FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

FACULTY OF BUSINESS,
ECONOMICS, AND LAW

Discovering Data Science Design Patterns with Examples from R and Python Software Ecosystem

Dmitrij Petrov

Nuremberg 2018

Discovering Data Science Design Patterns with Examples from R and Python Software Ecosystem

by

Dmitrij Petrov
Student ID: 21998127

A thesis submitted in partial fulfilment
of the requirements for the degree of
Master of Science
(International Information Systems)
at the University of Erlangen-Nuremberg
March 2018

Thesis Committee:

Prof. Dr. Freimut Bodendorf, Chair of Institute of Information Systems II, FAU
Research Assistant Isabella Eigner, Institute of Information Systems II, FAU
Research Assistant Alexander Piazza, Institute of Information Systems II, FAU

© Dmitrij Petrov 2018

Some Rights Reserved.

This work is licensed under the Creative Commons Attribution-ShareAlike 4.0 International License. To view a copy of this license, visit https://creativecommons.org/licenses/by-sa/4.0/.

STATUTORY DECLARATION

Ich versichere, dass ich die Arbeit ohne fremde Hilfe und ohne Benutzung anderer als der angegebenen Quellen angefertigt habe und dass die Arbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen hat und von dieser als Teil einer Prüfungsleistung angenommen wurde. Alle Ausführungen, die wörtlich oder sinngemäß übernommen wurden, sind als solche gekennzeichnet.

I assure that this thesis is a result of my personal work and that no other than the indicated aids have been used for its completion. Furthermore, I assure that all quotations and statements that have been inferred literally or in a general manner from published or unpublished writings are marked as such. Beyond this I assure that the work has not been used, neither completely nor in parts, to pass any previous examination.

March 27, 2018 in Nuremberg, Germany _____

Dedicated to my parents,

my thesis committee,

my reader,

and myself.

ACKNOWLEDGEMENTS

First and foremost, I would like to express my very great appreciation to my parents who allowed me to pursue my dreams, in many cases by sacrificing their own. Hearing their countless advices and having a spiritual and financial support since my childhood, I hope that this graduate thesis will be rewarding to them the same way as it has been to me.

Even more importantly, I would like to thank Prof. Dr. Freimut Bodendorf, M. Sc. Isabella Eigner and M. Sc. Alexander Piazza – my research co-supervisors – for their academic supervision, guidance, encouragements and critiques of this work.

Additionally, I am very grateful to Dr. John Morley of Manchester University in creating the *Academic Phrasebank* – an invaluable resource for all academicians.

Finally, I owe a debt of gratitude to the scholars at the University of Michigan in the USA who have developed a LaTeX template that I have used, with adjustments, for my dissertation.

ABSTRACT

Background: A variety of software design patterns in the computer science are documented assisting practitioners to reuse best solutions to commonly occurring problems. However, lacking researchers' attention so far, design patterns were not yet formulated for a newly established *data science* field which aims to extract actionable knowledge from data.

Objective: From the perspective of data scientists and other engineers, the study set out to discover and understand *data science design patterns*. Consequently, by supplementing such candidates with R and Python code examples utilizing packages from their software ecosystems, the objective was to develop *Data Science R and Python Toolkit Matrix* as well.

Method: Based on the *data-driven design pattern production* methodology, the research first gathered relevant and contemporary source material. Later, through a general inductive approach, data were coded resulting into emergence of an exhaustive list of key themes from which a group of pattern candidates was described according to a defined template. The purposeful sampling of utilities from CRAN and PyPI repositories further contributed to developing above mentioned matrix too.

Results: In this study, ten data science design patterns were formalized and *Data Science Toolkit Matrix* was derived consisting of thirty-two R and Python tools deemed capable of addressing frequently occurring problems faced by practitioners when analysing data large and small.

Limitations: Thesis' validity might have been hampered by the collection and interpretation of sources on which the study relied on to discover design patterns. To mitigate, a workshop with experts from the field could improve patterns' quality and gain their greater validation. From the perspective of the toolkit matrix, the work focused only on tools for two-dimensional data, and hence omitted technologies for computer vision, all of which should be addressed in the future research.

Conclusion: The thesis has contributed to the study of design patterns by specifically focusing on data science domain while considering relevant R and Python tools from their software ecosystems. For aspiring data scientists, this work laid out a foundational ground by providing them an overview of best practises for dealing with typical issues in data collection, manipulation and visualization. As a result of identified design patterns and R and Python utilities, findings also help to better understand what other programming languages such as Julia need to offer and support in order to become widely used for the knowledge discovery purposes.

Key Words: Software ecosystems, design patterns, data science, R, Python, FOSS, packages

TABLE OF CONTENTS

LIST OF APPENDICES

Appendix

LIST OF FIGURES

Figure

LIST OF TABLES

Table

LIST OF CODE LISTINGS

Listing

LIST OF ABBREVIATIONS

3D2P Data-Driven Design Pattern Production

API Application Programming Interface

BI Business Intelligence

CPU Central Processing Unit

CRAN The Comprehensive R Archive Network

CRISP-DM Cross Industry Standard Process for Data Mining

DS Data Science

DSTM Data Science R and Python Toolkit Matrix

DW Data Warehouse

ETL Extraction-Transformation-Loading

FOSS Free(/Libre) and Open Source Software

HTML Hypertext Markup Language

GIA General Inductive Approach

GoF Gang of Four

GUI Graphical User Interface

IDE Integrated Development Environment

IT Information Technology

KDD Knowledge Discovery in Databases

KPI Key Performance Indicator

ML Machine Learning

OLAP On-line analytical processing

PLoP Pattern Languages of Programs

PyPI The Python Package Index

RAM Random-Access Memory

SECO Software Ecosystem

SLR Systematic Literature Review

SVM Support Vector Machine

SQL Structured Query Language

UI/UX User Interface & User Experience

UML Unified Modeling Language

This page is intentionally left blank.

CHAPTER I

Introduction

In the following pages, the reader is introduced to this thesis. At the beginning, the subject matter is delineated, followed by laying out the problem statement, objectives, methodology and delimitations of the endeavour. Finally, document's structure is described too.

1.1 Overview

A term *pattern* embodies a multitude of concepts which can be used in a variety of contexts, for example in nature to describe wavelike shapes on sand dunes in the desert. Besides its use in the art or in the fashion for decorative purposes, patterns in mathematics are commonly known as *fractals* which are never ending geometric shapes that exhibit the "*same 'type' of structure on all scales*" (*self-similarity*; Weisstein 2018).

From the *knowledge discovery in databases* (KDD) perspective, diverse algorithms are applied to uncover "*interesting, [insightful,] unexpected and useful patterns in*" data (Fournier-Viger 2013; Witten et al. 2016; Goebel and Gruenwald 1999). In accordance with a *data pyramid*, ultimately turning such information into a wisdom (Rowley 2007). Yet the focus of this study considers the meaning of patterns being first formulated in a book entitled *A Pattern Language* by Alexander et al. (1977).

Since Christopher Alexander's profound publication in the field of architecture, patterns have become a topic of a great interest and have had a large influence in a wide range of areas as well. Perhaps the most important classical literature work appeared some twenty years later by Gamma et al. (1994, p. 11) who introduced twenty-three *design patterns* for developing "*reusable object-oriented software*" applications. The authors known as *Gang of Four (GoF)* described their abstract approaches as a "*recurring solution to a standard problem*", essentially concerned with "*a way to solve a specific problem of design*" (Schmidt et al. 1996, p. 37; Ulrich 2006).

Importance of Patterns

Over, the past four decades, a growing body of research in the computer science has recognised the importance of patterns as useful techniques facilitating for instance better communication and collaboration between engineers and their managers (Schmidt et al. 1996). Rooted

and validated in the practise, they are "*identified and verified through careful observation*" providing a reliable, "*orderly resolution of software development problems (…) and help new developers ignore traps and pitfalls that have traditionally been learned only by costly experience*" (Ulrich 2006; Schmidt et al. 1996, p. 37). Therefore, their adoption for "*documenting complex architectural designs*" gave them a firm place during the phases of software analysis, design and implementation (Nadschlager 2017, p. 47; Giridhar 2016).

Data Science ∧ Design Patterns

Stepping aside for a moment, companies nowadays collect digital information that over time has grown both in volume and its variety. In order to realize its full potential, in cases like preventing financial fraud, new technologies are used to transform captured data from various internal and external locations with a goal to extract valuable knowledge and gain a wisdom for business operations through data visualizations and storytelling (Henke et al. 2016). Thus, eventually, making corporate processes, products and services more data-driven. However, unfortunately as it has been documented in countless instances, when raw data are translated into actionable insights for employees, reoccurring issues are often faced (Holley et al. 2014; Hakeem 2017).

Subsequently, one of the challenges encountered by many researchers, *data scientists* and other *big data* engineers has been how to effectively share among each other best practises and gained experiences to frequently confronted obstacles. The answer, which has been suggested, involves applying design patterns not only for communication and documentation purposes but also for the entire knowledge management. This due to their capability of addressing hurdles efficiently by means of having a narrow scope with defined structure that captures a problem and offers a solution (Hakeem 2017; Dearden and Finlay 2006).

Research Gap

As later shown in the **chapter II**, to date, a significant pattern research has been carried out in the education, sociology or in the multidisciplinary field of information technology (IT). There, patterns have been examined for the enterprise architecture or targeting specific software applications such as *Hadoop* (Fowler 2002; Miner and Shook 2012). Surprisingly, however, little research has formally studied design patterns in relation to data science (DS) or machine learning (ML) – the research gap that is identified and addressed. Indeed, to the best of author's knowledge, design patterns have been so far undocumented in a broad DS domain which is the main focus of this work.

Arguably, one of the reasons has been an ambiguity of such umbrella term DS and its dynamic nature which is manifested in the ever-evolving technologies and approaches to business analytics (Shan et al. 2015). Therefore, the goal of this thesis is to systematically discover and describe such general, "*time-tested design solutions*" to common problems that appear during the data analysis (Heer and Agrawala 2006, p. 853). The significance of this subject matter is also underscored by a forthcoming book entitled *Data Science Design Patterns* by Morley (2019).

To further complement the intended research, in the past decade academicians and practitioners have tried to determine what programming language and its ecosystem is "more valuable" to learn, "more powerful" to work with and generally "better" to use for KDD purposes (Ma 2016; Wayner 2017; Theuwissen 2015; Lee 2015). According to different questionnaires, forum discussions and rankings, there seems to be an agreement that two programming languages – R and Python – are especially important for DS (Muenchen 2018). R ∧ Python

In this study, firstly, the objective is to devote an attention to the research gap faced by the interdisciplinary talented practitioners when they conduct typical analyses of data. Through identifying and describing best practises in a design pattern form and being consistent with authors such as Ezust and Ezust (2011), *pattern candidates* are then supplemented with R and Python code examples and tools from their ecosystems. These utilities can be chosen by a next generation of quantitative analysts to support their work on data-driven knowledge discoveries (Mosaic 2014). As a result, it also permits to gain a better understanding of what computer programs from two DS environments are available to solve domain specific hindrances. Intentions

Because the current research has so far lacked surveys of applications from either programming landscape, the second aim of this inquiry is to develop *Data Science R and Python Toolkit Matrix*, henceforth referred to as DSTM. This can be used as a guideline by all interested stakeholders to obtain a larger overview of applications capable of addressing their needs during the analysis of continuously growing data.

Consequently, this thesis intends to have an impact not only by creating a reference point for DS design patterns but also offering R and Python utilities developed in mind with them.

1.2 Problem Statement

The difficulty being dealt with is that while numerous design patterns for object-oriented programming and enterprise IT architecture already exist, no previous studies were identified looking at patterns in the field of DS and data analytics. Indeed, due to a lack of such formalized approaches supplemented with R and Python code examples, practitioners often struggle to fully leverage the potential of presently known solutions to commonly occurring problems. As a result, causing for instance fragmentation because of creating own programs and reinventing the wheel instead of identifying applications from well-developed ecosystems.

1.3 Objectives

Even though *Goal-Question-Metric* technique is most commonly applied in the software engineering for specifying "*a measurement system*", the overall research objective of this work

can be described in accordance with it too (Basili et al. 1994, p. 2). As Koziolek (2008) explains, goals are stated on the conceptual level and operationalized by respective study questions that use objective and subjective data to answer them. In short, the guiding principle throughout this thesis is formulated:

> The goal is to formalize and expand the understanding of DS design patterns and available free and open source (FOSS) tools from R and Python ecosystem that can address reoccurring problems in the data analysis. All this from a perspective of data science beginners as well as experienced professionals, both of whom aim to derive a value from raw information in the enterprise contexts.

What is to be achieved?

As it was outlined, attempting to advance the research gap, this study broadly formulates three central and open-ended questions that it intends to investigate:

RQ1: How do scientists understand the concepts of *software ecosystem*, *data science* and *design pattern*?

RQ2: Based on the additionally reviewed literature, which data science-oriented design patterns can be recognized?

RQ3: Surveying R and Python FOSS ecosystems, what are the specific tools that solve common problems when discovering knowledge from data?

The main outcome of this work, besides establishing an overview of key terms, is an identification and description of DS design patterns that can be employed for typical KDD tasks (Goebel and Gruenwald 1999). Concurrently, for each pair of a problem and solution, code examples are provided from sampled software tools in order to understand how both landscapes support users during a life-cycle of information discovery.

Targeted Audience

The primary target audience of this work are DS newcomers to the field. Thus, the choice of a tailored language that should make it easy for them to understand each pattern even if they are not entirely familiar with related DS terms. Nonetheless, the results of this study ought to be useful to already experienced practitioners as well because it facilitates a better comprehension of best practises in association with key FOSS-based R and Python packages and frameworks, henceforth referred to as *tools*. Notwithstanding this group, big data engineers and other experts such as user interface and experience (UI/UX) designers can benefit from the used terminology in this work too, when communicating with IT professionals as well as project managers.

Through documenting well-proven approaches, the ambition is to provide a common vocabulary to help users analyse data more efficiently and effectively, thus avoiding common

pitfalls in the DS process and inventing "sophisticated new solutions" instead of leveraging simple and existing ones (Landset et al. 2015). With this in mind, the contribution to a small body of existing research carried out in the subject of domain-specific design patterns and software ecosystems (SECO) of programming languages, all being at the cornerstone of this work, is following:

1. Analytical *design pattern candidates* are discovered to illustrate proven solutions to repetitive obstacles which are faced by data scientists allowing them to reuse already well-known practises and techniques.

2. A state-of-the-art survey is conducted into R and Python software ecosystems seeking to record relevant open source programs in the toolkit matrix which may act as a reference point to an initial perspective into both universes for insights discovery from data.

Given the observed research gap, the objective of this work is to pursue a higher-level perspective where DS design patterns shall on the one hand generally focus on various stages of KDD process while at the same time they are supported with practical examples from R and Python ecosystem. To the best of author's knowledge, this work is first of its kind which provides a comprehensive examination into domain-specific design patterns by putting them in the context of two programming ecosystems.

1.4 Research Methodology

A brief outline of the methodology is introduced next which together with a research design is extensively explained in the **chapter III**.

This graduate thesis follows an *interpretivist philosophical* approach to science in which qualitative methods of inquiry are used to understand the context and answer research questions, see also Table 1 (Myers 1997; Saunders et al. 2015). Having an examination that is descriptive, narrative and investigative in nature, the aim is to stay more flexible, be exploratory and not being bounded with strict formalities of purely quantitative methods. Hence, allowing to potentially document impressions during data collection and analysis whereby, at first, common obstacles and their solutions are observed and formulated to which specific programs from both open source ecosystems are collected and presented (Myers 1997).

Fundamentally, it is necessary to understand key terms used in this study through reviewing a literature on *software ecosystems*, *data science* and finally *design patterns* themselves. Once these concepts are defined, what comes next is an inductive discovery of DS design patterns, essentially theoretical artefacts. For this, Inventado's and Scupelli's (2015) *data-driven design pattern production* (3D2P) methodology is followed where diverse sources of information are

3D2P ∧ GIA

prospected, namely collected and screened. Subsequently, in the *pattern mining*, by applying *general inductive approach* (GIA), sources are thoroughly inspected, coded and interpreted in order to develop prevailing themes (Thomas 2006; Myers 1997; Seaman 1999). At last, by following guidelines of Meszaros and Doble (1997), Wellhausen and Fiesser (2011), and Douglass (2002), pattern candidates are described using a pattern template which demonstrates a set of wisdom in a structured format and an easy-to-understand form.

DSTM

After outlining a design pattern candidate, in line with the third research question, a sample of relevant R and Python packages and frameworks is *purposefully* identified and presented in code snippets (Jansen 2010; Patton 2015; Trochim 2006). Because of such concurrent process, the knowledge acquired leads to developing DSTM that visualizes data science-related software applications. In this matter, one can inspect software landscapes of two formal languages within the realm of formalized design patterns and KDD steps.

1.5 Delimitations

To stay within the boundaries, it is necessary to further clarify and restrict this work due to facing time and scope constrains. Among others, these manifest into not pursuing to identify a final and exhaustive list of design patterns, but instead, rather a smaller group of pattern candidates that are most important to learn about.

Anti-patterns

This research focuses exclusively on design patterns which provide time-tested, reusable solutions and best practises to typical DS tasks during the analysis of data large and small in volume and variety. Therefore, their counterparts, the anti-patterns describing "*poor design practice together with descriptions of how the design could be repaired*" are naturally omitted (Dearden and Finlay 2006, p. 59).

FOSS

Moreover, a restriction is made to demonstrate sample examples by only collecting (standalone) open source programs that are found in the R and Python ecosystem and which are not build into languages themselves. As such, proprietary products and on-line services – developed for example by business intelligence (BI) vendors like Informatica or Qlik offering commercial off-the-shelf software used for extraction-transformation-loading (ETL) pipeline, data management or visualizations – are not considered. Indeed, this research targets strictly FOSS which is accompanied by an open source license to the source code and has its "*machine instructions available publicly, in the plain text*" – a distinguishing feature to the closed source software (Petrov 2016, p. 1). Although there are differences between *free* and *open source* software, because of the insignificance to this thesis, both terms are used interchangeably (Petrov 2016). Consequently, accepting only such type of software, identified packages and frameworks should fit KDD purposes and be available (or with a vision to be released there) on two repositories named *The*

Comprehensive R Archive Network (CRAN) and *The Python Package Index (PyPI)*.

Tabular data

Furthermore, a limitation is put to survey applications which work with what Liu et al. (2015, p. 2) have described as "*table-based data*" or in other words "*classical format, [where] a dataset consists of a set of N [observations] (…) with s features*" (Mikut and Reischl 2011, p. 3). These tools supporting two-dimensional type of structured data have been fundamental in the BI which – until the arrival of big data – has exclusively worked with dimensional modelling and processing of transactional information. Due being developed since at least 1980s, such programs have been well established both commercially as well as in open source communities (Goebel and Gruenwald 1999). As a result, software applications targeting computer vision and operating with images, video or other type of high-dimensional data are not taken into consideration in this investigation. Although being of significance, they are often field specific to biology, speech or object recognition (Mikut and Reischl 2011). From this follows that software that is nowadays applied for instance in the *deep learning* goes initially beyond the study's scope.

Difference to other studies

Because of such delimitations, the research differs from others in two central items, namely having a distinct and a narrow focus. Even though there is a growing body of literature concerned with a concept of design patterns, the examination to date into related areas of DS has been scarce, scattered across separate locations and targeting different themes. Thus, it is attempted to provide a holistic look into DS domain by putting a spotlight on the most influential tasks and problems arising during the analysis of information.

Secondly, solutions to frequent obstacles are not only presented on the theoretical level through their rather generic conceptualization but importantly also put into a practical perspective with specific examples that use R and Python tools from their software ecosystems. By answering the third study question, it allows to gain an initial perspective into their technical landscapes via Data Science R and Python Toolkit Matrix.

1.6 Document Structure

The desire of this work is to present *data science design patterns* with R and Python examples in easy-to-understand format. Therefore, together with this introduction, the overall structure consists of five chapters, see Figure 1. A brief overview of each part of the document is provided next.

Chapter II: To resolve the first research question, the thesis begins by attempting to clarify *software ecosystem* with relation to two programming languages. It then goes on to provide explanation of other major terms, namely *data science* and *design patterns*.

Chapter III: Subsequently to the understanding of two formal systems and other phrases,

an outline of the methodology for discovery of design patterns and respective tools is made.

Chapter IV: Immediately next and addressing two research questions, pattern candidates are established, described and a survey of relevant FOSS utilities is conducted. As a result, Data Science R and Python Toolkit Matrix is developed too.

Chapter V: Finally, results are summarized and conclusions are made with drawing together key findings of the research by critically discussing it.

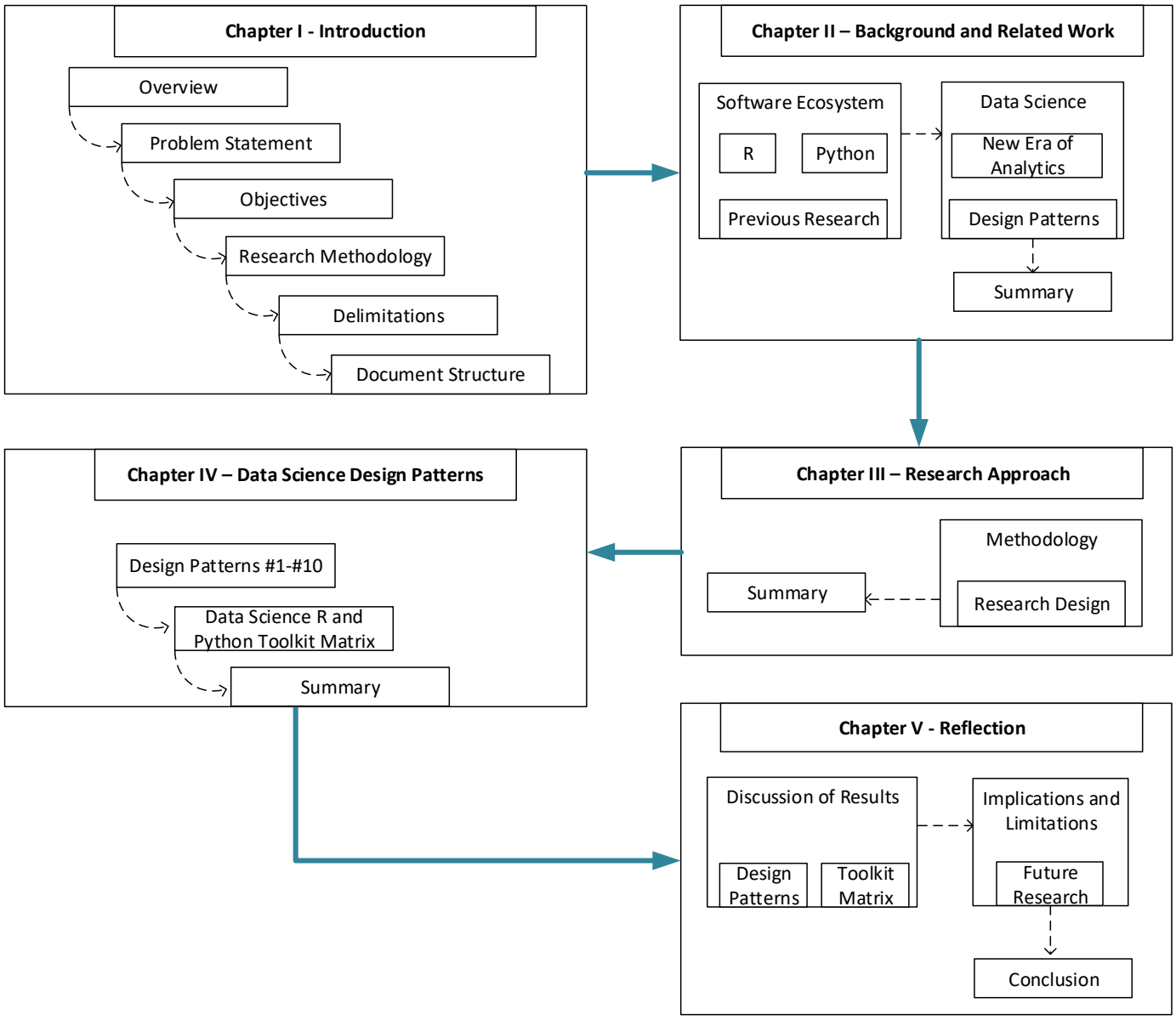


Figure 1: Illustrates thesis' structure in accordance with slightly renamed chapters up to level three.

