #update via PATCH	0.00120s
	routes to #destroy	0.00110s
admin/users/edit		
	renders the edit user form	0.11984s
admin/users/index		
	renders a list of users	0.07609s
admin/users/new		
	renders new user form	0.03216s
admin/users/show		
	renders attributes in <p>	0.03641s
analyses/index		
	renders a list of analyses	0.11410s
analyses/new		
	renders new analysis form	0.04398s
analyses/show		
	renders attributes in <p>	0.24959s
assumptions/edit		
BlankAssumption		
	renders the edit assumption form	0.16946s
QueryAssumption		
	renders the edit assumption form	0.11744s
TestAssumption		
	renders the edit assumption form	0.13093s
assumptions/index		
	renders a list of assumptions	0.04906s
assumptions/new		
BlankAssumption		
	renders the edit assumption form	0.06975s
QueryAssumption		
	renders the edit assumption form	0.04622s
TestAssumption		
	renders the edit assumption form	0.05676s
assumptions/show		
	renders attributes in <p>	0.18577s
datasets/edit		
	renders the edit dataset form	0.09453s
datasets/index		
	renders a list of datasets	0.05167s
datasets/new		

	renders new dataset form	0.02874s
datasets/show		
	renders attributes in <p>	0.05951s
models/edit		
	renders the edit model form	0.09746s
models/index		
	renders a list of model names	0.05940s
models/new		
	renders new model form	0.04910s
models/show		
	renders attributes in <p>	1.19360s
preferences/edit		
	renders the edit preference form	0.35746s
preferences/index		
	renders a list of preferences	0.37552s
preferences/show		
	renders attributes in <p>	0.27215s
research_questions/edit		
	renders the edit research_question form	0.06979s
research_questions/index		
	renders a list of research_questions	0.05031s
research_questions/new		
	renders new research_question form	0.04230s
research_questions/show		
	renders attributes in <p>	0.30257s

Figure 31: RSpec test suite export (Pages 11-12).

E Source Code

The core source code (without any configuration files and without any additional required scripts) of this project contains around 5000 lines of code, spread over ≈ 200 files. Representing this in an PDF will not really help the reader to understand the project structure and its files. Hence, these files have not been included into the project. However, the source code is available on <https://github.com/sebastianzillesen/small-data-analyst/tree/Release> and has been tagged with "Release". This tag will not be changed anymore after the submission of the thesis. The uniq SHA1 identifier for the last commit is "#ae5e3fff2cb1cf3ad7b23afdb39e7ea432cce69c".