Table 1: Presents research questions, methods and data sources used.

Research Questions	**Methodology**	**Data Sources**
RQ1: How do scientists understand the concepts of *software ecosystem*, *data science* and *design pattern*?	Literature review	Reputable journal articles and conference proceedings found in Scopus and Google Scholar databases
RQ2: Based on the additionally reviewed literature, which data science-oriented design patterns can be recognized?	Qualitative: Utilize 3D2P methodology to mine patterns using a general inductive approach	Similarly, a variety of information sources such as reports, books and grey literature is collected and analysed
RQ3: Surveying R and Python FOSS ecosystems, what are the specific tools that solve common problems when discovering knowledge from data?	Qualitative: Conduct a purposeful sampling	Primarily from CRAN and PyPI repositories

CHAPTER II

Background and Related Work

This chapter, being narrative and descriptive and subdivided into two main sections, is designed to clarify this work and develop an in-depth understanding of the subject matter. Due to later sampling R and Python applications from their landscapes, a concept of *software ecosystems* is first elucidated. Next, before concluding with a summary, it is aimed to describe in adequate detail available information on *data science* and associate it with *design patterns*.

2.1 Software Ecosystems

Term *ecosystem* has been introduced in the early 20th century by a botanist Arthur Tansley. With an origin in ecology, it has been used to denote "*the physical and biological components of an environment when considered in relation to each other as a unit*" (Franco-Bedoya et al. 2017, p. 4). Accordingly, many authors have confirmed that although having differences to a biological ecosystem, the software one "*can be in some way mapped to phenomena in nature*" too (Dhungana et al. 2010, p. 101; Cusumano and Jansen 2013). For instance, landscape's growth and decline due to changing conditions or a need for "*a continuous input of energy in the form of new development or maintenance*" (Dhungana et al. 2010, p. 98; Mens and Grosjean 2014).

Business Ecosystem

In 1999, James Moore has been first who extended the phrase to the *business ecosystem* describing it as "*an economic community supported by a foundation of interacting organizations and individuals – the organisms of the business world*" (Moore 1999, p. 45; Cusumano and Jansen 2013). Simplified, it is a dynamic, constantly developing structure which revolves around producing goods and services to the benefit of customers and where three core characteristics have been identified. For one, a network of various stakeholders such as distributors and producers "*coevolve their capabilities and roles*" together by creating leaders, niche and bridge players – all moving towards the same vision (Cusumano and Jansen 2013, p. 45; Li 2009). Secondly, it is an important notion of a *platform* that gives a common ground for interaction between community members, examples of which are Microsoft Windows operating system, a vehicle

manufactured by Ford or Eclipse integrated development environment (IDE; Moore 1999). Consequently, these platform holders become *keystone companies* in the ecosystem because they can exercise their power whereby others will depend on them. Lastly, Li (2009) has explained that a business setting allows synergic cooperation between stakeholders that results into pursuing shared interests together, while if acting alone they might be individually never successful.

Software Ecosystem

Building upon Moore's understanding, around 2000 a conception of *digital business ecosystem* has emerged and this gave birth to a *software* one which is considered by some to be its subset (Peltoniemi and Vuori 2004; Plakidas et al. 2017). Being a "*very wide and arguably complex*" phrase, it embodies a multitude of concepts that have been proposed by academicians (Manikas 2016, p. 29). In principle, however, SECOs "*are made of software vendors, suppliers, and users*" who are put into a socio-economic context to interact with each other on a familiar technology that provides "*possibilities for the actors to benefit from their participation*" (Manikas and Hansen 2013, p. 1298; Christensen et al. 2014).

Since the introduction, scholars have significantly advanced the understanding of software ecosystems. While the research took off approximately in 2005, only eight years later Manikas and Hansen (2013) have conducted a large scale systematic literature review (SLR) of ninety papers from the field published between 2007 and 2012 – a fundamental study in this research (Manikas 2016). Trying to answer questions as to how was the term defined or what were reported results, scientists have concluded that not only there has been "*little consensus on what constitutes a software ecosystem*" per se but also "*little research (…) [has been] done in the context of real-world ecosystems*" (Manikas and Hansen 2013, p. 1294).

Classification

Besides attempting to define rather nebulous term, literature has dealt with various categorization efforts. In another significant study, Finkelstein et al. (2009) has identified several SECO challenges and argued that vendors should "*separate ecosystems in three levels*":

(a) software *ecosystem level* (for example developing strategies and policies on how companies should cooperate in the environment to maximize their profitability),
(b) software *supply network level* (for example establishing relationships with buyers and suppliers through promoting events) and
(c) software *vendor level* (for example forming blueprints and guidelines for product line planning, knowledge management and software extensibility).

As a consequence, levels should be addressed individually in order to build, strengthen and maintain company's position and role in the engaged software landscape (Christensen et al. 2014, p. 1477).

On the other hand, Cusumano and Jansen (2013, p. 55) have produced and applied a classification model "*to identify four different classes of software ecosystems*". Scientists have called their factors:

(a) the *underpinning (base) technology* (for example Apple iOS platform),
(b) *coordinators* (where it is controlled by a private company),
(c) *extension markets* (offering its users a commercial depository) and
(d) *accessibility to an ecosystem* (paid submissions to the App Store).

After governance tools have been described, authors have contributed by proposing a model for environment's health preservation and improvement (Cusumano and Jansen 2013). In fact, this area of ecosystem's health and its impact on the quality has been a significant sub-domain of SECO that was extensively researched by studies including Iansiti and Levien (2004), Hartigh et al. (2006), and Jansen (2014).

Bosch (2009) has further provided an important "*basic categorization between ecosystems centred on operating systems, applications, or end-user programming*". Besides, differentiating them according to the platform "*(desktop, web, mobile) the ecosystem is deployed on*" (Plakidas et al. 2017, p. 4).

Because SECOs are a part of information systems research, they are closely related to other areas too. They "*include [organizational] theories and methods from a variety of different fields such as software engineering (…) or network analysis*" (Barbosa et al. 2013; Manikas 2016, p. 29).

While covering a ***platform-centric*** perspective which "*highlights technical and social aspects of a set of software projects, technical platforms and communities*", Franco-Bedoya et al. (2017, p. 3) have put SECO into a relationship with FOSS. Subsequently, the phenomenon of ***FOSS ecosystem*** has been defined as "*a SECO placed in a heterogeneous environment, whose (…) keystone player is (…) [a FOSS] community around a set of projects in an open-common platform*" (Franco-Bedoya et al. 2017, p. 24). In their article, authors have systematically mapped the current state of FOSS ecosystem research and have stated that even though FOSS has been playing a strategic role in many public and private sectors, the body of knowledge about its relationship within a holistic universe is still scarce and in its infancy (Franco-Bedoya et al. 2017; Petrov 2016).

Key Components

Although definitions have varied among researchers, two fundamental features could be conceptualized across all studies (Manikas and Hansen 2013). On the one side, it is a network of actors like individuals and organizations with different roles, variable business structures and models (Joshua et al. 2013). On the other hand, it is a common interest in using a specific technology (Christensen et al. 2014). These are then supported by establishing interactions, the connecting relationships that include promoting activities during conferences, sharing openly development resources or providing training and consulting services (Manikas and Hansen 2013; Dhungana et al. 2010). All of which further grow and strengthen the ecosystem, building a sustainable environment.

Perspectives

Sketched out previously, in terms of SECO's views, it has become a commonplace to distin-

guish a *platform-centric* angle which is technologically oriented dealing with software projects and a *business-centric* one that sees the ecosystem rather as a socially-oriented "*network of actors (…) and companies*" interacting with each other (Franco-Bedoya et al. 2017, p. 3; Constantinou and Mens 2017). For the research goals, by studying distinct definitions and explanations of aforementioned concept, the term is referred to as following:

> **Software ecosystem** is a symbiotic relationship between a set of different actors and a collection of software projects that operate on a common technological platform as a unit. Hence, evolving together and sharing the same information, environment, resources and group of users and developers whose goal is to address their universal interests and needs.

To summarize, software projects are increasingly co-developed in parallel, becoming more interdependent on each other, and thus attracting new participans (Constantinou and Mens 2017; Mens and Grosjean 2014). Within the forming community, resources are combined to experience the outcomes collectively by benefiting from economies of scale and diversity of members, resulting into achieving stated objectives faster.

With regard to widely used programming languages, these exhibit strong relationships between various stakeholders which manifests into the quality and maturity of their SECOs. In the case of R and Python, both environments extending base languages have also been a critical factor and if not *the one* why two formal systems have become popular in the DS and ML.

2.1.1 The R Project

R being a "*system for statistical computation and graphics*" began as a research project in 1993 lead by Ross Ihaka and Robert Gentleman at the University of Auckland in New Zealand (R Core Team 2018b,c,d; Smith 2016). Heavily influenced by two programming languages designed in the 1970s named S and Scheme, scientists have adopted a syntax of the former one and continued to maintain backwards compatibility with it. Additionally, investigators' aim was to modernize it for purposes of data analysis, and thus "*the underlying implementation and semantics [were already] derived from Scheme*" (Ihaka and Gentleman 1996, p. 299; Ihaka 2009). For this reason, R is oftentimes called a dialect of S, a modern implementation in a new, vastly improved and enlarged language (R Core Team 2018a).

Naturally, R, formally governed by R Foundation that supports its development overseen by *R Core Team*, can be extended by wrapping a set of functions (its main building blocks), documentation, data and tests into a unit of distribution called *package* (Smith 2016). As a result, this can be uploaded to software repositories such as CRAN[1] being the largest central

CRAN

[1] https://cran.r-project.org – The Comprehensive R Archive Network

extension market and from which R community can download and use developer-contributed tools. In December 2017, because of language's growing popularity, CRAN has reached a milestone of providing 12 000 packages on the network with many more in development at *R-Forge*, *GitHub* and available for example on *Bioconductor* that focuses only on utilities for genomic data (Smith 2017; German et al. 2013; Tippmann 2014; Bengtsson 2017).

Association with Data Science

Supporting multiple programming paradigms including object-oriented and mainly the functional style, R has become "*the lingua franca of statistical computing [(…),] reproducible statistical research*" and one of the dominant forces in field of data analytics (Everitt and Hothorn 2006, p. 7). In part, this has been attributed to CRAN with a vast number of packages that provide powerful yet flexible applications, for instance for data visualizations or imputation of missing data (Plakidas et al. 2017). Likewise, being an interpreted language that can interface complied ones (C++ or Fortran), R executes various operations on data sets by storing them in Random-Access Memory (RAM) instead of on a hard-disk (Kane et al. 2013; Hornik 2016; Smith 2013a). Last but not least, R specifically targets scientific and statistical computing and having its roots in the academia, it is being used across various disciplines (Grolemund and Wickham 2018; Brunner and Kim 2016). Among others, in the biostatistics, finance and psychometrics.

Additionally, even though not being a novelty compared to other languages, in the *interactive mode* user types in various R expressions into its command line shell (R console), where they are parsed and evaluated by the interpreter and results are immediately returned (so-called read-evaluate-print-loop; Smith 2009). This allows to quickly iterate on manipulating the data set and applying various functions to it. Although "*by default R is command-line driven*", it naturally supports the *batch mode* as well where R scripts can be created and executed in IDEs, *StatET* or *Visual Studio* to name a few (Pabinger et al. 2014, p. 29).

Dynamically Typed Languages

Together with Python, both languages are *dynamically-typed* where it is not necessary to indicate if a variable is of a data type integer or a string, see listing 1 for an example (Montanaro 2012; Lutz 2013). Indeed, because typing is linked with a value rather than a variable, developers can change variables to diverse types (Zumel and Mount 2014). On the contrary, *statically-typed* languages require indicating data types and once a variable has been set to a specific type, it cannot be changed due to being associated with a variable rather than the value. The reason being mentioned is that the choice of a typing system has important consequences for the programmer, prominently the beginner (VanderPlas 2016). While the dynamic typing allows programs to be more flexible and compact, and thus these languages are perceived easier to start with, a disadvantage is having some evidence suggesting it negatively affects the runtime efficiency because of a type interference and prevents detection of programming errors by reason of decreased readability (Ousterhout 1998; Cady 2017).

R is nowadays used in many research fields as it claims to have a mature ecosystem of packages, which are "*developed primarily by non-software engineers*", and support provided by keystone players – notably RStudio Inc. that develops arguably the most prominent and homonymously named IDE (German et al. 2013, p. 244). Nonetheless, there are two important shortcomings that various groups of practitioners often talk about. Being created and maintained largely by statisticians and mathematicians, it is considered to be a domain specific formal system where in addition commonly cited weakness is difficult to pick-up and somewhat inconsistent syntax, resulting into having a steeper learning curve (Rickert 2010; Muenchen 2017). Moreover, R and Python are both – in their native implementations – single-threaded programming environments (Walkowiak 2016). Therefore, even though third-party alternatives such as Microsoft R Open or packages exist supplementing some parallel functionality and taking advantage of multi-core central processing units (CPUs), there are still limitations when compared to C++ and its multithreading capabilities (Lim and Tjhi 2015). Drawbacks

2.1.2 The Python Language

Python was conceived by Guido van Rossum in the 1980s when he was working at the Dutch research institute gaining his first experience with a programming language called ABC (Perez et al. 2011; Venners 2003). By advancing his knowledge during its development and feeling inspired by others, a decision was taken to apply some design principles when creating a new one (Peters 2004).

The goal was set to develop a formal system which would "*serve as a second language for people who were C or C++ programmers*" along with focusing on code readability, software quality and developer productivity (Venners 2003; Sanner 1999; Lutz 2013). As a result, Python's syntax is "*a remarkably simple and elegant*" to follow making the language overall easy-to-read and easy-to-write (Sanner 1999, p. 3). To guide the core team during its further improvements, many of these design principles were later summarized in the philosophical text named *Zen of Python* (Peters 2004). Vision

Being specified in the *Python Language Reference*, several compatible implementations have been developed with *CPython*, written in C, being a referenced one as well (Python Core Team 2018c,d). Additionally, there are for instance *PyPy* or *Jython* – where the former is focusing on performance providing just-in-time compiler and parallelism while the latter "*compile[s] Python source code (…) to provide direct access to Java components*" allowing Python to "*script Java applications*" as much as CPython allows to do the same with C(++) (Lutz 2013, p. 34).

Even though first publicly released in 1991, only starting with version 2.x published in 2000 Python has grown to become a popular programming choice by being supported through IDEs and nowadays covering many use cases in the computer vision, Internet of Things and offering Evolution

```r
# in R (and in Python - not shown here), types are dynamically inferred
> exA <- 5       # assign an integer value to a variable
> print(exA)
[1] 5
> exA <- "hello" # overwrite an earlier value with a string
> print(exA)
[1] "hello"
```

```java
// in Java, types need to be declared before a variable is bound to value
public class StaticTyping {
 public static void main(String[] args) {
  int exB = 6;   // declare and assign an integer variable a value
  System.out.println(exB);
  exB = "a car"; // when compiled, an error is raised because variable was
   ↪  declared as an integer
 }
}
```

Listing 1: Example of dynamic typing in R and static typing in Java.

an array of web development frameworks.

In comparison to R, over the time the language has also evolved more significantly manifested by addition of new high-profile features. However, being able to afford them, a backwards incompatible version 3.x had to be released in 2008 (Rossum 2009; Lutz 2013). Its objective has been to get back closer to the *Zen of Python* by eliminating design flaws and removing duplicate ways of programming, thus making only "*one obvious way to do it*" (Klein 2018; Peters 2004; Lutz 2013). Although a native 2to3 translation tool has been available to ease the migration pain, for some time the community was split due to many developers resisting to refactor their programs for what many have considered to be a new language (Raschka et al. 2016a; Python Core Team 2018a; Lutz 2013).

General-Purpose Language

A major distinguishing characteristic from R is that Python together with C(++) are consider to be *general-purpose languages* (Cass and Diakopoulos 2017; Brunner and Kim 2016). R targets out of the box only the analytical domain with a substantial focus on statistical analysis. Hence, it does not offer any exemplar toolkits that support web development with Hypertext Markup Language (HTML), Cascading Style Sheets and JavaScript – with an exception of RStudio's *Shiny* framework. Yet its aim is to only help interactive data analysis and it cannot be used to build large-scale web applications that Python enables to create by using *Web Server Gateway Interface* and its compatible packages akin to *Django*.

Besides having capabilities for development of web, gaming applications or system-level scripting, Python community has developed a rich ecosystem for numeric programming as well (Python Core Team 2018b). Therefore, the language has been increasingly used these days for a scientific computing. Particularly, it was adopted for artificial intelligence applications such

as *TensorFlow* as their "first-class citizen" becoming de facto a "go-to" utility for working with multi-dimensional data (Raschka et al. 2016b). Similarly supporting multiple programming paradigms including primarily object-oriented style with some elements of the functional one too, it has found its place in many organizations covering multiple use cases and objectives (Ramalho 2015).

A downside of Python is that its execution speed may not be the fastest when compared to low-level languages like Fortran (Lutz 2013). On the other hand, R and Python allow calling a complied and optimized C(++) and Fortran source code either through native approaches or third-party packages – for instance *Rcpp* for R and *Cython* for Python. As a result, developers can combine advantages of low-level and high-level programming. Whereas the former allows engineers to be close to the hardware speeding up compute-intensive calculations, the latter one permits to experience the ease of programming and use – both of which have contributed to languages' popularity in the DS universe (Perez et al. 2011; VanderPlas 2016). Obstacle

In the same way to R, the Python Software Foundation maintains PyPI[2] which is currently serving over 130 000 packages on its network. Even though it has been further beyond the scope of this work to compare in depth two languages and possibly their ecosystems, a brief overview with essential characteristics is presented in Table 2 and next section.

2.1.3 Summary of Previous Research

2.1.3.1 R and Python Landscape

While some studies from the beginning of section 2.1 are considered pioneering in the field, they have only dealt with SECO's theoretical conceptualization, modelling and categorization (Christensen et al. 2014). To better understand two programming ecosystems, a closer practical perspective was taken through an inspection of several relevant articles, see Figure A.16.

Notwithstanding scholars' best effort in collecting available quantitative data for instance on a number of downloaded CRAN and PyPI packages which would be used to gauge a popularity, health and growth of the ecosystem, *reliable* statistics have been hard to acquire for both platforms. In fact, a lack of historical data from various open source extension markets has contributed to little research analysing other programming languages as well. A thorough search in *Google Scholar* and *Scopus* databases, in addition to inspecting a sample of SECO literature, has revealed that only Ruby with `RubyGems.org` has been extensively looked at in the academia, see Syed and Jansen (2013). At the same time, examinations of software environments such as Java or Node.js have also been published (Constantinou and Mens 2017). Though, they have been largely a domain of blog posts and magazine articles which have used them to answer how Data Sources

[2]https://pypi.org – The Python Package Index Warehouse

Table 2: Presents a high-level overview of two programming languages and how can they be characterized.

Factors	**R (in version 3.4.4)**	**Python (in version 3.6.4)**
Initial public release – Foundation	1993 – R Foundation in Austria since 2002	1991 – Python Software Foundation in the United States of America since 2001
License	GNU General Public License v2.0	Python Software Foundation License (BSD-style)
Language paradigms	Procedural, functional, object-oriented and imperative, see Cady (2017) and Raschka et al. (2016b)	
Object-oriented system	3 natively: S3, S4, RC (Reference Classes) and others via packages such as `R6` and `R.oo`	Only one "default" object-oriented system
Compiled or Interpreted Language	Both are considered to be interpreted languages, executing source code line by line	
Level of Abstraction	High-level languages, with an ability to interface low-level ones, mainly C(++) and Fortran	
Focus/Target Audience	*Domain-specific*: its core focus lies on domains of mathematics, statistics and analysis of data due to being traditionally used in the academia and research	*General-purpose language*: used by a wider audience because of covering a variety programming use cases including web & graphical user interface (GUI) development, system-level scripting or scientific computing
Code blocks – control flow defined by	Curly-brackets { }	Indentation ("off-side rule")
Support for 64-bit integers	Not natively, some limited support through `bit64` package	Native support ("int" datatype)
Support for two-dimensional data structures	Natively `data.frame()` and `matrix()`	Natively only limited `lists = []`, `tuples = ()` and `dictionaries = {}`, thus packages like `numpy` or `pandas` are necessary
Package format and installation	One common format supported natively `install.packages(pkgs = 'car', ...)`	*Egg* used by "easy_install" or newer and natively supported *Wheel* by "pip"
Extension markets (besides *GitHub* and similar)	CRAN and Bioconductor	PyPI, conda-forge
Available implementations (besides the referenced one)	Microsoft R Open, Oracle R Enterprise	PyPy, Jython, IronPython, ActivePython

has a specific language been popular compared to others (Wittern 2016; Poslek 2015; Robinson 2017; Humble 2012).

For analysing R, the situation has been complicated by the fact that only one CRAN mirror, created by a keystone player RStudio, can provide anonymized data about package downloads from October 2012 onwards (Wickham 2013). However, public statistics to more than forty other mirrors are not available due to maintainers' lack of interest, capacity to provide them or simply accessible technology. In comparison, Python community and PyPI have renewed offering download figures only since January 2016 through means of Google's *BiqQuery* warehouse (Stufft 2016). Consequently, both data sources have been used as a proxy to gain an initial insight into R and Python userbase.

Examination of R

While some data have in recent years become ready to use, there is nonetheless a paucity of empirical research analysing software ecosystems and tools within them. R's landscape, and what Asay (2016) has claimed to be a super-linear or an exponential growth of contributed CRAN packages, has been studied by several scientists including German et al. (2013) and Plakidas et al. (2017).

Specifically, the former have focused on understanding "*code characteristics and dependency relationships between the R platform and externally developed*" programs (Plakidas et al. 2016, p. 90). They found that not only there have been differences in the frequency of releases and maturity of different sets of packages but similarly in the quantity and quality of their documentation (German et al. 2013). At the same time, they have observed that all libraries have been well maintained, their size was stable across above-mentioned types and additionally there has been a persisted drift towards having fewer dependencies too.

Likewise, building upon a previous study of Plakidas et al. (2016), a recent quantitative analysis by the same author has examined the "*evolution of the R ecosystem's dependency network and the contribution and collaboration patterns of the developer community*" through CRAN and Bioconductor metrics (Plakidas et al. 2017, p. 3). Moreover, they have investigated environment's health concluding that increasingly, as R's adoption has grown, newcomers have emerged "*display[ing] a 'selfish' behaviour, feeding off the ecosystem without much involvement on their part*" (Plakidas et al. 2017, p. 62). Overall, by comparing two main repositories, authors' analysis has shown "*a broad but relatively shallow network, dominated by a few 'big' actors that supply fundamental extensions, and which once established retain their position*" (Plakidas et al. 2017, p. 62).

Examination of Python

Python's ecosystem has been to date studied only by Hoving et al. (2013, p. 13) who have focused on the ***supply-network level*** attempting to investigate "*which characteristics [could be] identified within*" the FOSS ecosystem when examining Python's universe. Through analysing PyPI, they have described attributes such as SECO's growth and evolution over the time and

compared their results to a similar study done by Kabbedijk and Jansen (2011) about Ruby. Additionally, scholars have discovered significant relationships between different groups of developers ("lone-wolfs" and "one-day flies"), third-party packages (*Eggs*) and communities of users (Hoving et al. 2013).

When looking at other reports and at the same time updating a study by Manikas and Hansen (2013), in a longitudinal work Manikas (2016, p. 2) has extended theoretical and empirical knowledge by "*analyz[ing] 231 papers from 2007 until 2014*" from the field. Going into a greater detail, the author has categorized available literature where various software environments have been discussed concluding that research into SECOs has noticeably matured over the years, "*both in volume and empirical focus*" that has contributed to a deeper understanding of this concept (Manikas 2016, p. 2). Furthermore, it has extended itself to other areas including previously mentioned FOSS too. Nonetheless, even though there has been a considerable amount of studies showing "*signs of maturity*", it has been argued that they have lacked specific "*theories, methods and tools (…) to software ecosystems*" as well (Manikas 2016, p. 28).

2.1.3.2 Surveys of Tools

Importance

Because of later having a purpose to review available R and Python libraries for DS design patterns, another decision was taken to investigate existing studies that have conducted (qualitative) surveys of software applications. The goal was not only to become familiar with a process of gathering and analysing tools but also developing a critical look on what was done in the past and could have been improved when carrying out similar research in this work.

Predominantly, diverse programs have been assessed by academicians in domains of biology and computer science, see Figure A.17. Within the latter group, due to a complexity and size of big data and data mining, a particular focus has been put on providing practitioners an overview of different ML frameworks, storage systems, processing engines or applications for orchestration of server clusters. Notably, *Hadoop* and its related toolbox has been thoroughly studied.

However, long before *Hadoop* was developed, already in 1999 Goebel and Gruenwald have conducted a review of knowledge discovery software. They have identified forty-three software applications and discussed numerous challenges that are necessary to be addressed including having a seamless integration with databases or providing extensibility to offered algorithms.

Similarly, Mikut and Reischl (2011) have surveyed 195 FOSS and commercial programs according to seven criteria and have categorized them into nine types ranging "*from (…) data mining suites, (…) business-centred data warehouses [(DW)]*" and to what they have called ***solutions*** – technology for a specific purpose or a field (Mikut and Reischl 2011, p. 11). Scholars have concluded by suggesting that although there has been "*a wide range of software products*"

covering many different use cases, "*generalized mining [utilities] for multidimensional datasets such as images and videos*" had been missing so far (Mikut and Reischl 2011, p. 11). Although a variety of libraries have been developed since then, not coincidentally, the previous factor was a motivating reason for focusing only on the two-dimensional data format as outlined in section 1.5 too.

Once *Hadoop* has arrived in 2006, an extensive examination by both practitioners as well as scholars has been conducted into its ever-growing family of big data open source software. From recent times, Liu et al. (2014) have investigated real time processing systems and compared their architectures and use cases. On the other hand, Landset et al. (2015) have put their attention on the ecosystem of frameworks for *ML* whereby this term has been associated with applying "*induction algorithms to data*" which allow computers to learn from past patterns and by that improve future predictions (Provost and Fawcett 2013b, p. 357). Consequently, scientists have evaluated four applications according to criteria such *scalability*, *speed* and *coverage* of different classes of algorithms. Before identifying potential areas for improvements, they have "*assigned a rating to each of the four*" utilities in a qualitative manner based on their "*exposure to each [application] and related [literature] works*" – as they have not conducted any experiments (Landset et al. 2015, pp. 27-28).

Even though primarily targeting data analytics and processing of big data with *Hadoop*, naturally surveys of programs have been conducted in other closely-interlinked domains too. For instance, Aceto et al. (2013) have surveyed commercial and FOSS cloud monitoring tools in addition to considering their properties, open issues and challenges. Alternatively, Bonchev and Thomas (2010) have reviewed software for network analysis in molecular biology. Furthermore, visualization programs for life sciences have been studied by Pavlopoulos et al. (2008) and Pabinger et al. (2014) as well.

2.2 Data Science

After introducing the concept of software ecosystems including R, Python and related research, it is necessary to narrow down and put into a relationship two critically important terms – *data science* and *design patterns*. Therefore, to begin this segment, an evolution which has led to DS being a "*hot career choice*" and the "*sexiest job of the 21st century*" is provided first (Davenport and Patil 2012; Provost and Fawcett 2013a, p. 51).

Historical Roots

Discussing data science, a frequently arising topic these days is its origin and how it was born to be a new discipline and a job position in many enterprises. While practitioners as well as scholars such as Provost and Fawcett (2013a) and Carbone et al. (2016) have argued that a newly established term stands on its own to realize its full potential, others like Larson and Chang

(2016) and Vasconcelos and Rocha (2017) have seen DS evolving from business intelligence and being a next step in the data-driven decision making. Not coincidentally is DS considered by some to be under the same umbrella of the BI which dates back to 1958 when Hans Peter Luhn has for the first time introduced a concept of this system. According to Rostcheck (2016a) and Chen et al. (2012), both ultimately refer to gaining actionable knowledge by analysing raw data, though from somewhat different perspectives.

Business Intelligence

Luhn (1958, p. 314) has defined BI as "*an automatic system (…) to disseminate information to the various sections of any industrial, scientific or government organization*". Even though in the past sixty years a precise definition of BI has proven to be elusive as manifested in the study of Al-Eisawi and Lycett (2012), a general understanding of it has been given in the late 1980s (Tutunea 2015). At that time Howard Dressner has attributed it to "*concepts and methods to improve business decision making by using fact-based support*" (Negash and Gray 2008, p. 176). Thereupon, the associated technological means have led to become strategically and tactically important part of reaching informed conclusions across all company levels and industries (Arnott et al. 2017).

Since the end of 1960s, BI has also gone through multiple advancements and saw "*a progression in the development of such information systems*" that support business activities and enable decision-makers to gain a business value from the data (Shollo and Galliers 2016, p. 342). Initially, managers have commonly used management information systems (Watson et al. 2006; Shollo and Galliers 2016; Arnott et al. 2017). These produced "*standardised, (…) period reports that did not allow (…) on-line queries*" and there was no integration between different data sources (Shollo and Galliers 2016, p. 342; Larson and Chang 2016). Although authors have identified and described *generations* differently, two periods of time can be nonetheless clearly observed (Gentile 2015; Shah 2014). Both of them have led to professionals nowadays demanding interactive and personalized reports available on their mobile devices to better understand customer behaviour and make predictions about future sales in the real time (Wilkerson 2016; Evelson 2017; Kimble and Milolidakis 2015).

First and Second Generation During 1980s and 1990s, BI 1.0 was characterized by deploying solutions on-premise, at company's servers, where such tool-centric systems were owned, maintained and further developed by IT departments (Shah 2014). Unfortunately, they were responding to business requirements by providing the knowledge and wisdom about its operation in an inadequate fashion, and thus added little to no agility which company undertakings really needed (Chen et al. 2012).

Single Source of Truth

The back-end and foundation of BI 1.0 system has been laid out in the DW which has finally allowed to store data from heterogeneous and legacy IT sources producing "*a central*

repository with integrated [and cleaned] data for analysis" ("getting data in"), see an architectural sketch of such system in Figure A.18 (Larson and Chang 2016, p. 704; Hejdánek 2016; Watson and Wixom 2007). As a result, it has unified a way of extracting, transforming and presenting data to business users and has established a single source of truth acting as a "*primary [origin] for BI information*" in the company (Heinze 2014; Negash and Gray 2008; Thornthwaite and Ginnebaugh 2012).

Yet BI front-end applications such as portals were complex and inflexible to manage even by IT professionals, nota bene by managers and other end-users (Collier 2011). Moreover, the industry has been dominated by conglomerates akin to SAP or Oracle who brought with them a vendor lock-in due to solutions being proprietary, expensive and demanding to implement, maintain and use for enterprises (Joly 2016). All this culminated into many projects turning out to be unsuccessful, not being used and ultimately abandoned (Collier 2011). At the same time, even if they were implemented, employees could only create limited reports and conduct narrow exploratory data analyses. In essence, they had no capability to make advanced predictions by applying statistical models as these were not widely available and easily exploitable.

When BI 2.0 has arrived in early 2000s, it has started to embrace a new platform, namely the web (Trujillo and Maté 2012). Whereas BI 1.0 has been dominated by analysing structured, internal information stored in relational databases, the advent of web brought semi-structured and unstructured data generated for example on social media (Watson et al. 2006). This gave rise to "*a new class of [non-relational] databases known as NoSQL*" offering new processing, storing and querying paradigms to real time data (Davenport 2013, p. 5; Loukides 2011).

Self-Service BI

In addition, observing a market opportunity, vendors have begun focusing on business employees allowing them to tell engaging stories around user-friendly, interactive dashboards and creating actionable reports coupled for instance with geospatial data ("getting data out"; Fiřtík 2017; Hejdánek 2016; Loukides 2011; Watson and Wixom 2007). By offering self-service BI programs and targeting non-experts who could complete their work without the interference from technical personnel, it has enabled business users to become less dependent on IT units (Heinze 2014). Leading to making the entire process more agile and business-centric instead of IT-centric (Joly 2016).

All this prompted to exploring data in a new fashion, making statistics more attractive and visual for end-users, and therefore providing them better than just simple static reports (Gurjar and Rathore 2013). Similarly, as the volume of collected data from the web continued to rise, analyses have been enhanced by the introduction of *cloud computing* which permitted companies to move from limited on-premise hardware to flexible and instantaneously deployable, on-demand services (Trujillo and Maté 2012). Moreover, studying data has been moving from purely historical to more experimental and increasingly – by using advanced statistical models –

predictive one as well, attempting to answer questions such as "what if" and "why" (Chen et al. 2012; Evelson 2017).

However, even larger potential of *big insights* through the richness of available data was not sufficiently realized and delivered on – resulting into remaining an ongoing struggle gaining actionable knowledge, unresolved with BI 2.0 (Shah 2014). Even though acquisition costs for BI solutions have decreased over time because of the competition from new entrants and availability of cloud solutions, the vast array of gathered information has not yet been fully transformed into bringing a valuable understanding to business operations (Chen et al. 2012).

Seen as a transition, BI 2.0 has stipulated creation of new user-friendly tools while also looking increasingly into the future as opposed to just into the past. Indeed, for some time it was considered as "*the latest in a long line of technologies that have been developed*" for the same purpose of "*creat[ing] knowledge useful for decision-making*" (Shollo and Galliers 2016, pp. 341-343; Evelson 2017). Albeit difficult to pinpoint to a specific year, since approximately 2010 new buzzwords such as big data or ML have emerged and these have further accelerated changes in the BI landscape (Chen et al. 2012; Larson and Chang 2016).

2.2.1 The New Era of Analytics

Genesis

According to Donoho (2017, p. 10), the origins of data science could be traced all the way back to 1962 when a mathematician John Tukey noted "*that something like today's Data Science moment would be coming*". Decades later, after William Cleveland in 2001 and since 2008, D.J. Patil with Jeff Hammerbacher have become credited with describing and giving the term a complete meaning (Jifa and Lingling 2014; Davenport and Patil 2012; Donoho 2017; Zumel and Mount 2014). Principally, they have all argued that the goal of science of data is finding interesting and robust patterns with great predictive powers that could be practically useful by incorporating ML and computational statistics (Dhar 2013; Vasconcelos and Rocha 2017). Hence, finally delivering on *big insights* through utilizing scientific methods of inquiry to understand the vast array of gathered information – "*the realm of data science*" (Provost and Fawcett 2013b, p. 1).

Cao (2016, 2017) has described that DS has evolved from a paradigm shift in statistics which occurred when "elementary" analysis has transformed into advanced data analytics. Building on top of that, data "*analysis has shifted from descriptive (…) to predictive and prescriptive*" one (Larson and Chang 2016, p. 708; Provost and Fawcett 2013a). While it has continued to draw from the traditional concept of DW, it has started to combine data from other external sources – being diverse in quality and type. With a continuing increase in produced and captured information, providing DS in the enterprises is nowadays seen as *the* latest in the development of information systems, technologies, processes and employee's roles that support

company's data-driven decision-making (Rostcheck 2016b; Harper 2014).

As mentioned previously, some authors see DS as a natural evolution of BI which in the corporate environments has long been foundational at supporting mostly explorative and descriptive data analysis and that has become impacted by the emergence of data large in volume (Waller and Fawcett 2013; Larson and Chang 2016). As a matter of fact, Davenport (2013) has called BI "Analytics 1.0" focusing on preparing data for rather short and ad hoc analyses. On the contrary, DS has been viewed as "Analytics 2.0" and a step towards developing data-enriched applications and services, see Table 3 too ("Analytics 3.0"; Larson and Chang 2016; Cao 2017).

Meaning of Data Science

Although attempting to introduce the umbrella term of DS, there has been little consensus about what it actually means – in part because of "*being a relatively young discipline itself, [which includes] facets from multiple other traditional fields of science*" (Carbone et al. 2016, p. 6; O'Neil and Schutt 2013; Shan et al. 2015). Nonetheless, even though diverse definitions have been proposed, authors including Roe (2013) and Corea (2016) have defined it in similar ways. Exemplary, Dichev and Dicheva (2017, p. 2151) has viewed it "*as an interdisciplinary field about scientific processes and systems to extract knowledge or insights from data in various forms*". A comprehensive review of the concept has also been provided in the series of articles by Cao (2017, p. 8, 2016) describing the domain "*that synthesizes and builds on statistics*", computing and communication "*to study data and its environments (…) in order to transform [them] to insights and decisions*". Accordingly, it is viewed as:

> **Data Science**, combining computer science with statistics, ML and domain understanding, is a systematic process of studying data to extract knowledge and actionable insights. This, in order to communicate engaging and enlightening stories to stakeholders and develop data-enriched, value-added, products and services.

Portrayal

Despite the stated definition, it has been challenging to characterise such multifaceted phrase due to for example Agarwal and Dhar (2014, p. 443) arguing that many of its pieces "*have been around for a long time*", in various forms, since at least 1980s (Cao 2016). Additionally and importantly, other similar terms related to data mining or KDD have not been clearly differentiated by the scholars (Ayankoya et al. 2014; Provost and Fawcett 2013a). However, particularly data mining has been commonly assumed to be one subcomponent of a more general KDD process whereby it lies between the data transformation and data interpretation and where specific algorithms are applied for extracting the knowledge (Goebel and Gruenwald 1999). On the other hand, DS acting as a catch-all subject matter incorporates a handful of data-oriented research fields such as big data and ML as well as data-independent ones like stakeholder communication with project management (Cao 2016).

Table 3: Presents a synopsis of a historical and a modern approach to data analysis and how can they be compared to each other. Even though being different in nature, usually elements from both strategies are used at various stages of the knowledge discovery process.

Dimensions	Business Intelligence/Analytics 1.0	Data Science/Analytics 2.0	References
Years	1980s-1990s	Since 2010	Chen et al. (2012) and Davenport (2013)
View of data analysis	Retrospective & Descriptive	Predictive & Prescriptive	Schmarzo (2014)
Perspective	Looks backwards in the history	Looks forwards in the future	Smith (2013b)
Role of BI analysts & Data Scientists	Focuses on exploration of past trends – applied mainly by business users	Uses past data to predict the future – applied by technology-skilled employees with interdisciplinary skills	Saint Joseph's University (2017) and Swanson (2016)
Objectives/ Goals	Assists employees in decision-making by providing access to high-quality, historical data by means of designing DW	Uses hybrid data to develop predictive models that could drive the business by implementing data-driven functionality, for instance recommendations	Larson and Chang (2016) and Rostcheck (2016a)
Typical techniques and outcomes	Executes slice and dice or roll-up/down operations on on-line analytical processing (OLAP) cubes to develop visualizations with key performance indicators (KPI), standard and ad hoc reports	Through experimental & exploratory data mining, creates predictive models and analyses. Develops interactive visualizations, forecasting reports and story-telling presentations	Dietrich (2014)
Targeted questions	What has happened…, how much did… ⇒ already known questions are answered	What will happen if…, why… ⇒ helps to discover and answer new questions	Merritt-Holmes (2016) and Dietrich (2014)

Continued on next page

Table 3 – *Continued from previous page*

Dimensions	Business Intelligence/Analytics 1.0	Data Science/Analytics 2.0	References
Data Sources	Relational databases; flat files; enterprise resource planning, customer relationship management and other (legacy) IT systems – all of which are integrated into the central DW to provide a single source of truth with structured, well-defined information	In addition to the previous ones, uses externally generated data from the web which are often stored in NoSQL and graph databases	Dietrich (2014) and Smith (2013b)
Data Age	Processing and propagation of captured data across the whole BI system (as in Figure A.18) can take several hours due to complex ETL processes – users query historical data (older than 1 day)	Data processing, propagating and querying usually happens instantaneously – users often work with real time data	Larson and Chang (2016)
Data Quality	Detailed, preplanned architecture of DW is aimed to present only complete, cleaned and formatted data of high-quality	Works with data sets which are of diverse quality, hence it relies considerably on probabilities and confidence levels	Merritt-Holmes (2016) and Smith (2011)
Used applications	Most commonly acquires commercial off-the-shelf software, typically for back-end and front-end applications; Uses of SQL for querying relational databases and spreadsheets files	Prevalence of FOSS for (No)SQL databases and ML frameworks, though oftentimes supplemented with commercial add-ons; In addition to SQL, uses other programming languages including R, Python or Java for end-to-end manipulation	Rostcheck (2016a), Smith (2011), and Halper (2016)
Role of Agile analytics and methodology	Adopting agility in a diverse team has not been simple, and thus not widely practised – largely due to a complexity of IT systems and projects	Due to iterative approach, workflow and the speed of having early results, it is naturally predisposition to take advantage of agility with far greater extent	Collier (2011), Larson and Chang (2016), and Krawatzeck et al. (2013)

To help outlining this interdisciplinary conceptualization that applies to a variety of sciences, its three fundamental components are introduced next. At the core, the innovative *technology* has been developed by a heterogeneous *team* of engineers and scientists. Then, by following particular *processes*, the group is led to iteratively establish the "*data-to-knowledge-to-wisdom thinking*" (Cao 2016, p. 72; Corea 2016; Dhar 2013; Provost and Fawcett 2013a).

Big Data With arrival of web, companies have been focused on acquiring competitive advantage in their industries by exploiting the availability of gathered information not only from internal transaction systems but increasingly from new external sources, too, sensors and internet connected devices to name a few (Provost and Fawcett 2013a). To describe a large amount of data collected, a vague term has been coined by the end of 20th century named *big data* which has been associated with three key properties (Carbone et al. 2016; Larson and Chang 2016; Jifa and Lingling 2014; De Mauro et al. 2015). Namely, data have become large in *volume*, large in *variety* (unstructured text and video) and of high *velocity* (speed at which they are captured and need to be acted upon; Halper 2016; Mao et al. 2014; Jagadish et al. 2014). These characteristics are also known as 3Vs with additional dimensions being added which have been most frequently *veracity* (quality and trust) and *value* (potential insights gained; Cao 2016). 3Vs

Principally, the phrase refers to raw information that is too *large* and mainly *complex* to be stored by the traditional data management tools "*within a tolerable time*" – typically where conventional applications known from BI would work well due to keeping more consistent and structured type of data in the DW (Cao 2017; Provost and Fawcett 2013a; Mao et al. 2014, p. 173). Accordingly, to facilitate their transformation, it has been required to develop new analytical technologies such as *Hadoop* and *Spark*, majority of which are open source further opening up the vendor lock-in and lowering the costs of acquisition and maintenance (Provost and Fawcett 2013a; McAfee and Brynjolfsson 2012; De Mauro et al. 2015). Hence, new processing and storage paradigms have allowed to leverage big data with advanced ML algorithms that are applied to extract previously unknown facts. All that by using for instance the power of *cloud computing*, and thus effectively serving audiences with better advertising or supporting manufacturing companies in the supply-chain during their forecasting needs as well (Waller and Fawcett 2013).

The *data deluge*, a situation when complexity and volume of information is overwhelming companies to manage and make use of it, has resulted into ever greater possibilities in understanding and profiling customers in a deeper way as well as tailoring services precisely to their wishes (Cao 2016; Mao et al. 2014). As a result, Davenport (2013) has argued that no longer should big data just help managers in business decisions but instead take a leading role and "*drive the business*" itself – through developing more valuable personalized services and prod-

ucts (Larson and Chang 2016, p. 704).

Yet, despite being often used interchangeably, *data science* is not equal to *big data* (Halper 2016). The distinctive factor lies in the latter one focusing on managing enormous amounts of information with help of for example distributed file systems, while the former one uses it together with ML when it tries to produce actual value from data large and small (McClure 2015). DS ≠ Big Data

Data Scientists With regard to professionals, a novel field has also brought new requirements for a "*next-generation [of] quantitative analysts*" who need to have a broader and a multidisciplinary skillset with analytical mind (Waller and Fawcett 2013; Davenport 2013, p. 5; Provost and Fawcett 2013a). However, even though many engineers are calling themselves *data scientists*, because of a multitude of technologies needed to work with analysing (big) data, enterprises have been faced with a shortage of right human talent who could acquire "*evidence (…) from data by undertaking diagnostic, descriptive, predictive, and prescriptive analytics*" (Cao 2016, p. 73). All this with the aim to provide actionable insights and intelligence for business operations (Dichev and Dicheva 2017; Brunner and Kim 2016).

According to scholars, one of the reasons for such lack has been a necessary skillset where data scientists need to learn diverse technologies ranging from applying *Hadoop*'s ML frameworks in the cloud to being capable of developing predictive models using R and Python ecosystem (Landset et al. 2015; Cao 2017). As illustrated by Venn diagrams of Conway (2013) and Widjaja (2015), these specialists must have programming skills as well as a proficiency in the mathematics, statistics and business analytics (Loukides 2011; Jifa and Lingling 2014; Dhar 2013). At the same time being creative problem solvers, curious and have excellent communication and writing skills (Provost and Fawcett 2013a; Cao 2017). Consequently, by possessing domain understanding and an ability to set up processes for transformation and integration of various sources of information, they shall be qualified for uncovering deep business insights through descriptive, predictive and increasingly the *prescriptive* modelling as well. This is used for telling stories about future courses of action and anticipate and simulate in advance what might happen, and therefore taking proactive responses (Cao 2017). Required Knowledge

Jurney (2013, p. 6) has described that these experts have a role to "*explore and transform data in novel ways (…) and combine [them] from diverse sources to create new value*". Ultimately, data scientists look for unknown, ask improbable questions and make explorations into diverse data sets. Among others, they make visualizations to expose "*early and often*" hidden facts and patterns around which business stories can be told and issues solved (Jurney 2013, p. 6). Roles and Responsibilities

It has been further noted that data scientists have diverse responsibilities during the KDD life-cycle (Cao 2017). Once understanding the objectives, they need to formulate research ques-

tions, transfer business problems into analytical tasks and subsequently understand data they intent to work with. When this has been collected and potentially enriched with other sources, they build complex pipelines which prepare data for applying ML algorithms to create powerful models aiming to turn raw data into information, later into insight and at last in "*business decision-making actions*" (Goebel and Gruenwald 1999; Cao 2017, p. 30). Furthermore, the role involves typical project management duties, including careful project planning, resource allocation, reviewing requested changes, mitigating risks and last but not least ensuring a proper project closure. Thus, by leveraging the predictive modelling and big data, data scientists have become a centrepiece to many, these days, data-driven businesses – be it in the finance or marketing. In fact, through the results of data analytics, they are competent to transfer them "*into benefits for society*" at large and companies alike (Dhar 2013; Carbone et al. 2016, p. 6).

Iterative, team-based development Larson and Chang (2016, p. 705) have further reported that DS "*involves iterative [and incremental] development of analytical models where [they] are created, validated, and altered until the desired results are achieved*". Hence, the importance of ML that gives computers the ability to learn by themselves without being preprogrammed doing so (Samuel 1959). Because of this interactivity during which (big) data are transformed and prototypes are developed, tested and adjusted, any novel discoveries may lead into changing earlier results, thus being in constant feedback loops (Goebel and Gruenwald 1999).

Nonetheless, even though desired, one single person often does not possess necessary skills to cover all areas of DS which span analysis, development and deployment of solutions (Carbone et al. 2016). On these grounds, a continues collaboration and communication within usually a smaller analytical team of data engineers, product managers, UI/UX designers and researchers is necessary for successful value-based driven knowledge extraction in enterprises (Jurney 2013; Domino 2017; Shan et al. 2015).

Agility

According to the literature, when compared to the traditional BI that has usually applied a waterfall method for development, DS has also been better predisposition to benefit from using agile practises due to inherently practising them with far greater frequency and success (Collier 2011; Krawatzeck et al. 2013). As described by Larson and Chang (2016), the nature of DS encourages persistent interaction with stakeholders to confirm the direction and quickly respond to changes and results rather than following a strict plan. Moreover, the aim is to work on smaller and shorter releases while at the same time have a valuable and usable documentation – precisely where design patterns can have a noticeable impact (Larson and Chang 2016).

CRISP-DM

While the technological landscape has advanced profoundly with arrival of big data, the methodological approach to *data discovery*, which starts as soon as raw data have been acquired, has not changed significantly over the past decades (Larson and Chang 2016; Landset et al.

2015). Indeed, a host of models has been developed and one of these processes "*with reasonably well-defined stages*" for acquiring knowledge and delivering actionable business information has been Cross Industry Standard Process for Data Mining (CRISP-DM; Provost and Fawcett 2013a, p. 56). Claiming to be one of the most frequently applied and adjusted frameworks, it has been published in 1999 to service the needs of data mining community by helping them to solve common pitfalls in data-driven projects in terms of understanding, preparing and analysing their information (Chapman et al. 2000; Zeutschler 2016; Fiřtík 2017; Grolemund and Wickham 2018).

As shown in Figure A.19 and seen next, the cyclical and iterative KDD process model consists of six phases which according to researchers such as Jurney (2013) should also fit all within one to four weeks of development, named *sprints*:

(a) *business understanding* (assessing situation and determining objectives),
(b) *data understanding* (collecting and exploring data),
(c) *data preparation* (data cleaning and formatting presenting the largest effort overall),
(d) *modelling* (selecting, designing and building statistical models),
(e) *evaluation* (of results and reviewing the process) and finally
(f) *deployment* (of solutions, making use of outcomes and finalizing the project).

Each of these stages has other *general* and *specialized* tasks and by following these steps, the framework providing "*the overview of data mining life-cycle*" should help businesses to reduce operational costs, increase time-to-market and support knowledge transfer within organizations through appropriate, domain-specific management of analytical projects (Zeutschler 2016, p. 16; Horvath 2011). Ultimately, providing a tool-, industry- and technology-neutral model in the business context (Chapman et al. 2000).

2.2.2 Design Patterns

Attempting to put into a relationship DS and a concept of *design patterns*, the last key term is introduced. As briefly outlined in the introduction, in 1977, a civil engineer named Christopher Alexander has published a literature work, nowadays a perennial seller, entitled *A Pattern Language* which gave a birth to a pattern movement (Salingaros 2018; Pan 1998). Being grounded in the domain of architecture, the profound work of Alexander et al. (1977) has defined each pattern portraying a "*problem which occurs over and over again (…), and then describes the core of the solution to that problem, in such a way that you can use this solution a million times over, without ever doing it the same way twice*" (Spinellis 2001, p. 93).

Patterns described in prose are useful means "*of capturing time-tested[, optimized and generalized] design solutions and facilitating their reuse*" – thus helping "*people reason about what they do and why*" through provided terminology (Heer and Agrawala 2006, p. 853; Schmidt et al.

1996, p. 37; Douglass 2002). In the words of Pan (1998, p. 13), they establish the ability to discuss and "*record design tradeoffs*".

Unfortunately, a single pattern may be insufficient to describe a wide-ranging challenge and there might be even alternative solutions too. Therefore, Schmidt et al. (1996, p. 37) has stated that "*when related [context-dependant] patterns are woven together they form*" a system of ***standard vocabulary*** and effective pieces of advice to "*recurring problems*" in various fields (Fowler 2002, p. 10; Inventado and Scupelli 2016, 2015; Geist et al. 2012; Wilson 2008). As a result, a ***pattern language*** such as the one for planning and construction purposes or in the **chapter IV** represents a network of relationships to resolve the complexity in specific situations (Heer and Agrawala 2006; Dearden and Finlay 2006).

Value

Correspondingly, the advantages of using systematic approaches to commonly occurring problems for instance in phases of software analysis, design and implementation are several fold. First and foremost, according to Spinellis (2001, p. 93), design patterns "*offer a convenient way to capture, document, organise, and disseminate existing knowledge from a given area in a consistent and accessible format*". Furthermore, the use of patterns in a conscious way can improve the overall understanding of sophisticated concepts and provide a better transfer of experience within a company as part of keeping and enhancing its organisational memory (Giridhar 2016; Schmidt et al. 1996; Dearden and Finlay 2006). The existing body of research has also suggested that they "*lower total cost of ownership*" due to fostering communication of ideas, coordination and collaboration of work between stakeholders (Leal and Boldt 2016, p. 1; Fowler 2002). Principally, however, patterns are responsible for making better design decision through using "*simpler, [scalable,] more flexible, modular, reusable, and [generally] understandable*" techniques, best practises and proven solutions (Chen 2004, p. 1).

Drawbacks

Nonetheless, while having benefits of improving project documentation and being a *lingua franca* in communication with domain experts, at the same time patterns might prolong time required for development of applications as well (Hakeem 2017). Moreover, if used improperly, they can decrease source code understandability due to a larger complexity of the implementation (Dearden and Finlay 2006). Besides, Hakeem (2017, p. 2) has listed that they could limit design options and "*leave some important details unresolved*", thus acting more like templates that demand further refinement and which cannot be blindly used (Fowler 2002). For these reasons, as noted by Schmidt et al. (1996, p. 37), "*all solutions have costs, and pattern descriptions should state the[se] clearly*". From the supplementary perspective of their characterization, Norvig (1996, p. 4) has likewise reported that because they are invisible, they are difficult to notice and formalize as it is a human who shapes them to "*explicitly appeal to aesthetics and utility*". On the other hand, once established they can become an invaluable resource for stakeholders' daily tasks.

Pattern Structure

Both Douglass (2002) and Pan (1998) have remarked that writing a pattern consists of three core components. Namely establishing a relationship between a *problem* it is supposed to address, a *solution* which is the pattern itself describing various "*elements that make up the design*" and a specific domain context in which it would be applied (Gamma et al. 1994, p. 13; Meszaros and Doble 1997). In addition, researchers have stated that patterns should have an evocative *name* forming a common vocabulary for effectively sharing experiences and knowledge, a *diagram* or *sketch* for instance in the Unified Modeling Language (UML) and a *summary* describing its purpose (Fowler 2002; Heer and Agrawala 2006; Norvig 1996). Last but not least, there are *forces*, *implementation details*, *consequences* of its use and *examples* demonstrating the application (Chen 2004; Nadschlager 2017; Guerrero and Fuller 2001; Meszaros and Doble 1997).

Application

Not being invented but rather discovered "*by trial and error and by observation*", design patterns were applied in various domains from education, e-learning and communication to over being applied for game design, sociology or in the business (Kolfschoten et al. 2010; Bergin et al. 2012; Dearden and Finlay 2006, p. 63). Focusing in this work on their use in the IT, numerous documents have already been published summarizing widely used approaches in many computer science disciplines. Case in point, books by Hansen (1995) and Mattson et al. (2004) have identified patterns for concurrent, parallel and distributed programming. Works of Erl (2015), Fehling et al. (2014), and Blaha (2010) have dedicated their attention to cloud computing and database modelling. Fowler (2002) with Hohpe and Woolf (2003) have presented related best practises when integrating and developing enterprise, service-oriented middleware applications where such patterns have an impact on the whole system due to their extensive scope (Douglass 2002).

GoF

However, perhaps the most important pattern-related publication has been released in the software engineering by Gang of Four authors in 1994 who popularised twenty-three patterns for object-oriented software design, categorizing them into *structural* (examining relationships), *behavioural* (dealing with object interaction) and *creational* ones (handling object formation; Pan 1998). Having a clear proposition and being local in scope, practitioners have explained in the "*collection of relatively independent solutions*" that their patterns describe a communication and collaboration of objects and classes "*that are customized to solve [issues] within a particular [programming language independent] context*" (Heer and Agrawala 2006, p. 853; Schmidt et al. 1996, p. 39).

It has been only since Gamma's et. al (1994) fundamental text that the study of design patterns has gained a momentum in the research and has extended itself into other areas of human-computer interaction including UI/UX design (Pan 1998; Dearden and Finlay 2006). This effort has been also spearheaded by *The Hillside Group* through which participating schol-

ars have developed an extensive body of knowledge that is being shared on *Pattern Languages of Programs (PLoP)* conferences. Indeed, given their importance for the community, both information sources have been an instrumental gateway in this understanding of design pattern research.

Bridge Pattern

When looking at a typical example described by GoF, a structural *bridge pattern* has been illustratively implemented in the Java Database Connectivity application programming interface (API). Its intent has been to solve a problem of "*decoupling abstraction from implementation so that the two can vary [easily and] independently*", essentially separating two concerns, and therefore improving their extensibility (Gamma et al. 1994, p. 171). Hence, the goal of such pattern is assuring that there is a class separation where the developer does not call a platform- or vendor-specific implementation but instead uses its abstraction, a general and independent interface. Besides, GUI frameworks like *Qt* have adopted it to provide a set of consistent interface elements that could be used across all supported hardware platforms while at the same time considering various specificities of operating systems.

Context

Turning now the attention to design patterns used for DS and encompassing domains of BI, big data and ML, these shall be capable of capturing "big ideas" and best practises across different levels of abstraction – providing both theoretical and technical solutions (Mosaic 2014; Giridhar 2016; Dearden and Finlay 2006; Delibašić et al. 2008). As such, the discovered *data science design patterns* should offer effective and efficient means to problems found in the DS lifecycle. This, when following for instance CRISP-DM and its stages of preparing, modelling and evaluating information which might be large in volume, variety, velocity and is often of a great uncertainty (Hakeem 2017). The objective of these *higher-order abstractions* is to symbolize the knowledge that is implicitly understood by the experts in order to share the wisdom with other KDD professionals and stakeholders in the organization (Heer and Agrawala 2006). Thus, precisely describe a solution to an issue "*in a given context*", see next (Spinellis 2001, p. 93).

> **Data Science Design Patterns** represent reusable design solutions that solve a frequently occurring problem in the interdisciplinary field which studies data by means of applying scientific methods to extract valuable and actionable knowledge and insights.

2.2.2.1 Related Research

SLRs

Moving on now to consider related research in the field of design patterns, several mapping studies have been conducted and one of possibly the most comprehensive ones has analysed 637 articles published between 1995 and 2015. Bafandeh Mayvan et al. (2017) have broadly classified the research into six pattern categories dealing with:

(a) *development* (also the most frequent objective),
(b) *usage* (pattern utilization and application),
(c) *mining* (techniques for their detection),
(d) *quality evaluation* (assessing their impact),
(e) *specification* (methods and notation for their description) and
(f) *miscellaneous issues* (for example their use in refactoring).

Earlier, Ampatzoglou et al. (2013, p. 14) have arrived to a similar taxonomy when they have specifically targeted GoF-related "*studies that deal[t] with the effect of pattern application on quality*" features. Consequently, they have concluded that patterns "*enhance one quality attribute [for instance the functionality and usability] in the expense of another*" one – hence they "*cannot be characterized as universally 'good' or 'bad'*" (Ampatzoglou et al. 2013, p. 14).

After having discussed two general studies that were conducted to analyse the field of pattern research, an extensive search in scholarly databases such as Scopus and Google (Scholar) has revealed that *data science(-oriented) design patterns* have so far received little attention as suggested by the paucity of studies that have been dedicated to them. According to Cao (2017, p. 2), one possible explanation for this lack could be attributed to DS being a broad concept and "*at a very early [exploration] stage*". Though, on the other hand, once being further narrowed down to its individual sub-fields like big data processing or prediction utilizing ML methods, a handful of relevant studies as well internet resources have been identified and catalogued in Figure A.20 too. Therefore, following paragraphs consider numerous subject areas, one of which is the information visualization – the outcome of data analysis and arguably the primary focus of DS by which stories can be told and action can be taken.

Visualization

Indeed, this has already been extensively studied by scholars who have documented various approaches to "*model[ing], design[ing] and perform[ing]*" decorative tasks on data, see Chen (2004, p. 5), Ware et al. (2013), and Heer and Agrawala (2006). A classical literature work by Tutfe (1983) has laid out foundational grounds and inspired both designers and programmers in incorporating visual thinking processes into common design practices. Therefore, maximizing information value for the end-user while at the same time minimizing a development effort for software engineers or UI/UX creators (Ware et al. 2013). Specifically, Chen (2004) has presented nine high-level solutions for interactive visualization which have been categorized into *data*, *structural* and *behavioural* patterns illustrated on examples of *visual encoding*, *graphic grid* and *brushing*. On the other hand, at a lower-level Heer and Agrawala (2006) have proposed a pattern language consisting of twelve problem and solution pairs including *scheduler* and *camera* which have been observed in various GUI frameworks and applications.

MapReduce

Miner and Shook (2012) have classified over twenty design patterns into six overarching themes dedicated to *Hadoop* and *MapReduce* paradigm. Moreover, they have stated, not sur-

prisingly and in line with findings of this study too, that research has heretofore been scarce and scattered across diverse, often unreliable blogs, websites or hidden in books and articles where a small chapter is devoted to recurring issues and their solutions in data analytics.

Big Data

While Gamma et al. (1994) have been dealing with classes and objects, from an architectural point of view, patterns discovered in big data have been touching multiple different components (Fabian 2013; Fowler 2002). Hence, trying to understand how complementary enterprise systems have been designed, Hopkins et al. (2013) of Forrester Research have interviewed professionals in eleven companies and have uncovered four architectural concepts. Ultimately arguing that *hub-and-spoke approach* has been providing best capabilities and extensibility through having a common low-cost storage layer to which various DW, BI tools or mathematical packages could be connected and later used by company's departments for their specific needs.

The on-line catalogue at `BigDataPatterns.org` has given further insights into necessary IT artefacts describing diverse processing and query engines and other applications that have been found within the current analytical systems in organizations (Arcitura 2018; Erl et al. 2016). Once *technological mechanisms* have been presented, the same authors have introduced thirty-five big data design patterns which have been later grouped into a new perspective of compound patterns. These combine a set of stand-alone ones to create models or technology-sets – for example *analytical sandbox* or *big data processing environment*.

Layered Architecture

Being known from the enterprise IT architecture, *layered design* pattern has suggested to separate systems into logical, discrete "*groupings of the functionality*", irrespective of hardware's "*physical location*" (Microsoft 2009). Analogous to authors such as Hakeem (2017) and Landset et al. (2015), Mysore et al. (2013) have described four building blocks of a solution platform for insights discovery, see Table 4. Going more in depth, it has been suggested for company employees "*to think in terms of big data requirements and scope*" instead of layers (Mysore et al. 2013). Thus, *atomic*, *composite* and *solution patterns* have been proposed.

Addressing specific requirements, the first group has been further divided into patterns for *data consumption*, *processing*, *storage* and *access*. On the other hand, the composite ones such as *store and explore* encapsulate several dimensions to solve "*a given business problem*" and together with atomic ones contribute to establishing the final category dealing with business scenarios, for instance *take the next best action* (Mysore et al. 2013). Overall, layers are important because it is them which programmers and researchers look at when mapping existing software tools and identifying gaps that need to be addressed by developing new technology (Fowler 2002; Mysore et al. 2013). As a result, they should enable better understanding of what (non-)functional requirements and tasks are necessary to perform in these contexts too.

Deep Learning

The yet-to-be-published book by Perez (2018) entitled *Design Patterns for Deep Learning*

Table 4: Presents logical layers as they have been described by Mysore et al. (2013).

Tier	Meaning/Examples
Big Data sources	Format and point of data location and collection
Data massaging and store layer	Distributed file systems, relational databases
Analysis	Recommendation engine
Consumption	Presentation/visualization capabilities

Architectures concerns a biologically inspired concept within the artificial intelligence called deep learning (Arel et al. 2009). In author's ongoing research, patterns have been categorized into seven thematic areas ranging from the ***representation*** ("*how neural networks represent data*") and ***explanation*** dealing with "*different kinds of output from neural networks*" to ***serving*** (deployment of models; Perez 2018).

Besides another forthcoming book by Morley (2019) mentioned already in the introduction, up to now very little works have included phrases of DS and design patterns together. Illustratively, Mosaic (2014) has defined this compound concept as "*reusable computational pattern[s] applicable to a set of data science problems having a common structure, and representing a best practice for handling such problems*". In their five technical blog posts, supplemented occasionally with R code examples, authors have looked at approaches that transform and combine individual variables or handle the missingness in data.

Data Science

Last but not least, Fabian (2013) has written a chapter in his work on data-oriented software design where he listed, among others, eight specific patterns like *in place transformation* which modifies data (structures) within the same container or *tasker* that concurrently runs transformation tasks on many data pieces without considering other parts. Being foundational, both have been implemented in many programming languages including R and Python and within their ecosystems as their primary object of concern are raw data.

Data-Oriented Patterns

2.3 Summary

To summarize, by pursuing a first study question, the goal of **this chapter** has been to provide a background to important concepts in this thesis.

It has begun with section 2.1 which has not only allowed to gain a better understanding of two programming languages but also familiarize the audience with previous research in the field. Consequently, its importance lies in deepening and broadening the knowledge that is further needed for answering the second and third study question.

SECO and R & Python

Albeit having strengths in different areas, one of the main reasons why two formal systems are applied in the DS community is their ecosystem offering a wide range of cutting edge

algorithms that are indispensable in order to effectively mine data for obtaining a wisdom. As touched upon in the introduction, R and Python have been often compared to each other trying to identify which is a preferred choice to accomplish all developer's demands and use cases. Nonetheless, it is usually required to use a multitude of tools, and therefore Theuwissen (2016) has highlighted the need to learn one programming language after another as they also share parallels in terms of orientation and functionality.

While both have been actively evolved by their core team of contributors, Python has continued to attract larger attention due to being generally purposeful and having more visible changes between the releases (Cass and Diakopoulos 2017). On the contrary, R has aimed for a backwards compatibility all the way back to its roots, and therefore has followed a more conservative approach whereby relying on community efforts and particularly CRAN more significantly.

Additionally, when looking to gain a deeper understanding of programming ecosystems and especially those of R and Python, section 2.1.3.1 has uncovered that research has been done largely on the *software ecosystem* and *supply network level*. In fact, taking a holistic look at their tools, conducting analyses of used packages and principally studying the *vendor (developer) level* has been avoided. Unsurprisingly, due to a lack of necessary data and complexity of their acquisition, existing studies have primarily analysed R. Moreover, only the work of Constantinou and Mens (2017) has attempted to examine ecosystems in combination and no other studies were found mapping available tools for DS purposes or interconnecting them with *design patterns* as well. This is where this thesis aims to intersect between subject matters when aggregating rather specific R and Python application universes with a more generic conceptualization of DS design patterns.

When taking a deeper look at various surveys of utilities published to date, their review in section 2.1.3.2 has demonstrated shortcomings when observing that only higher-level inquiries of diverse types of programs were made. Indeed, a strong focus has been put on *Hadoop* family, meanwhile, R's and Python's specific features and KDD ecosystem capabilities have not yet been thoroughly investigated. Furthermore, scholars have usually inadequately explained how their surveys have been methodologically conducted in terms of tools' discovery and selection – other than according to their defined objectives. This lack of details has been pointed out by Jansen (2010) and as a result it is aimed to sufficiently explain how a concurrent collection of applications is done in section 3.1.1.2.

With an exception of design patterns, after investigating SECO and DS and attempting to understand their exact meaning, an unfortunate fact could be highlighted that none can be defined with one simple and common definition covering all aspects and views. This is due to terms' ambiguity, often labelled as "umbrella" phrases, and thus creating a number of rationales

and interpretations by researchers. To have an overview and overcome this challenge, an attempt was made to relevantly define each concept taking into account different understandings that have been proposed in the literature.

Data Science

As seen throughout the segment 2.2, DS is difficult to pinpoint exactly because it is interconnected with other related fields such as BI and statistics from which it was born and evolved into nowadays being indispensable for companies to preserve their competitive advantage in the business (Provost and Fawcett 2013a). Accordingly, it was defined as an overarching concept that inherently encompasses technology for big data and ML techniques as well. Similarly, Ayankoya et al. (2014) have refereed to DS as a convergence of three themes, namely BI, advanced analytics which is typically manifested in the form of domain-specific ML capabilities and big data. In this work, once three typical characteristics were described – the use of big data by a new class of professionals who create statistical models utilizing Agile principles and following frameworks like CRISP-DM – it has additionally allowed to illustrate and compare what Davenport (2013) has called Analytics 1.0 with Analytics 2.0 in Table 3.

Design Patterns

Subsequently, design patterns were finally introduced and being at the core focus of this work it was attempted to put them into a relationship with aforementioned process of extracting insights by applying scientific procedures. Taken together, the investigation has established that to date little research was found in the academic literature with regard to design patterns and their application in the DS. Although some works and internet resources have touched upon related patterns in one way or the other, they were narrow in focus on one particular stage of data analysis, paradigm or for example attempted to provide a perspective by way of layers and components of big data systems.

Continuing the quest of addressing the research gap, the **next chapter** details the methodology of how patterns are discovered.

CHAPTER III

Research Approach

Thus far, the thesis has provided a fundamental understanding of key terms. Hence, this chapter continues by outlining a research design for a comprehensive synthesis of design patterns in the DS context and applications from R and Python software ecosystem.

3.1 Methodology

To create "*a good research strategy*", it is first necessary to talk about two principal schools of thoughts that have underpinned study designs (Klakegg 2015; Creswell 2013). Historically, academicians across disciplines have argued which research approach, qualitative or quantitative, is superior to the other one (Small 2011). These stem from different philosophical views on science of knowledge, particularly understandings of ontology dealing with researcher's position on a nature of reality and epistemology which investigates from where and how is information captured (Carson et al. 2001; Klakegg 2015). QUAN ∨ QUAL

Simplified, on the one hand, the *positivist* ontology attempts to analyse outside world through formalized mathematical techniques, and therefore staying free of emotions and feelings, resulting into using "*rational and logical approaches to research*" (Edirisingha 2012). The quantitative methods of inquiry have been utilized to "*discover objective truth*" within a single reality independent from humans (Small 2011; Carson et al. 2001). On the other hand, ***interpretivism*** believing that knowledge is socially constructed "*adopt[s] a more personal and flexible*" research structures to accommodate human interaction and stay "*open to new knowledge throughout the study*" (Edirisingha 2012). Consequently, often "*considered 'softer' or 'fuzzier'* ", qualitative methods have been applied for interpreting multiple realities, "*subjective experience[s]*" and gaining a contextual understanding (Seaman 1999, p. 558; Small 2011; Carson et al. 2001).

Due to being confused and misused for stating that one scientific paradigm is "better" than the other one, it has been noted that qualitative methods refer to "*any kind of research that*

produces findings not arrived at by means of statistical procedures or other means of quantification" – where collected data are typically expressed through words or pictures (Corbin and Strauss 1990, p. 17; Symonds and Gorard 2010; Carson et al. 2001). On the contrary, the quantitative data are "*represented as numbers or other discrete categories*" (Seaman 1999, p. 563).

Mixed Methods

Attempting to address a long-standing debate between qualitative and quantitative proponents, scholars from psychology and sociology have as early as 1960s started to employ multiple methods in a single study with a motivation "*that confidence in one's findings increases when different methods are in agreement*" (Small 2011, p. 61; Saunders et al. 2015). Hence creating a ***mixed methods methodology*** which has been underpinned by a ***pragmatic*** research philosophy, a third approach to science. The stand-alone research technique attempts to meaningfully integrate qualitative and quantitative data and their analyses to "*gain a more complete understanding of*" questions and hypotheses – exploiting "*the strengths of both*" styles (Guetterman et al. 2015, p. 554; Small 2011; Cameron 2009).

Even though being capable of triangulating methods and data, taking into account multiple perspectives when building upon previous findings and providing better interferences, a clear drawback is a required time to implement this design and its resulting complexity for the audience (Wall 2018; Creswell 2013). Additionally, mixed methods need to fit research goals and scholars have to be skilled in using and incorporating both methodologies while at the same time managing any inconsistencies when collecting and analysing data (Malina et al. 2011).

Interpretivist Ontology

When considering all three major approaches, due to a nature of the subject matter a decision was taken to carry out only a qualitative type of research. Thus, DS design patterns were discovered based on observations of collected literature and other sources (Nadschlager 2017). The qualitative way was here more suitable because it was not an objective to numerically develop such understanding from empirical data via structured interviews or quantitative surveys (Schmidt et al. 1996). Moreover, the interpretative methods were more appropriate as design patterns are socially constructed by humans with their subjective views and understandings. For that reason, arguably, no single objective and "correct" truth could be discovered.

3.1.1 Research Design

Given that a clarification of overarching intentions was already made in **previous chapters**, a thesis protocol in Figure A.21 was developed to serve as a gradual roadmap, demonstrating a dependable and coherent strategy and making the process clear throughout the whole reporting.

Okoli and Schabram (2011) have presented eight rigorously defined steps that are required to adhere when conducting SLR in order to provide a "*background for subsequent research*" in the field. Even though this work did not administer SLR, some of its principles were borrowed as well, for example transparently presenting the methodology (Hajimia 2014; Coyne 1997).

Therefore, aiming to be "*systematic in [accordance with] a methodological approach, explicit in explaining the procedures by which [the study] was conducted (…) [and] comprehensive in (…) including all relevant material*" (Okoli and Schabram 2011, p. 1).

To uncover design patterns, numerous techniques were already described in PLoP journal articles and conference proceedings (Inventado and Scupelli 2015). To illustrate, the *introspective approach* has been used by Miner and Shook (2012) who have relied on their own vast experiences and knowledge to identify and share domain-specific design patterns. Alternatively, this has been further combined with *pattern mining workshops* that included brainstorming sessions. Others have created algorithms for "*identif[ing] behavioural and structural patterns through static and dynamic analysis*" of the source code (Ampatzoglou et al. 2013).

3D2P

This examination has followed *data-driven design pattern production methodology* of Inventado and Scupelli (2015), see the adaptation of pattern discovery for this study in Figure 3. With a goal to address the second research question and formalize DS design pattern candidates, the inquiry began with a predominant collection of qualitative data to later meaningfully integrate their interpretation and further observations through a general inductive approach (Inventado and Scupelli 2015; Thomas 2006).

3.1.1.1 Pattern Prospecting

The aim of *pattern prospecting* stage is to gather available information that could lead to discovering DS design patterns (Inventado and Scupelli 2015). Therefore, in line with Nadschlager (2017), this phase commenced with collecting a sample of relevant literature ("the data") in order to reveal various dominant relationships and significant themes as they have been reported as well as being implicitly utilized.

Searching Literature

To identify a set of pattern candidates that after being verified and evaluated might qualify to become practically used *design patterns*, author's background knowledge from DS field has guided the collection for both primary (original documents) and secondary literature resources (their interpretations; Inventado and Scupelli 2015). Correspondingly, a continues search was conducted in relation to frequently occurring issues and problems that data scientists often face and where solutions could be provided using tools from R and Python ecosystem. At first, it was looked for journal articles and conference proceedings in the Scopus and *Web of Science* databases taking into consideration KDD stages. Albeit not being fully reproducible, search queries combined keywords noted in Table B.6 with an objective to gather diverse set of information spanning different (sub-)fields of application. Additionally, due to a recency and nature of the subject matter, the collection was enhanced by searching Google (Scholar) for grey literature including working papers or reports from even unreliable internet sources.

Practical Screen

Before recording them in the database, see Figure A.22, documents were skimmed for their

content and abstracts (if any) were read to decide for the inclusion and exclusion in this study. This resulted into twenty-six initial pieces of text – some of which were also selected from **chapter II** because of their promising relevancy for this study. To capture distinct data, while no restrictions were made to any specific scientific area (other than stated goals in relation to DS field), time-frame or research design, only English documents were considered. At this stage, the most significant criterion was the potential relevancy and soundness of information. If the identified resource was not deemed applicable to DS, it was excluded due to research intentions (Mao et al. 2014).

Next, when scanning the literature work itself, references were inspected permitting a forward search by using aforementioned databases. Thus, the compilation was enriched using the "*most widely employed method of sampling in qualitative research*", the ***snowballing*** (Noy 2008, p. 330; Webster and Watson 2002). Indeed, forty new samples were further collected which included books dealing with R and Python and their use for analytical purposes, DS-oriented video presentations or seemingly relevant blog posts – all of which are further referred to as *literature works*. As a result of the conducted search, sixty-six information sources were gathered being empirical as well as theoretical, talking for instance about application of best practises in data mining, see Figure 2 too.

Data Extraction

Once all data have been inspected, it was necessary to systematically extract the relevant information "*to serve as the raw material for the synthesis stage*" (Okoli and Schabram 2011, p. 29; Inventado and Scupelli 2015). Hence, commonly known techniques from the information systems research, namely the GIA and ***grounded theory methodology***, were considered to modify the data representation "*for data analysis purposes*" (Matavire and Brown 2011, p. 119). While the goal of the latter one is to "*build theory relevant to the discipline*" which is grounded in the data, a more generic GIA was selected because it attempts to identify most influential themes "*to research objectives*" that are evident in the text (Thomas 2006, p. 241; Matavire and Brown 2011, p. 119). Being similar, both strategies are undergird by an inductive procedure of creating ***codes*** where design patterns are essentially drown "*from a number of [literature] observations*" through multiple readings and interpretation of data (Klakegg 2015; Thomas 2006; Matavire and Brown 2011, p. 119).

Inductive Coding

Consequently, during close and iterative examination of sixty-six sources, parts of text or phrases that describe DS reoccurring problems, tasks, applied methods, effective solutions with best practises or lessons learned were labelled using a "*systematic procedure for analysing qualitative data*" – the ***open coding*** (Thomas 2006, p. 238; Inventado and Scupelli 2015). The intention was to develop a scheme consisting of high-level and low-level categories and themes which were derived from the most frequent keywords and interpretation of "*specific text segments*", see its outcome in Figure A.22 (Thomas 2006, p. 241; Jansen 2010; Okoli and Schabram 2011;

Bafandeh Mayvan et al. 2017). Therefore, labels were created for key meanings and associations found in the specific literature work according to the best efforts and being "*shaped by the assumptions and experiences*" of the evaluator (Thomas 2006, p. 240). At this stage no further links or relationships were established as this was planned to be realized next.

Unfortunately, a substantial amount of information originated from non-academic and unreliable internet sources where related best practises and solutions to frequently arising obstacles are shared for example in discussion forums. Not surprisingly, during examination of all sources, twenty-five pieces were further excluded to increase the quality of a sample due to considering them too general in terms of relevancy or otherwise unfit for study objectives. Quality Appraisal

3.1.1.2 Pattern Mining

According to Inventado and Scupelli (2015, p. 5), the objective of *pattern mining* stage "*aims to discover or explain patterns*" that are gained through insights and uncovering interesting relationships in the data – the coded documents. As a result of previous pattern prospecting phase, each resource was inspected and initial set of dominant topics were established in what Webster and Watson (2002) have called a *concept matrix*. Synthesis

Having read and labelled all information sources, it was necessary to begin with structuring and continuous revision of labels as well as stepwise "*reduc[ing their] overlap and redundancy among [the identified] categories*" by determining relationships (Thomas 2006, p. 242; Okoli and Schabram 2011). Indeed, the inductive coding – as illustratively explained by Seaman (1999, p. 563) and Seaman (2013) in the context of empirical studies in the software engineering – helped "*transforming qualitative data into quantitative*". Subsequently, the overarching high-level and low-level properties were organized to establish hierarchies that translated into plausible ideas for DS design patterns (Thomas 2006; Matavire and Brown 2011). Therefore, twenty-one *axial codes* across dimensions were now created "*presenting the key concepts*" which linked multiple important pieces of text together utilizing "*a logical approach to [their] grouping*" (Webster and Watson 2002, p. 17). Through such iterative process of interpretation and evaluation, findings "*emerge[d] from the frequent, dominant, or significant*" thoughts which were presented in a brief and compact format (Thomas 2006, p. 238). Ultimately "*captur[ing] the [essential] aspects*" in a vast, complex data by way of a smaller number of labels that suggested "*effective solutions*" or identified "*recurring problems*" (Thomas 2006, p. 242; Inventado and Scupelli 2015, p. 5). Reducing Overlap

Webster and Watson (2002) have also described Figure A.22 as an evolution from the author- to the concept-centric orientation where the goal is to "*aggregate (…) and compare*" information to make a comprehensive sense of it (Okoli and Schabram 2011, p. 30). The "*synthesis [of qualitative data] by interpretation and (…) explanation*" led to analysing forty-one

literature documents and other sources "*by looking at what people do [and write about], observing things that work [and what happens in the field], and then looking for the 'core of the solution'* " (Fowler 2002, p. 10; Okoli and Schabram 2011, p. 30). While many potential pattern candidates could have been established, due to a limited scope of this work, a decision was taken to describe only ten DS design patterns.

Once a broad outline of specific problem and solution was given, what followed next was a qualitative survey of R and Python tools in order to provide typical examples of how two formal systems with corresponding ecosystems can address a described issue. Hence, for each pattern candidate and programming language, a review of DS tools was conducted.

Qualitative Survey Surveys have been usually associated with quantitative descriptions "*of variables (…) in the population*" by way of establishing frequencies that can be extracted, illustratively, from interviews (Jansen 2010). However, as it has been shown in section 2.1.3.2, qualitative types were administered too. In relation to them, Jansen (2010) has broadly stated their purpose as "*determining the diversity of some topic of interest within a given population*".

The objective of this work is not only to formalize DS design patterns but supplement them with R and Python code examples that utilize tools from both ecosystems. Therefore, through a gained knowledge of their landscapes, address a third research question leading to establishing DSTM that helps both newcomers as well as experienced data scientists acquiring an overview of what the relevant tools are. Consequently, this open-ended investigation was answered in a qualitative matter as well.

Searching for Tools

Once having a blueprint of a pattern candidate, a sample of tools addressing identified problem was *purposefully* gathered. This was in line with stated delimitations and authors such as Patton (2015) and Sandelowski (1995) who have argued that all sampling in qualitative research is "*based on a specific [objective] rather than [chosen] randomly*" and for that reason "*represent[ing] the diversity of the phenomenon under study within the target population*" (Teddlie and Yu 2007, p. 80; Coyne 1997). Accordingly, while the *target* population relates to all tools in the whole R and Python ecosystem, the smaller *study* population is specifically linked to DS where they provide technical solutions to described obstacles (Hajimia 2014). On these grounds, packages coming from two software repositories exhibit *homogeneous* characteristics, allowing them to adequately fit into the surveyed DS domain due to being of a similar kind in the scope and intentions (Hajimia 2014; Suri 2011; Persson 2004).

For compiling a sample of R and Python packages, suitable key words and phrases to each pattern were used for searching the web in addition to inspecting:

(a) forty-one information sources,

(b) dedicated books including Everitt and Hothorn (2006) or

(c) websites of software organizations and foundations like *NumFOCUS*.

Relying on "*data saturation in order to decide when to stop*" searching for new applications, an attempt was made to cover only frequently described and suggested programs which are representative of the R and Python universe (Creswell 2013; Pukhovskaya 2014, p. 13).

Once seemingly applicable library was identified from one of the previously mentioned sources, before recording it in the database (Figure A.23), it was similarly screened and quality appraisal was administered according to the criteria listed in Table B.7. These principles had an objective to understand in-depth its aim and judge whether it provides a solution to a specific problem that was described by a pattern candidate. While preferring to be available on CRAN or PyPI, packages being in development for instance on *GitHub* were also considered. Even though tools might have a clear data science-oriented purpose and are extensively documented, they were excluded due to not having a new release since January 2016. Therefore, all displayed packages and frameworks in the DSTM shall demonstrate a strong community engagement that is manifested for example by developers helping other users in forum discussions. DSTM

3.1.1.3 Pattern Writing and Evaluation

To coherently draft discovered "*effective solutions and recurring problems*", a dedicated form needs to be followed (Inventado and Scupelli 2015, p. 5). Such "*uniform structure to the information*" makes "*design patterns easier to learn, compare*" and apply, resulting into communicating their essential set of information (Gamma et al. 1994, p. 16; Meszaros and Doble 1997). However, in section 2.2.2 it was observed that authors' use of *pattern templates* has differed, principally in naming conventions, order of appearance as well as in the mandatory and optional elements. Nevertheless, through works of Wellhausen and Fiesser (2011, p. 1), Meszaros and Doble (1997), and Douglass (2002), a pattern form was chosen consisting of nine items "*enabl[ing] to write a pattern in a simple but complete format*", see Table 5 (Fowler 2002, 2006).

After design patterns are practically implemented, *pattern evaluation* is the last (stand-alone) stage in the 3D2P methodology, where their subsequent use is evaluated (Inventado and Scupelli 2015). Albeit not conducting any practical assessments or deliberations with experts in the field, it was nonetheless necessary to iteratively refine and adapt pattern candidates to strengthen their case for DS domain. The reason why from the beginning it was referred to them just as the *candidates* is the fact that they were "*interpreted from data containing partial, incomplete measures [and] therefore they are considered potential [ones] until verified*" (Inventado and Scupelli 2015, p. 5). Consequently, a decision was taken that once each was sketched out, it was looked at whether one can find, implicitly or explicitly, its practical use. In fact, not only this was done before patterns were described in **chapter IV** (during *pattern prospecting* and *mining* stages) but also after each pattern was already finalized. All this in order to further increase their validity, Quality Validation

quality and substance, resulting to be "*shared more confidently with the community*" of stakeholders and be "*satisfied with [their] definition*" (Inventado and Scupelli 2015, p. 5). Moreover, this step has allowed to better recognize a need when more data were necessary to gather and for that reason repeating the aforementioned research procedure.

3.2 Summary

To summarize, **this chapter** has described the path to the discovery of design patterns in the context of an interdisciplinary field that combines deep domain understanding, expertise in the computer science and an informed use of scientific techniques to uncover valuable knowledge from data. The research strategy, by adopting interpretivist approach to the study, was based on the qualitative methods of inquiry through analysis of data by interpretation (Okoli and Schabram 2011).

Indeed, the foundational element of this work – by which ten design patterns were identified – lied in gathering and synthesising information which was extracted from a sample of sixty-six documents that were published in journal articles or in books (Trochim 2006). Accordingly, after the quality appraisal, only forty-one sources were used to find "*recurring [DS] problem[s]*" that could lead to pinpointing "*a solution, which might be drawn from background knowledge or literature*" (Inventado and Scupelli 2015, p. 5). Once sources were coded using the GIA, the overarching labels were established helping to examine "*relationships between data features that suggest (…) [DS] problems or effective solutions*" (Inventado and Scupelli 2015, p. 4). Afterwards, an initial concept of ten design pattern was formulated which was constantly refined and further improved.

Simultaneously with formalizing design patterns, a qualitative survey of mainly CRAN and PyPI was made to recognize packages and frameworks that address a specific design issue. The provided illustrative examples dedicated to each DS design pattern have allowed to develop Data Science R and Python Toolkit Matrix displaying tools from both ecosystems that can be used as a solution to a problem. At the same time, when put into a map, show an overview of two software landscapes.

Overall, by following the outlined study approach, it has permitted to answer the second and third research question where, successively, DS design patterns were developed and further combined with a view into R and Python software ecosystem. The results obtained by using this methodology are described in the **next chapter**.

Table 5: Presents pattern template sections.

Name of the pattern	**Context**, a "*description of the world*" and reality in which it can be applied (Delibašić et al. 2008)
Problem describing a faced issue and which is to be addressed	**Forces** examining considerations when choosing a solution to a problem and why it is hard to solve it
Solution detailing approach to an obstacle taking forces into account	Positive and negative **consequences** arising from its use
Known uses that illustrate application of a pattern, potentially including some specific variations	Where appropriate, **related patterns** that are in relationship with others or provide alternative solutions
R & Python **code samples** to illustrate its use	

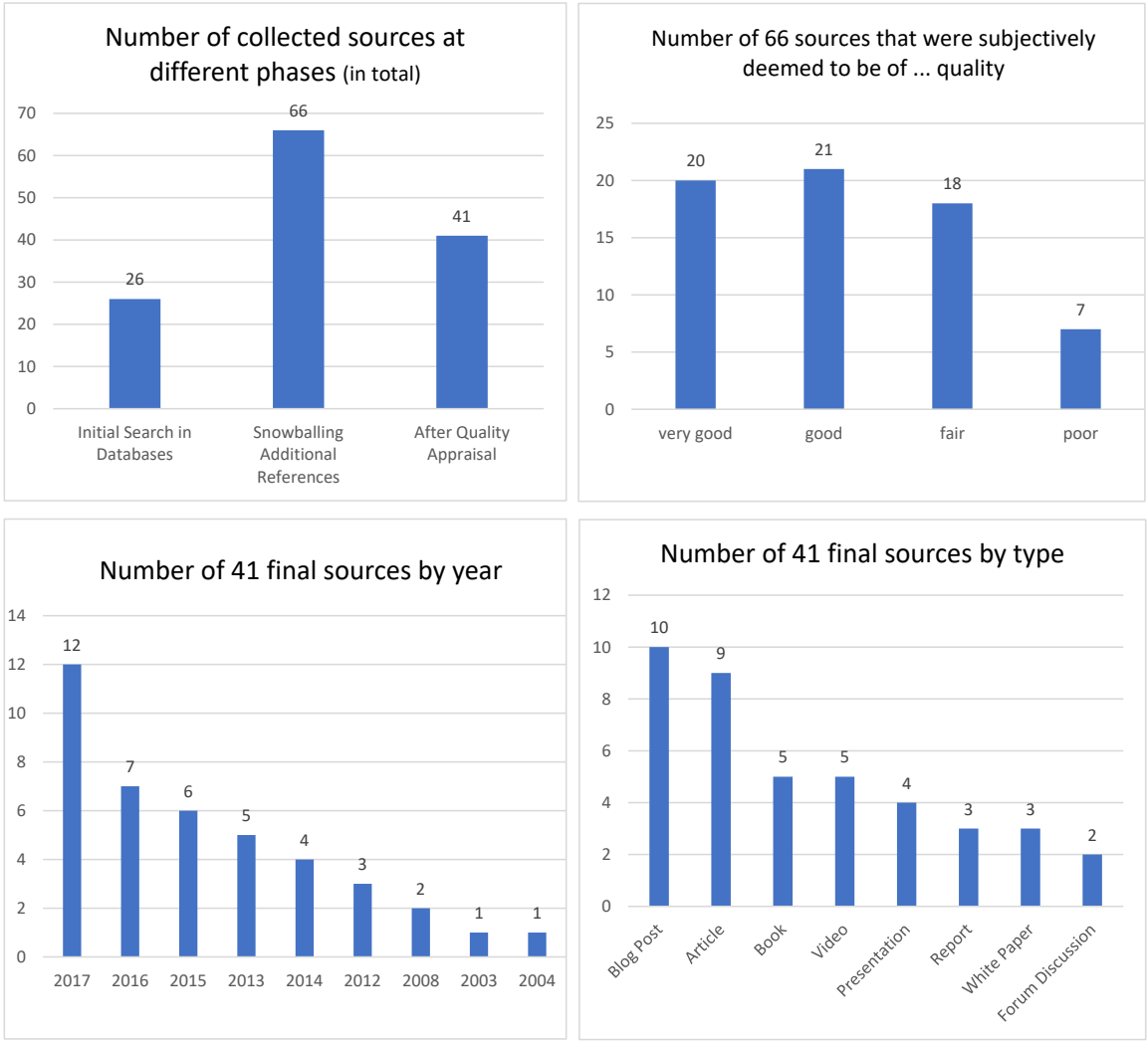


Figure 2: Illustrates statistics of gathered information sources.

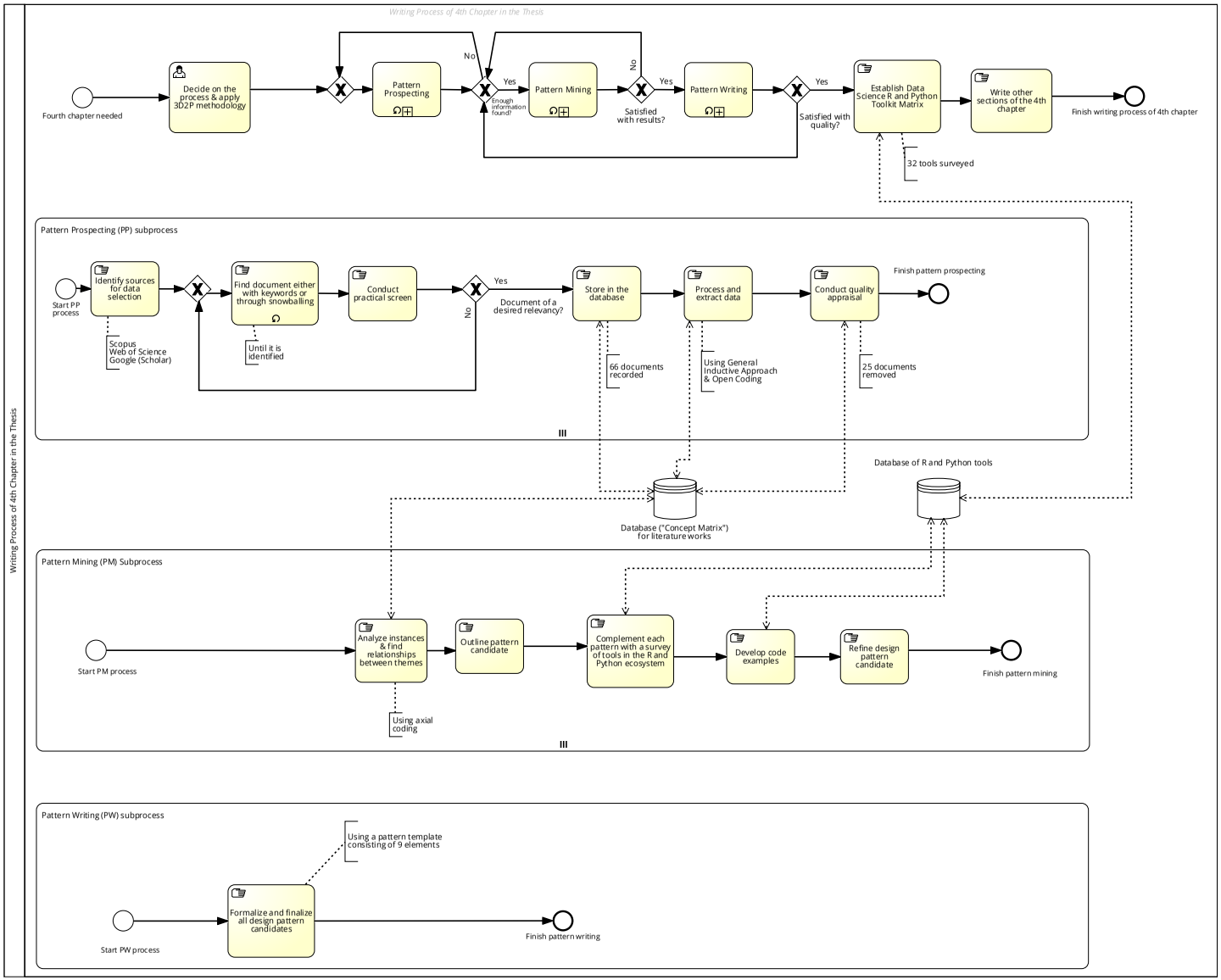


Figure 3: Illustrates an adapted and simplified pattern mining approach based on 3D2P methodology consisting of three main iterative stages, here "processes", being modelled using *Business Process Model and Notation 2.0*, see `BPMN_MDesign.pdf` (Okoli and Schabram 2011; Inventado and Scupelli 2015; Thomas 2006).

CHAPTER IV

Data Science Design Patterns

Having outlined key concepts and methodology, and after literature studies were collected, inspected and coded using general inductive approach, this chapter constructs and formalizes ten data science design patterns. It then establishes Data Science R and Python Toolkit Matrix too.

4.1 Pattern 1: Notebook

Context An iterative, interdisciplinary DS process, as illustratively seen on the CRISP-DM methodology and Figure A.19, consists of many different phases that are accomplished during a life-cycle of an analytical project.

Problem To enable easier communication with stakeholders in the course of insights discovery from data large and small, it is necessary to document approaches to questions being asked and manage acquired knowledge in a systematic way.

Forces

- DS is an incremental procedure where exploratory analysis needs to be explicitly described alongside the information such as what are the business objectives to solve, who are the stakeholders involved, where do data come from and how are they treated.
- Being related to the software engineering, the aim is to stay within a workflow of the centralized repository for source code management. Thus, the document should be stored in an open format enabling tracking of changes in it by means of a version control system like `Git`.

Solution Initially, KDD process may start from DS *notebooks* which support a lightweight markup language such as Markdown used to chronicle the progress and that in addition are

accompanied by specifically marked code snippets. Upon running this master file, results are printed in line with the other text rendered too. In short, it combines source code with readable description that is capable of recording not only small comments but also DS project plan with gathered information from the stakeholders. At the same time, *notebooks* present data analysis in a visual and concise form as well (Wilson et al. 2017; Nichols et al. 2017).

Consequences

Benefits: Sharing *notebooks* simplifies reproducibility due to publishing the source code with graphics (VanderPlas 2016). Likewise, by transparently communicating research activities that lead to achieved outcomes, they enhance organizational collaboration and memory through careful document management (Whitmore 2016; Sutton 2012). Moreover, they permit fast collection of the feedback from the DS team by executing individually *notebook's* code examples, thus promoting lean and interactive development style (Eslami 2012; Boire et al. 2003).

Liability: Being split into smaller code chunks instead of stored in cohesive files, source code becomes hard to orient in. Because of a tendency of having code snippets across multiple *notebooks* ("here and there"), it creates a need for following a priori and well-defined structure with appropriate naming conventions for data, files and folders to minimize plausible disorganization (Cady 2017).

Known uses

- *Notebooks* encourage quick prototyping and iteration on modelling and processing steps for the data analysis, see later *Prototyping Design Pattern*.
- Because of richness of supported presentation formats, they are used for storytelling of insights through an explanation of chosen approaches, conducted tasks and a discussion of actionable results.
- Additionally, authors like VanderPlas (2016) utilize these to write tutorials, data dictionaries, technical reports and even books teaching computer science concepts or programming languages.

Related Patterns *Notebooks* can contain embedded interactive visualizations which enable users to interplay for instance with check boxes or sliders, see *Interactive Application Design Pattern*.

Not having DS project including *notebooks* checked into the previously mentioned version control systems such as `Git` that supplement recording of a *project history* culminates into lacking a holistic view of its past development and ability of restoring changes by going back in time.

Sample Code Practitioners and researchers can choose between `RMarkdown` package (typically in conjunction with RStudio IDE) and `Jupyter`, a browser-based application. Both tools provide functionality for keeping a DS journal and support many different output formats including HTML. Similarly, they are programming language independent whereby R, Python, Ruby and Haskell could all be used within the same file.

To organize the R and Python source code and simplify its reproducibility, where appropriate, all snippets in **this chapter** are presented by means of either instrument, see Figure 4.

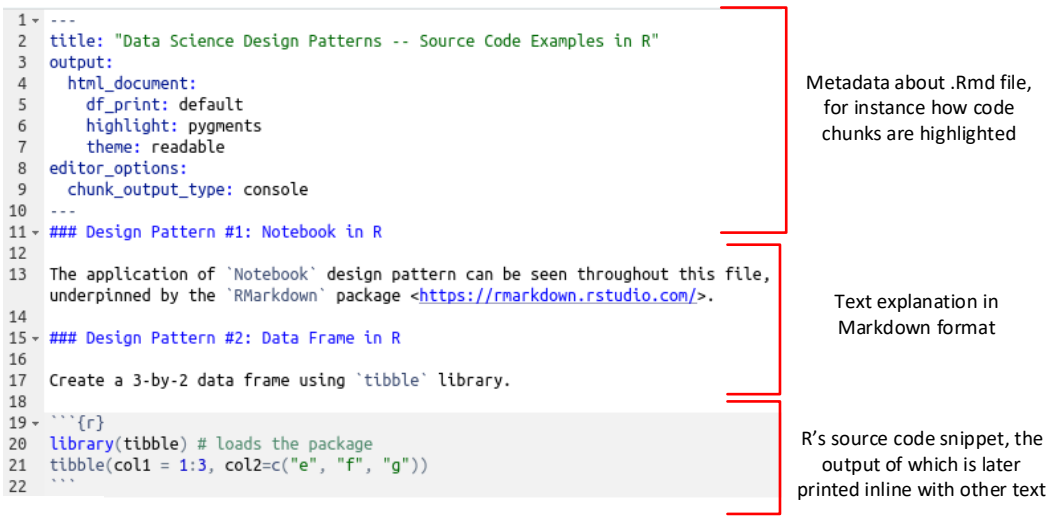


Figure 4: The example for *Notebook Design Pattern* shows R Markdown file (`code_data/R_masterThesis.Rmd`) consisting of R snippet alongside the comments. When this file is executed, HTML page is rendered, part of which can be seen in Figure 5 of the next design pattern. The alternative utilizing `Jupyter` can be found in `code_data/Py_masterThesis.ipynb`.

4.2 Pattern 2: Data Frame

Context After stakeholders asking a set of questions that have a direct impact on company's sales, hypotheses together with an unambiguous project plan and well-defined objectives and scope are developed. Next, right data sources are identified which is finally followed by their acquisition.

Problem When using programming languages like R and Python, data scientists need to programmatically manipulate and query two-dimensional data by applying one common interface.

Forces

- Data are stored in a variety of locations and formats which encumbers any further combined operations on them.
- Providing wisdom about business operations by storing high-quality internal and external data in the centralized DW (Figure A.18) is significantly limited by the available SQL functions which usually do not contain advanced statistical methods, thereby prohibiting their analysis.
- These days, however, there is a need for flexible application of research's latest and cutting edge algorithms that are typically not immediately accessible even in the conventional BI portals, ETL and visualization tools.

Solution Another fundamental principle on which DS depends are *data frames*. These abstract and most commonly in-memory data structures allow engineers to store and process various types of raw information in a unifying fashion and without a reliance on other third-party applications, mainly the database management systems. Being in a tabular form, *data frames* consist of rows representing observations and which might be heterogeneous and columns describing variables with labels (these should be as a whole homogeneous; Mikut and Reischl 2011).

Consequences

Benefits: Taking an inspiration from the data warehousing, *data frames* can be sliced and diced using operations such as `subset()` or `aggregate()` for further data manipulation, including their visualization (Trujillo et al. 2001). Besides treating missing data in a consistent matter, they can be also serialized into binary files, and thus easing the reproducibility and sharing of the results (VanderPlas 2016).

Liability: Storing a condensed *data frame* is significantly restricted by the available RAM, and therefore persistence of data, its velocity and volume being important properties have to be considered in advance (Walkowiak 2016; Halper 2016). Additionally, because data structures vary by underlying implementations, any performance tuning and exposed operations and syntax through API, it brings a gradual learning curve to known them thoroughly when working with multiple languages and tools offering similar functionality.

Known Uses

- A typical use case scenario is extracting and loading data for example from databases or spreadsheet files, see Zumel and Mount (2014) and Bruce and Bruce (2017).
- Principally, being essential in the end-to-end KDD process, *data frames* are used at all

DS stages for steps that include pre-processing, merging data with other datasets and transforming them for exploratory analysis and statistical modelling.

Related patterns *Data Matrix* is another two-dimensional data structure on top of which *data frames* are build. Yet, its implementation details differ in the R and Python universe. While R's natively offered `matrix()` as well as `Matrix()` from `Matrix` package can only consist of data of the same type, for example only integers, `matrix()` implemented in Python's `NumPy` package may hold data of different types too (Walt et al. 2011; Maechler and Bates 2018). Therefore, being more like *data frames*.

Sample code Two R (`tibble` and `data.table`) and Python (`pandas` and `blaze`) packages were identified providing a mechanism to store data in a tabular format, see Figure 5.

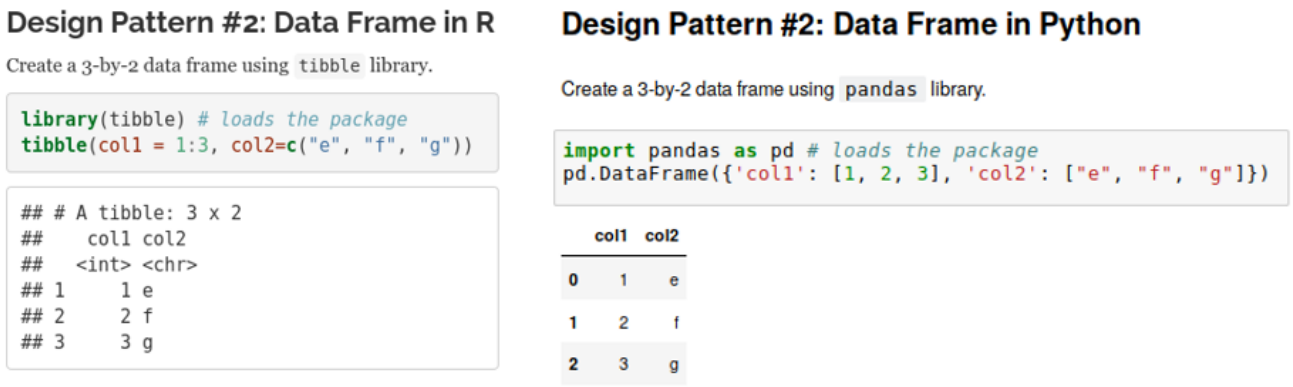


Figure 5: The example for *Data Frame Design Pattern* demonstrates two *data frames* being created. While on the left `tibble` is displayed when rendering the previous Figure 4, on the right Python's `pandas` is shown. The source code for this pattern is located in `code_data/R_masterThesis.Rmd` file, `code_data/Py_masterThesis.ipynb` respectively.

4.3 Pattern 3: Tidy Data

Context While DW may already store high-quality, rectangular tables in an easy to work with structure, externally captured information is organized in a variety of formats and diverse in their shapes. After importing it from different IT systems and web services, data frames are formatted to facilitate knowledge discovery. Although processing has been argued to take the largest effort, depending on studies anywhere between 30% and 80%, the accurate and clean data stored in a right way make it simple to handle any DS task (Zeutschler 2016; Taft 2015).

Problem Most typically encountered are messy data which are usually organized in a *wide* format where a table contains one observation (row) across many different variables (columns) for each subject that can be for example a person or a country (Boehmke 2016). Hence, they need to be efficiently changed to become ready for the subsequent analytical investigation.

Forces

- Modifying (big) data by hand may not be accomplished due to their sheer size, complexity and significant time effort leading to potentially introducing errors and having a non-agile DS by being distracted from deriving the actionable business results.
- At the same time, different KDD steps, methods and tools expect one exact arrangement of data forcing professionals to reshape them frequently in the journey through the data pyramid.
- Working with data that may contain several variables "*in one column*" or "*in both rows and columns*" affects the inquiry on account of having difficulties in their understanding as well as conducting any exploratory operations (Wickham 2014, p. 5).

Solution Spearheaded by Wickham (2014, p. 5), the concept of *tidy data* has been suggested to "*provide a standard way of structuring a dataset*". Modelled after Codd's (1970) third normal form, three core principles were specified, namely (a) each variable forming a column, (b) each observation forming a row and (c) each type of observational unit forming a final table. As such, this manifests into the *long* data structure where a table spans each subject over multiple observations because of having a column (a key) with variable types such as land area or a population and another one for their corresponding values. Being a part of a broader data management where metadata, data stewardship and security are of an importance, the technique of tidying the dataset into a particular layout has become a design convention for organizing analysis-friendly information (Wilson et al. 2017; Boire et al. 2003; DAMA 2017).

Consequences

Benefits: Having data in the *long* format simplifies their manipulation and further transformation including filtering or ordering (Zumel and Mount 2014). In this matter, *tidy data* increase users' overall experience by leveraging a consistent approach in coding style due to a potential availability of closely related tools that make the analysis flow easily together, for instance with a sequencing "%>%" pipe operator (Boehmke 2016; Grolemund and Wickham 2018).

Liability: Contrarily, *wide* data might be more straightforward to enter and inspect as they can have column names as values like "day1" or in ranges such as "$10-20k" (Wilson et al.

2017). Moreover, albeit language neutral, the concept is largely R centric where it stipulates a domain-specific language, a "*sub-dialect of the R*" called *tidyverse* – thus plausibly not directly replicable to other languages (Rickert 2017).

Known Uses

- Principally, the evidence stored in the *long* style is most frequently imperative for the modelling and visualization purposes and corresponding packages that enable it, see *Prototyping* and *Interactive Application Design Pattern* (Wickham 2014).
- Within the R's *tidyverse* sub-ecosystem, data in the corresponding shape play an integral role as they work hand in hand with complementary tools, including the previously mentioned `tibble` (Grolemund and Wickham 2018).

Sample code For R, a variety of options exist that provide operations for *tidy data*. When data frames are build using `data.table`, it also offers `melt()` and `dcast()` functions for their reshaping which are claimed to outperform the identically titled ones from the original `reshape2` package (Dowle and Srinivasan 2018). Additionally, within the tidyverse ecosystem, a dedicated application has been developed named `tidyr` too, see its use in Figure 6.

On the contrary, the previously mentioned `pandas` is indispensable in the Python's DS ecosystem due to being comprehensive in capabilities it provides, including for data transformation. Nonetheless, when data are in a more traditional shape of an array or a matrix, `NumPy` library, being a part of the *SciPy* sub-ecosystem, offers its own methods as well.

Figure 6: An example for *Tidy Data Design Pattern* shows how a data frame is created using R' `tibble` and subsequently transformed from a *wide* (messy) to a *long* (tidy) format. The source code for this pattern is located in `code_data/R_masterThesis.Rmd` file, `code_data/Py_masterThesis.ipynb` respectively.

4.4 Pattern 4: Leakage

Context After data cleaning and formatting, it is still rare that data sets are complete allowing the analysis and modelling to proceed further and make valid conclusions to stakeholders reflecting the stated objectives. Indeed, when some information is missing or not being in the right quality and quantity, among some not always available and preferred options, it can be crowdsourced from users or be acquired from third-party providers (Domino 2017).

Problem Before ML prototypes are created, data scientist need to solve table's missing data as their occurrence is undesirable for many algorithms which often simply omit potentially valuable observations with some lacking values. Therefore, the analysis of incomplete facts might be misleading and absent data need to be predicted or otherwise "imputed".

Forces

- Dropping rows or columns which contain missing values is universally not advisable as it reduces a sample size and introduces a bias to parameter estimates like correlation or standard error (Gelman and Hill 2006).
- At the same time, replacing missingness with median, mode or mean can lead to biased estimates too due to outliers, not considering relationships between features and general uncertainty about the outcomes that is not reflected in the imputed data (Little and Rubin 2002).
- The other so-called *single imputation method* may involve regression where missing values are predicted from complete observations using a regression model. Though, as described by Zhang (2016, p. 6) and García-Laencina et al. (2010), the "*variability of missing values is underestimated*" by this approach as only a single regression curve is created.

Solution In order for data scientists to appropriately handle missing data, one needs to answer an integral question, namely why are they missing. According to Karpievitch et al. (2012, p. 3) three missing data mechanisms are distinguished and consequently the approach to a treatment "*should ideally rely on the mechanism that caused the values to be missing*" in the first place.

Little and Rubin (2002) have talked about a concept named *missing at random (MAR)* where missing data (Y_{mis}) are related to those observed (Y_{obs}) and can be predicted from them. A special case of MAR is when the nature of missing data is not related to input variables – then data are said to be *missing completely at random (MCAR)*. This usually happens when the equipment malfunctions or there is an error in data entry – thus Y_{mis} is neither related to Y_{obs} nor to Y_{mis}. Both cases are said to be *ignorable* because the reasons for missingness

are ignored and "*it is appropriate to impute*" such data (McCleary 2002, p. 340). At last, a *nonignorable* situation named *missing not at random (MNAR)* means that the probability of missingness depends on the missing variable itself, for example when a "*sensor cannot acquire [statistics] outside a certain range*" (García-Laencina et al. 2010, p. 266; Schafer and Graham 2002; Gelman and Hill 2006).

Even though, the best solution is not to have missing data, an advanced statistical model can be specified to predict missing values (Hasan et al. 2017). Particularly under the MAR condition, of a great interest to data scientists should be *multiple imputation* where missing values are imputed m (typically three to twenty) times creating m complete but different datasets (Schafer and Graham 2002). Then, a study of such missing data is conducted and finally results are pooled (averaged) together by incorporating the "*variability between [and within] the m analyses*" (McCleary 2002, pp. 340-341).

Consequences

Benefits: If the underlying assumptions were met, *multiple imputation* allows missing data to be accounted for in a statistically valid and unbiased way (Hasan et al. 2017). Showing to be flexible and performing well under the different conditions, it reflects the uncertainty of true missing values due to having multiple sets of complete data which are combined to derive final results – by preserving the sample size and using all available information (McCleary 2002; Schafer and Graham 2002).

Liability: When data set is large and contains significant amount of missingness, *multiple imputation* becomes complex to compute (due to creating a "right" model), analyse and combine together (Horton and Kleinman 2007). Even though selecting a single best-looking imputation can be necessary for the *Prototyping* task, it is important to keep in mind that true values do not exists, and thus these proxies cannot be taken with absolute certainty (Mittag 2013).

Known Uses

- The complete case analysis can be considered when only a very small amount of data is missing ($\leq 5\%$) and being under the MCAR condition (Schafer and Graham 2002; Azur et al. 2011; Heijden et al. 2006).
- The simplistic methods of mean and regression treat an imputed single value as a true point. Yet, both make biased estimates too as they fail to account for variability in the data and should not be generally used (Takahashi and Ito 2012; Baraldi and Enders 2010).
- Schafer and Graham (2002) and Azur et al. (2011) have recommended two state-of-

the-art approaches which are aforementioned *multiple imputation* and its a basic alternative called *maximum likelihood with expectation-maximization (EM) algorithm* further described by Baraldi and Enders (2010) and García-Laencina et al. (2010).

Sample code Given R's strong statistical foundations, a variety of packages exist that address imputation of missing data. Notable of these are `mice` and `VIM`. On the other hand, for Python and exploration of missing data, data scientists can utilize `missingno` package while for the actual imputation `fancyimpute` library shown in Figure 7.

Design Pattern #4: Leakage in Python

After loading `fancyimpute` library as well as importing `New York's 1973 air quality` data set into Python's environment, it is inspected and observed that it contains 44 missing cases.

```
airquality.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 153 entries, 0 to 152
Data columns (total 6 columns):
Ozone      116 non-null float64
Solar.R    146 non-null float64
Wind       153 non-null float64
Temp       153 non-null int64
Month      153 non-null int64
Day        153 non-null int64
dtypes: float64(3), int64(3)
memory usage: 7.2 KB
```

Now, using `Multivariate Imputation by Chained Equations` with `Predictive Mean Matching`, missing data are imputed and a complete data frame is derived.

```
imputedValuesArray = fi.MICE(impute_type="pmm").complete(airquality.values)
completeDF = pd.DataFrame(columns=airquality.columns, data=imputedValuesArray)
```

Figure 7: An example for *Leakage Design Pattern* shows how missing data can be imputed using Python's `fancyimpute` package. The source code for this pattern is located in `code_data/Py_masterThesis.ipynb` file, `code_data/R_masterThesis.Rmd` respectively.

4.5 Pattern 5: Prototyping

Context Executives have described in accordance with specific, measurable, assignable, realistic and time-related (SMART) criteria multiple deliverables that they want to receive with a goal of addressing their clients in a better way, and therefore increasing their value to the company. Such mapping of a business problem to DS tasks might include data acquisition and mainly

classifying clients into different segments, based on the past transactions predicting sales for the next month and developing simple association rules which might recommend them purchasing new products.

Problem After understanding the nature of data through the exploratory analysis, their transformation, cleaning and potentially imputing missing values, clients need to be classified, sales be predicted and recommendations have to be provided. Therefore, data scientists need to experiment with simple and complex algorithms that permit them to gain useful insights.

Forces

- Having a decisive impact on answering business relevant questions, a variety of statistical models and families exist from which professionals have to identify the right ones and evaluate their performance in line with stated intentions as they all differ in the purpose, interpretability, complexity and accuracy (Wujek et al. 2016).
- Furthermore, numerous ML applications have become influential as they are flexible for diverse sets of data, easily extensible accommodating new models and metrics and permitting to store intermediate modelling results as well – all leading to lowering the DS bar (Goebel and Gruenwald 1999; Mao et al. 2014; Eslami 2012).

Solution KDD professionals have to identify comprehensive and modular APIs that offer appropriate algorithms for developing ML *prototypes*. Subsequently, these shall be evaluated to understand how the predictive models perform on unseen data in the future.

When collected information is labelled through attributes, (semi-)supervised learning methods such as Support Vector Machines (SVMs) or naive Bayes are used for predicting the output from input data. Depending on different learners and the analytical problem dealt with, two groups of supervised tasks are distinguished – a *classification* where a target variable is a discrete category or a *regression* where a prediction is a continues numerical value. On the other hand, when data are unlabelled with no response variable Y, more exploratory, unsupervised learning algorithms like k-means clustering or principal component analysis are employed to establish groups of similar objects or reduce dimensionality of data (Wujek et al. 2016; Zumel and Mount 2014).

Continuing with supervised learning, using a *holdout* approach, entire data are at first divided into training (typically about 75%) and testing set (remaining 25%; Kohavi 1995). Then, chosen algorithms together with training data are applied for model building. Because evaluating prototype's performance on the same data does not tell how well it generalizes, a model is taken with a holdout set to predict outputs and compare its forecast's capability with true data

(Provost and Fawcett 2013b; VanderPlas 2016).

Consequences

Benefits: ML is applicable to diverse industries and use cases and when appropriately applied, a wisdom can be gained which helps decision makers to set right company's objectives and improve products and services to better serve their customers. Additionally, reflecting the goals, a variety of prototyping metrics can be used for assessing final prediction. For example, for classification a confusion matrix is used which "*summarizes the (…) predictions against the actual known*" classes (Zumel and Mount 2014, p. 94).

Liability: When complex models are applied, there is a danger of running into a lack of their interpretability (Lavrač et al. 2004). Hence, they need to be justified in addition to being regularly updated, verified and improved reflecting changing business operations (Wujek et al. 2016). Building models may at times also lead to dead paths and accordingly it is necessary to learn quickly from mistakes, and thus maintaining a hub of the past work can support the organization in gaining the knowledge (Domino 2017; Chu et al. 2007; Sculley et al. 2015). Insufficient quality and quantity of data has a significant impact on the model's predictive power too and for that reason data scientists have to constantly iterate on created prototypes and gather better information (Zinkevich 2016; Shan et al. 2015).

Known Uses

- In the unsupervised learning, typical tasks include segmenting (clustering) movies according to the user preferences or creating association rules where market basket analysis establishes customers' items that are frequently bought together (Hastie et al. 2017; Jain 2010; Zhang et al. 2012; Aguinis et al. 2012). Furthermore, when data have many potentially redundant features, one may benefit from quicker computation by attempting to reduce d-dimensions into m principal components that explain most of the variance in the data (Cady 2017; Maaten et al. 2009).
- In the supervised learning, in the case of regression, it has been attempted for example to predict forest's burned area based on the meteorological data (Cortez and Morais 2007). On the other hand, detecting spam emails has been extensively studied by many researchers as a conventional classification problem (Blanzieri and Bryl 2008).

Related patterns The exposure to the *bias-variance trade-off* has to be addressed as it can make ML prototypes not generalizable beyond the training data, see next *Cross-validation Design Pattern*. In addition, ML libraries and frameworks have support a range of other capabilities such as later described hyperparameter optimization using *Grid* or leveraging ensemble methods

through *Assemblage Design Pattern*.

Sample code Because of being in the context of a specific task, an illustrative `Pima Indians Diabetes` dataset is obtained from Smith et al. (1988) to predict whether females of Pima Indian heritage have diabetes, being a binary classification task. Consequently, this is used for illustration in the following three patterns as well.

In the R ecosystem, two prominent packages exist that provide comprehensive tools for building and evaluating models – `caret` and `mlr`. For Python, multiple alternatives are available too, most notably well-established `scikit-learn` for general-purpose ML.

Design Pattern #5: Prototyping in R

After acquisition, data are split into training (75%) and testing (25%) subsets using `caret` library.

```r
library(caret)
splitIndex <- createDataPartition(PimaIndiansDiabetes$diabetes, p = .75,
                    list = FALSE, times = 1)
trainDataFrame <- PimaIndiansDiabetes[ splitIndex,]
testDataFrame  <- PimaIndiansDiabetes[-splitIndex,]

training <- trainDataFrame[,input_colNames] # select only input variables
trainDiabetesCl <- trainDataFrame$diabetes  # store output class separately

testesting <- testDataFrame[,input_colNames]
testDiabetesCl <- testDataFrame$diabetes
```

Train a classification model using `k-nearest neighbors` algorithm and predict new values that can be used later in the confusion matrix.

```r
trMod <- train(
          y=trainDiabetesCl, x=training, # training dataset -- vector and variables
          method ='kknn',                # k-nearest neighbors from kknn package
          metric = "ROC",                # metric for evaluation
          trControl=trainControl(classProbs = TRUE,
                                 summaryFunction = twoClassSummary))

predictions <- predict(object=trMod, newdata=testesting, type='raw')
```

Figure 8: An example for *Prototyping Design Pattern* shows R's `caret` library when applied for a classification task using a holdout approach. The source code for this pattern is located in `code_data/R_masterThesis.Rmd` file, `code_data/Py_masterThesis.ipynb` respectively.

4.6 Pattern 6: Cross-validation

Context After initial understanding of preferably *tidy* data and the domain being dealt with, a supervised learning algorithm was selected to plausibly satisfy stakeholders' objectives. Succeeding the *Prototyping Design Pattern*, the model has to be evaluated as it may not generalize beyond the data which have been used for its creation, causing *overfitting*. This means that while originally the learner scored highly on the training data, when previously unseen information arrives, prototype's high complexity leads to a very poor accuracy of a new prediction as it fits noisy data (Zumel and Mount 2014).

Problem A major task for any data scientist is to balance the so-called *bias-variance trade-off* where on the one hand ML prototype has to account for relevant data features (have high accuracy; in statistical terms low bias) while on the other hand also avoid capturing meaningless data – that is to have high precision, statistically low variance (VanderPlas 2016). Consequently, one needs to assess learner's predictive power with future data samples using a robust technique which could aid in selecting a model with the lowest bias and lowest variance (Kohavi 1995).

Forces

- If training and testing prototypes on the same data set, they "memorize" them very well but this is not representative of the general data trend (Provost and Fawcett 2013b; Wujek et al. 2016).
- Previously, a classical approach tackling overfitting was described where data are randomly split into two sets. This makes a fair approximation by holding back some of the data to validate the prototype with new information. However, a portion of valuable data is not training the learner as the test set may be significantly influenced when a random split is not characteristic and balanced (Zumel and Mount 2014; Provost and Fawcett 2013b).
- Although a frequently named alternative is partitioning data into three sets (training, validation and testing), because they are usually small in sample size, a number of examples for ML is further reduced. Hence, either holdout approach excludes some information and prevents learning from complete data patterns (Cady 2017).

Solution *Cross-validation* is a sophisticated method for "*generat[ing] multiple performance measures*" with an ability to tell what to expect from the forthcoming data by estimating test set (so-called generalization) error (Provost and Fawcett 2013b, p. 140). Enhancing the basic holdout technique, it repeats "*the construction of the model on different subsets of the available*

training data and then evaluate[s it] only on data not seen during construction" (Zumel and Mount 2014, p. 111).

The widely used *k-fold cross validation* splits a data set into k, typically five or ten, folds with a goal of training and scoring the prototype k times (Provost and Fawcett 2013b). In each iteration of the cross-validation, $k-1$ different partitions are combined on which the learner is trained and finally its accuracy is estimated on the last fold deriving the overall test performance. Having multiple results, these are united by mean for checking the bias and standard deviation for the variance (Zumel and Mount 2014).

Consequences

Benefits: Compared to the holdout technique described in the *Prototyping Design Pattern* or *bootstrapping* outlined next, each sample created from cross-validation is certainly used for training and testing step, resulting into reducing the bias as the number of k parts increases (Gutierrez-Osuna 2002). Even though it has been sometimes argued that when cross-validation is repeated multiple times, model's variance decreases too, some researchers have not been able to confirm this hypothesis (Vanwinckelen and Blockeel 2012; Kohavi 1995).

Liability: Cross-validation is computationally (very) expensive as training and testing steps are rerun k times, specifically with a variation called *leave-one-out cross validation*. Despite $k = n$ number of examples, results can still exhibit high variance as well (Gutierrez-Osuna 2002).

Known Uses

- When new data cannot be acquired, cross-validation often becomes *the* method for model building and selection in DS through its universality and simplicity (Arlot and Celisse 2010; Hattotuwagama et al. 2008).
- Besides *k-fold cross validation*, variations such as *stratified* or *repeatable cross-validation* have been proposed as well. Whereas the objective of the former one is to address biased classes, thus balancing the distribution across each fold making it a good representative of the whole sample, the goal of the latter one is to repeat cross-validation n times creating random data folds whereby subsequent predictions can all be averaged (Kuhn and Johnson 2013; Vanwinckelen and Blockeel 2012).

Related patterns As seen next, *cross-validation* is frequently used with the *Grid* and *Assemblage Design Pattern*, particularly with the latter one which has the capability of further reducing the overfitting (Zumel and Mount 2014).

An alternative method to cross-validation represents *bootstrapping* which is mainly used for

obtaining confidence intervals and where a random sample with replacement of the same size as the original data set is taken for the training step. Then, the model is trained on each of them and before results are again averaged, it is also tested on the remaining "*examples that were not selected for training*" (Kohavi 1995; Refaeilzadeh et al. 2017; Gutierrez-Osuna 2002).

Sample code Using previously identified R's `caret` and Python's `scikit-learn`, a naive Bayes model is demonstrated next where it is applied to four data parts and tested on the fifth one, see Figure 9.

Design Pattern #6: Cross-validation in Python

After loading `scikit-learn` library, a `multinomial naive Bayes` algorithm for classification is specfied, with default hyperparameters.

```
clf = MultinomialNB(alpha = 1.0, fit_prior = True)
```

Then, `5-fold cross-validation` is selected with a seed value, whereby data are not shuffled before each split.

```
cv = KFold(n_splits=5, shuffle = False, random_state=32018)
```

At last, diabetes values of Pima Indians are predicted using the above-mentioned classifier and cross-validation.

```
predicted = cross_val_predict(clf, dt.iloc[:,0:8], dt.diabetes.values, cv=cv)
```

Figure 9: The example for *Cross-validation Design Pattern* shows application of `scikit-learn` whereby data are first split using five-fold cross validation to which a *multinomial naive Bayes* classifier is applied. The complete source code for this pattern is located in `code_data/Py_masterThesis.ipynb`, `code_data/R_masterThesis.Rmd` respectively.

4.7 Pattern 7: Grid

Context *Hyperparameters* are specified in the model before a training step can begin. While some algorithms have none and can be immediately used for a particular task, others like k-means clustering or random forest contain several that data scientists are required to set. Typically, in the latter case, one has to decide on a number of trees to build before averaging their predictions (Goodfellow et al. 2016). Overall, the importance of hyperparameters lies in having a significant influence on various costs of running the prototype as well as on its behaviour and final performance as seen for instance from *Prototyping Design Pattern*.

Problem Instead of making (non-)educated guesses to find best hyperparameters by trial-and-error, it is desirable to have an automatic procedure that optimizes them and therefore can

increase model's predictive power, its robustness and ultimately generalizability when facing new data.

Forces

- Albeit most frequently applied, searching manually for hyperparameters is very time-consuming effort because of their number and having necessary understanding of what they do and what impact they have on the learner (Goodfellow et al. 2016).
- Moreover, retraining each time the prototype because of manually adjusting a large set of hyperparameters can probably lead to omitting some better values.

Solution While designing DS pipeline, professionals should employ *grid search* which identifies hyperparameters that deliver best predictions. At first, a finite set of values for each hyperparameter is specified in a *grid* which can be in a matrix form through employing *Data Frame Design Pattern*, simplified for instance `{'k':[2, 4, 6, ..., j]}` and `{'size':[10, 100, 1000, ..., i]}`. Subsequently, the *grid search* algorithm takes the Cartesian product of previous sets $(i_{1...n} \times j_{1...n})$ with $n \in \mathbb{N}_{>0}$ and uses all combinations for training the model (VanderPlas 2016). After evaluating its performance, typically, by means of *Cross-validation Design Pattern*, the outcome of such comprehensive search allows to find best settings that should be utilized (Goodfellow et al. 2016).

Consequences

Benefits: Even though not guaranteed, choosing a right set of hyperparameters can boost learner's predictive strength by easily parallelizing the computation across a cluster of machines (Breck et al. 2016). Due to being a fairly general approach, one can optimize not only hyperparameters themselves but also finding out the most excellently performing ML prototype as well.

Liability: Searching for best hyperparameters is very CPU intensive operation that increases modelling expenses exponentially, particularly when their number to fix is large. Besides being of $\mathcal{O}(n^c)$ time complexity with c hyperparameters and n number of their values, determining the optimal *grid* to search in may be a task for itself, thus the reliance on experience and third-party resources where similar data or methods were applied (Goodfellow et al. 2016).

Known Uses Hyperparameter optimization may still lead to overfitting and therefore one ought to combine it with *Cross-validation Design Pattern* (VanderPlas 2016; Min and Lee 2005). In order to ensure forecast's high-quality, data are at first divided into the training and testing part. Then, cross-validation splits the training set into k-folds. For each Cartesian product

from the *grid*, a model is trained $k-1$ times and tested on the subset that was left out, leading to record the highest average performance for a particular instance with specific settings. At last, the one with the best results would be selected for the final step of testing the data from their first separation (Olson and Delen 2008).

Related Pattern Besides aforementioned *grid* and *manual search*, scientists such as Hutter et al. (2015) and Wujek et al. (2016) have described other techniques of tuning hyperparameters that have shown to be more convenient and efficient too. One of the alternatives presented by Bergstra and Bengio (2012) is named *random search* which takes an arbitrary sample of specified parameters from the *grid* space using a probability distribution like Bernoulli. This has proved to achieve very similar results compared to the exhaustive *grid search* but much faster.

Sample code Generally, the majority of comprehensive ML packages like those from *Prototyping Design Pattern* already offer functionality for hyperparameter optimization, including *grid* and *random search*. Previously used applications `caret` and `scikit-learn` are examples of it. Consequently, in Figure 10, `caret` package trains a naive Bayes model five times and evaluates it by applying a *Cross-validation Design Pattern*.

Design Pattern #7: Grid in R

Training a naive Bayes classification model and applying `Cross-validation Design Pattern`.

```r
library(caret)
ctrlFunc1 <- trainControl(
              method = "cv", number = 5,        # 5-fold cross validation
              summaryFunction = twoClassSummary,# binary classification
              classProbs = T)                   # task

predModel <- train(
              y=trainDiabetesCl, x=training,  # training dataset -- vector and variables
              method='nb',              # naive Bayes from klaR package
              trControl=ctrlFunc1,      # training control function
              metric = "ROC",           # metric for evaluating the model
              tuneGrid=expand.grid(     # search 2^3 = 8 combinations in the grid
                fL=c(0.1,0.2),          # 2 values for penalty
                usekernel=c(TRUE,FALSE),# whether kernel or normal density
                adjust=c(0.4,0.5)))     # bandwidth adjustment
```

Figure 10: The example for *Grid Design Pattern* show how naive Bayes model can be trained with `caret` library. More specifically, searching the grid with eight hyperparameters and applying five-fold cross-validation. The complete source code for this pattern is located in `code_data/R_masterThesis.Rmd`, `code_data/Py_masterThesis.ipynb` respectively.

4.8 Pattern 8: Assemblage

Context In the *Prototyping Design Pattern* it was noted that multiple different algorithms could be used for unsupervised and supervised learning. Particularly in the latter case, after training a baseline model and even optimizing its hyperparameters, data scientists ought to judge its performance not only by distinct metrics but also see it in the context of other prototypes to better understand how they behave and compare to each other. All that in line with DS premise of being an iterative process where several alternatives should be build and evaluated.

Problem To further improve achieved results, besides increasing volume of data and their quality through extensive pre-processing, applying for instance *Leakage Design Pattern* and engineering new features, data scientists may need to intelligently combine outcomes of many diverse models for gaining a greater predictive accuracy in contrast to individual methods alone.

Forces

- A multitude of tested estimators display only mediocre conclusions that are just slightly above a random guess.
- Moreover, even though cross-validation and grid search were administered, it is still possible to observe overfitting that hinders model's practical use for business purposes.
- The developed ML prototypes carry a large variance due to randomness in hyperparameters, probabilistic nature and selection of data (Zumel and Mount 2014).

Solution In the quest of designing high-quality model which is capable of "*reduc[ing] the effect of (…) overfitting*", engineers may leverage the power of so-called *ensemble methods* (VanderPlas 2016, p. 426). These integrate a collection of modelling instances from families such SVM or naive Bayes that are diverse and likewise perform differently on the same data (Nagi and Bhattacharyya 2013). Consequently, their *assemblage* shall lead to improved outcomes and especially *stacking* has been a prominent method. This leverages an independent and simple meta-model that combines other prototypes where at first they are individually trained in order to acquire not only their original features but predicted outputs too. Then, in the final step of meta learning, the base-level classifiers are basically "blended" by the super learner algorithm to possibly derive better results (Zhang and Ma 2012; Maclin and Opitz 1999).

Consequences

Benefits: *Assemblage* of methods attempts to address an instrumental trade-off in DS, namely reducing both high variance while at the same time high bias as well (VanderPlas 2016). Indeed, the accumulation of prototypes has shown to potentially improve final accuracy of

prediction by lowering generalization error on unseen data, notably when they contain high amount of variables along with being small in the sample size (Zumel and Mount 2014; Yang et al. 2010).

Liability: Being frequently applied with the *Grid* and *Cross-validation Design Pattern*, all three add to the required computing resources (Zhang and Ma 2012). Primarily, however, *assemblage* further hampers conclusion's interpretability and understandability by the managers creating an undesirable black-box (Bühlmann 2012). Furthermore, if predictions are strongly correlated, one will not be able to boost them with the *ensemble* (Provost and Fawcett 2013b).

Known Uses

- Due to universally claiming to improve returns on data of various quality and quantity, practical applications include winning DS competitions like Netflix Prize in 2009 where *Gradient Boosted Decision Trees* were applied on hundreds of individual learners to improve company's recommendation engine, see Koren (2009).
- Woźniak et al. (2014) and Nagi and Bhattacharyya (2013) have described other use cases such as improved detection of cyber-attacks, prevention of financial fraud, avoiding credit risk or in the medicine for diagnosis of diseases.

Related Pattern Although *stacking* is one of the more advanced *assemblage* techniques, other approaches exist as well (Wujek et al. 2016).

Considering random forests, these aggregate results from decision trees that are build using training subsets of data sampled with replacement and only with random features (Zumel and Mount 2014). By decreasing the variance, trees can be averaged in parallel to derive the best estimator – the procedure of which is called *bagging* (Maclin and Opitz 1999; Bühlmann 2012).

In contrast, *boosting* algorithms like AdaBoost reduce the bias and variance by incrementally building in sequence several "weak" models that are often only barely better than a random guess (Yang et al. 2010). Aiming to build a combined "strong" classifier, the idea is to learn from misclassifications of a model instance which culminates into reweighting training data at each round to emphasize shortcomings that should be addressed with a new iteration (Zhang and Ma 2012). Though, not being a silver bullet, overfitting can still occur with outliers and other noise in the data (Bauer et al. 1999).

Sample code For R, one can use an extension to the previously mentioned `caret` library called `caretEnsemble` or among others `SuperLearner` package. Contrarily, besides already shown Python's `scikit-learn` that can be supplemented with `Mlxtend`, the example in Figure 11 creates an ensemble of two methods using `ml-ensemble` utility.

Design Pattern #8: Assemblage in Python

After importing `scikit-learn` and `mx-ensemble` libraries, a `stacking ensemble` is created that applies 5-fold cross-validation. It scores the models according to best accuracy, while avoiding to shuffle data before each new layer.

```
ensemble = SuperLearner(folds=5, shuffle=False, scorer=accuracy_score,
                        random_state=32018)
```

Next, `random forest` and `SVM` algorithms are added to the base level and finally, a simple meta-estimator (logit classifier) is specified for the actual step of training and testing the ensemble model.

Only random seed is set -- all other hyperparameters take their default values.

```
ensemble.add([RandomForestClassifier(random_state=32018), SVC(random_state=32018)])
ensemble.add_meta(LogisticRegression(random_state=32018))
```

```
ensemble.fit(train_X, train_y) # train the ensemble model
y_pred = ensemble.predict(test_X) # predict the class outcomes
```

Figure 11: The example for *Assemblage Design Pattern* displays how a *stacking (logit classifier) ensemble* is constructed to use features from training data through SVM and random forest in its resulting prediction. The complete source code for this pattern is located in `code_data/Py_masterThesis.ipynb`, `code_data/R_masterThesis.Rmd` respectively.

4.9 Pattern 9: Interactive Application

Context After acquiring information about business activities and having answers to questions asked, being one of the concluding steps is when findings are visually communicated in an effective and understandable manner to stakeholders. This is done in order to add value to offered services and products and deliver return on DS investment.

Problem While the diversity of employee's background and roles manifests in seeking different perspectives to the same underlying data, this is not aligned with conventional reports and presentations which usually display static visualizations and are limited by their scope and a particular view presented – hence the need for a better alternative.

Forces

- Referencing big data properties, information changes rapidly and therefore static graphics may not be relevant in one-week time and so the conclusions based on them.
- Storytelling requires to visualize not only latest data but also enable audience to answer their questions by themselves through adjusting the way information is conveyed in an interactive fashion, for example by drilling it down, without unnecessary latency (Nguyen and Hyde 2015).

- Business managers are usually not interested in pure numbers without a context and explanation, and thus interpretable results told through diverse means that can easily translate into a set of operational processes and actions are indispensable (Provost and Fawcett 2013b).

Solution As an important part of the communication strategy, data scientists shall design and develop simple, web-based *interactive applications* using the identical programming language applied for other DS tasks to stay within the same workflow and avoid technological fragmentation. Indeed, such applications have to permit users to dynamically interact with presented tables or charts and customize them for their data journey by modifying various graphical and data attributes that reflect employees' own preferences, for instance filtering, zooming or changing the shapes (Ward et al. 2010).

Consequences

Benefits: Stakeholders are provided with a self-service tool that empowers further data exploration, gaining deeper insights by looking at information tooltips and understanding the story through animations and slideshows. The information being updated in close to real time makes visualizations display latest available data and consequently fostering a trust and a deeper integration in the organization by allowing to proactively react to the changing environment (Domino 2017; Cady 2017).

Liability: Developing high-quality *interactive application* carries an additional effort in its continuous maintenance due to ever evolving data sources, their granularity and employee desires (Clarke 2013; Halper 2016). Furthermore, it is necessary to understand the targeted technology platform and learn tools for its development. Besides, when the application has been improperly designed displaying excessive details, technical jargon or having a misguides choice of colours, it ultimately contributes to information overload and never being looked at (Eppler and Mengis 2004; Gershon 1998; Carr 1999; Nguyen and Hyde 2015).

Known Uses

- Dick (2014) and Segel and Heer (2010) have indicated that interactive stories may possibly enhance a reading experience of a journalistic work on the web.
- As already mentioned in the *Notebook Design Pattern*, dynamic visualizations can be embedded in the document making data analysis more visual from the beginning.
- BI portals, following best practises in the UI/UX design, make available a standard set of core components like sliders for creating KPI dashboards with interactive plots.

Related patterns Besides having *interactive applications* on-premise, organizations can take advantage of *cloud computing* services run by providers like Microsoft (*Azure*) or Amazon (*Web Services*). This in order to leverage features such as automatic scaling that ensures their uninterrupted availability, see next *Cloud Design Pattern*.

Sample code While `Shiny` is almost certainly the most widely used package for developing analytical and interactive web application using R, for Python besides `Bokeh` one can create them by utilizing `Plot.ly Dash` framework too, see Figure 12.

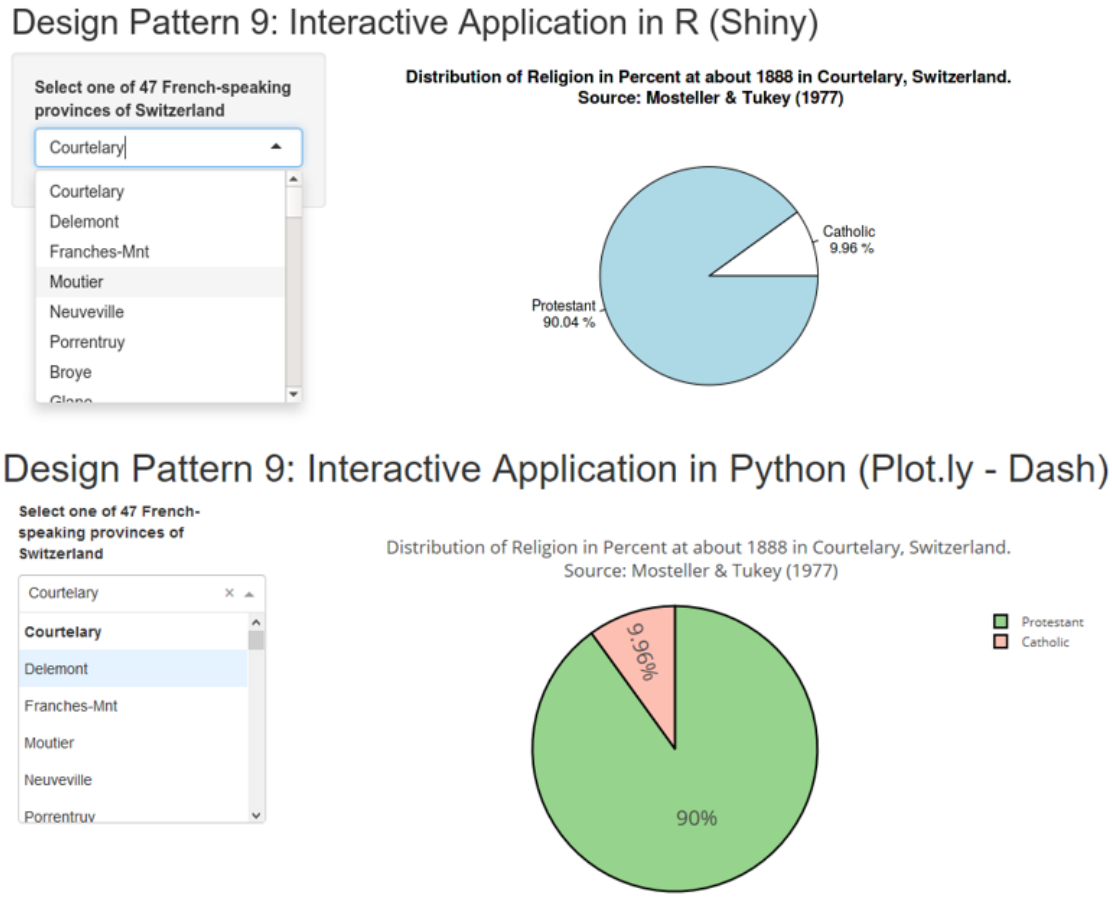


Figure 12: The example for *Interactive Application Design Pattern* shows exceptionally a result of R and Python source code located in `code_data/dp_9` folder. A simple pie chart with a dropdown component where users can choose from a list of provinces is displayed. This shows a distribution of religion in Switzerland based on the *Swiss Fertility and Socioeconomic Indicators (1888)* data, obtained through R's `datasets` library. Upon changing the region (the user interaction), the chart is immediately redrawn to reflect new values.

4.10 Pattern 10: Cloud

Context After making available on a local machine a proof of concept which is subsequently validated first by peers in the DS team and later by directly involved customers themselves, it is necessary to share the solution with all stakeholders of the organization by deploying it on the production system.

Problem Data scientists need to utilize computing resources in a flexible and simple matter without being constrained by the available hardware and software at company's premises. The objective of this is to allow for instance quickly scaling up and down DS interactive applications or integrating complex predictive models into firm's products and services.

Forces

- Organization's core mission is often unrelated to administrating and configuring hardware and software infrastructure. To avoid facing various constrains and difficulties, it has been claimed of being beneficial to outsource the management of servers to experienced specialists (Marston et al. 2011).
- Training ML models as well as pre-processing (big) data requires significant computing resources which might impede fast prototyping of DS solutions because of the unavailability of modern, on-premise CPUs, high-volume storage systems and RAM in the necessary quantity.

Solution Instead of building and maintaining own infrastructure, companies and data scientists should learn to utilize *cloud computing* made available by vendors like Google (*Cloud*) or Microsoft (*Azure*). While many different types of *cloud* services exist, see Zhang et al. (2010) for their overview, a core underlying feature is the ability of on-demand allocation of computing resources paid according to the pay-as-you-go pricing model. As such, the IT architecture including DS workflow shall be designed around leveraging *cloud* features where dedicated data mining virtual machines or enhanced applications for information storage and processing based on the *Hadoop* toolbox might be employed.

Consequences

Benefits: Perhaps the most important gain is avoiding managing on-premise infrastructure where instead a *cloud* portal permits a rapid provisioning of resources as they are deemed necessary by engineers. Additionally, it allows to scale organization's IT architecture organically and due to using a comprehensive platform which integrates with other offered functionalities,

data scientists can focus on understanding the data by taking advantage of a full set of related services (Patil 2013; Halper 2016).

Liability: Besides potentially expensive and hence necessity of monitoring used resources, selecting one provider can result into a vendor lock-in because of unique features that may not exist at competitors (or work in a similar matter; Opara-Martins et al. 2015). Moreover, security and privacy concerns can put *cloud computing* either in jeopardy or substantially increase technological and bureaucratic burdens due to legal compliance and regulations (Stone et al. 2010). Furthermore, companies may not be culturally and process-wise ready to take its advantages by reasons of lacking in-house expertise.

Known Uses

- Typically, the power of *cloud* complements natural language processing or object recognition in images and at the same time enables quicker training of predictive models and management of the whole DS workflow and ETL pipeline.
- Likewise, third-party *cloud* services can reinforce good software engineering practises such as continuous integration and platform agnostic development and deployment (Guha and Al-Dabass 2010).
- KDD newcomers without programming skills may use various *ML Studios* – cloud-based web applications, often in conjunction with *Notebook Design Pattern* – that offer a simple drag-and-drop functionality for building a visual pipeline that contains operations for importing data sources, their processing, model building and visualization.

Related Patterns As previously mentioned, *cloud computing* is applicable at many stages of KDD process – when both developing DS solutions including ML prototypes and interactive applications as well as during their subsequent final deployment.

Sample code In the past decade, due to a variety of established *cloud(-enhanced)* solutions by numerous providers, see Durao et al. (2014) and Zhang et al. (2010), the availability of R and Python packages that connect to these services is depending on vendors to provide them and the community interest to develop them.

Some examples include R' `cloudml` package that interacts with specific APIs offered by Google's *Cloud Machine Learning Engine*. Alternatively, `doAzureParallel` library supports parallel execution on Microsoft's *Azure* virtual machines. Similarly, several software development kits for Python exist as well. One of them is Amazon *Web Services*' `Boto3` package that encompasses numerous APIs which can access ML capabilities including what the company calls *vision* or *language services*.

For the simplicity of the illustration, previous examples from the *Interactive Application Design Pattern* can be deployed to two *platform-as-a-service (PaaS) cloud computing* environments. These provide a complete "*platform (…), including operating system support and software development frameworks*" for the management of applications while at the same time giving no control over the infrastructure itself (Zhang et al. 2010, p. 10; Pahl 2015). Consequently, `Plot.ly Dash` example is served by `Heroku.com`[3], whereas R's alternative using `Shiny` framework could be pushed to `shinyapps.io`, see Listing 2.

```
> library(rsconnect) # loads the package for shinyapps.io

# set up the user account with complete API keys, omitted for brevity
> setAccountInfo(name='dmpe',
        token='820739C8B8B9DE...',
        secret='gAHFGNCdDlUtzo...')

# once deployed, navigate the browser to it
> deployApp(appDir = "/code_data/dp_9/R-Shiny",
          appName = "DesignPattern10-InteractiveApplicationOnCloud",
          launch.browser = TRUE)
```

Listing 2: The example for *Cloud Design Pattern* displays how `Shiny` application can be deployed to a third-party PaaS hosting to leverage integrated features like automatic scaling and advanced monitoring. The source code for this pattern is located in `code_data/dp_10` folder.

4.11 Data Science R and Python Toolkit Matrix

As explained in the **chapter III** and seen previously, during pattern discovery and description, R and Python code examples were provided for each pattern candidate, see likewise `code_data` folder in the supplement. Indeed, thirty-two R and Python packages were identified having followed outlined criteria mentioned in Table B.7 and subsequently they were recorded in the database shown in Figure A.23. In line with the third research question and once all design patterns were refined, libraries were finally visualized in a mind-map in Figure 13. Thus, allowing to enact a foundational perspective on the ecosystems of two languages and their available tools (Suri 2011).

Moreover and where applicable, an attempt was made to present native approaches which were integrated into the respective programming language. Unfortunately, while four cases were identified in R, illustratively `reshape()` function that is related to *Tidy Data Design Pattern*, for Python none were recognized in its *standard library* that could directly interconnect with one of ten design patterns (Python Core Team 2018e).

[3] https://designpattern10.herokuapp.com/

4.12 Summary

Attempting to shed light on the second study question, after understanding key terms used in this work and presenting the methodology, **this chapter** was fundamental as ten data science design patterns candidates were formalized.

As stated in objectives of this work, it has been furthermore asked what R and Python tools from both ecosystems can be found solving common obstacles arising from the knowledge discovery process and being in relation to identified design patterns. By purposefully surveying both landscapes, a sample of thirty-two packages was visualized in the Data Science R and Python Toolkit Matrix as well.

In the **final chapter** and not being limited to, these core outcomes are individually discussed.

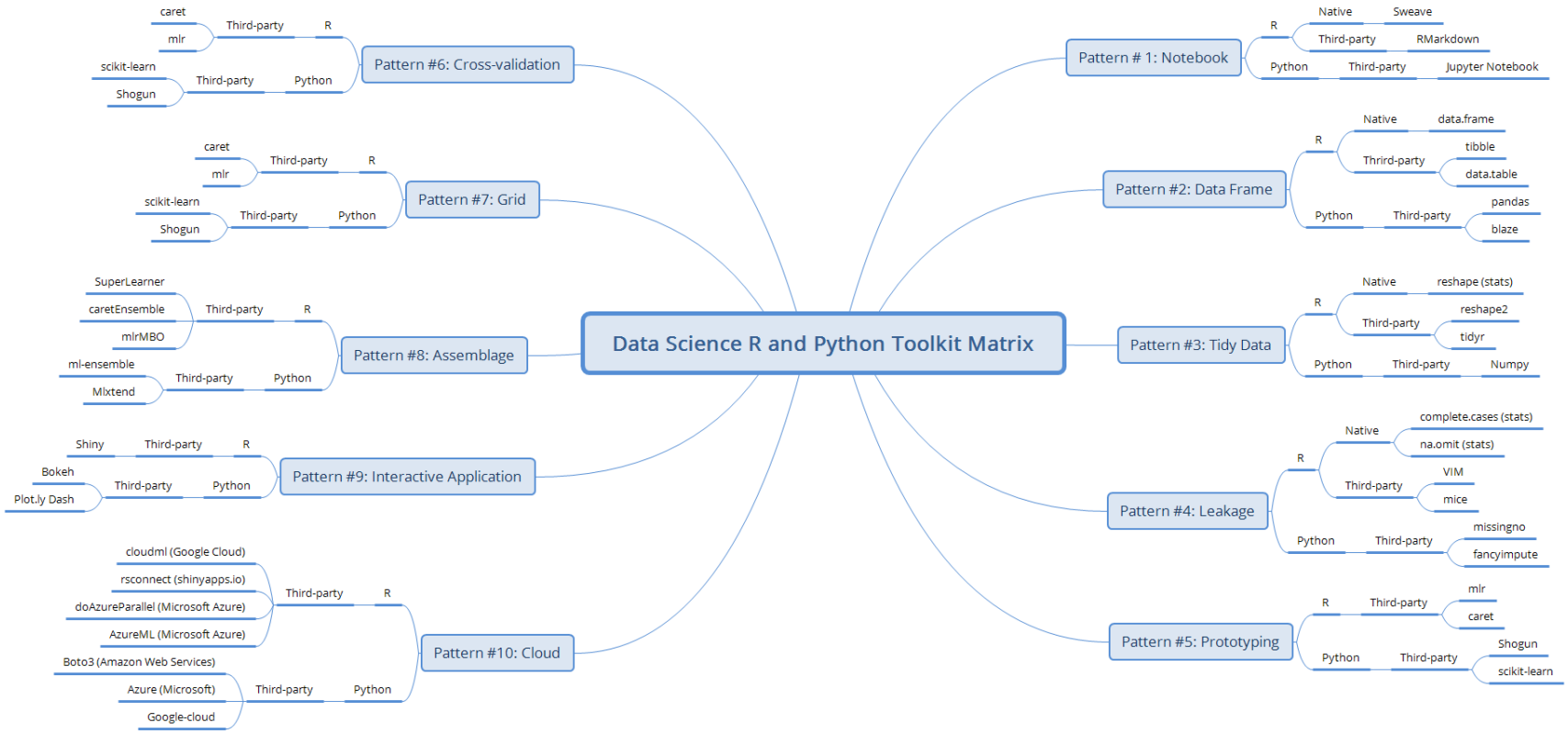


Figure 13: Illustrates Data Science R and Python Toolkit Matrix that consists of thirty-two packages and where applicable also with (R) native approaches that were incorporated into the language itself.

CHAPTER V

Reflection

To round up, this chapter discusses results of the endeavour taking into consideration various threats and limitations that arose during the examination. Moreover, with a perspective on future research, implications for scientists and practitioners are stated too.

5.1 Discussion of Results

The research questions in this thesis led to formalizing ten data science design pattern candidates that were supplemented with source code examples and subsequently the identified packages were presented in the Data Science R and Python Toolkit Matrix. In the following, a closer look is taken on both outcomes for the purposes of a discussion.

5.1.1 Design Patterns

Pattern Language

While each pattern was described relatively independently, as Fowler (2002, p. 10) has remarked, they should not be viewed "*isolated from each other*". Accordingly and ideally, they form a basis for a *pattern language* which helps stakeholders to utilize better DS solutions and give programmers and other engineers the knowledge to "*design complex systems*" and processes (Bafandeh Mayvan et al. 2017, p. 15). Specifically, ten patterns aid in creating a toolbox that data scientists not only need to be aware of but may actually actively use in their daily work when trying to make sense of data and produce conclusions for firm's target audience.

Through pattern's backward and forward references, their associated network was shaped and visualized in Figure 14 providing a "big picture" on a spectrum of related patterns targeting the same field (Dearden and Finlay 2006). Correspondingly, what interconnects the examined group is the analytical engineering which is applied on data that are potentially of high value to the organization (Grolemund and Wickham 2018; Provost and Fawcett 2013b; Ayankoya et al. 2014). The information from literature sources has likewise indicated that while DS patterns are likely unique due to conceivably useful (big) data, they are in many aspects similar to the

ones described in the traditional software engineering (Gamma et al. 1994; Sculley et al. 2015). Essentially, both are dealing with the source code and sensible development practices for design and architecture of computer programs that perform specific tasks. However, what depicted patterns in this thesis have additionally focused on is the process by which data are managed, prepared and utilized for extracting a wisdom as such understanding forms an integral part of DS.

In this study, the goal was not to develop a complete pattern language because of a very ambitious scope that would be necessary to consider. In fact, going back to the seminal publication of Gamma et al. (1994, p. 394), authors have argued that there will be hardly "*ever (…) a complete pattern language*". Therefore, theirs as well as the one of this work is "*just a collection*" and a family of accompanying problem and solutions pairs – a catalogue aiming to better understand the science of gaining insights (Gamma et al. 1994, p. 394; Schmidt et al. 1996).

Indeed, the completeness is influenced by several criteria, including the target audience which in this case were primarily the aspiring data scientists. Especially for them, the number of introduced patterns may be substantially larger when compared with already experienced professionals. Therefore, in order to develop a complete design pattern language, it would need to significantly expand the breath, aiming to cover the whole DS life-cycle – from business understanding to the final deployment as described by the CRISP-DM framework in Figure A.19.

Allied Patterns

As mentioned previously, the relationships between patterns are visualized in Figure 14 and at the epicentre of ten design patterns are two fundamental DS elements: *data frames* which are used for storing and processing information and ML *prototypes* which enable to acquire knowledge by applying supervised and unsupervised learning methods. Besides those examined in the **chapter IV**, additional four present other plausible and related candidates. One instance is *Data Robbery* where facts are acquired in an unauthorized way, for example through data scrapping. Alternatively, DS repository has to be not only searchable, well organized and sufficiently documented but also preferably under the version control system allowing to record project's history and development. However, in accordance with forty-one literature sources, due to not considering them as the most significant in the DS domain, they were omitted in this work's deliberation as it has been limited by the scope. Although only a set of ten patterns was formalized, the links between that describe a relationship such as "enhances" or "visualized in" make it possible to call them forming an incomplete *pattern language* in the DS context (Dearden and Finlay 2006).

Patterns Individually

Going into more detail, as it was suggested by Meszaros and Doble (1997), a pair of a

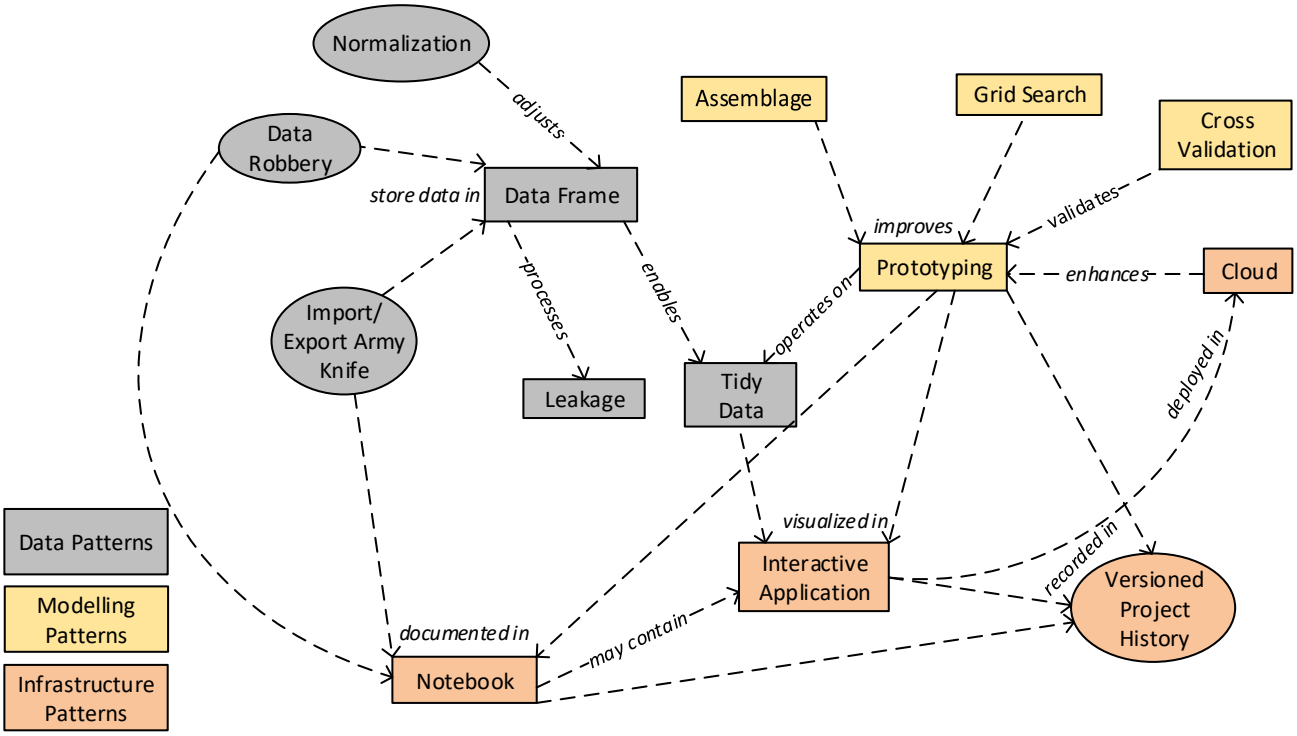


Figure 14: Illustrates a relationship between ten DS design patterns categorized into three distinctly coloured layers. Four other plausible patterns in an ellipse were not examined in this work.

Figure 15: Illustrates summaries of ten design patterns, see `Master_DS_DP.xlsx`.

Pattern Name	Problem	Solution
1. Notebook	How to document every step that is conducted in the DS life-cycle, including describing dead paths or how figures have been created?	*Notebook* -style applications combine a simple markup language with isolated source code snippets, both of which can be rendered in the same document.
2. Data Frame	How to programmatically store, manipulate and process tabular data?	Choose an in-memory *data frame* structure that acts as fundamental piece for all DS steps being conducted on data.
3. Tidy Data	How to organize messy information to facilitate its analysis in the DS?	A concept of *tidy data* prepares information by transforming it into the *long* format, thus programmatically enhancing its additional exploration.
4. Leakage	How to reliably complete missing data and have them ready for ML prototyping?	A model-based method named *multiple imputation* predicts missing data using a statistical model several times and combines such results into a single point estimate.
5. Prototyping	How to gain valuable and actionable insights into business operations using ML algorithms?	Experiment with designing several ML *prototypes* which deliver on stakeholders' goals through supervised and unsupervised learning methods.
6. Cross-validation	How to assess model's predictive power with an objective of potentially minimizing overfitting?	The family of *cross-validation* techniques divide data into multiple equally sized folds allowing model building on training sets and evaluation on the testing fold.
7. Grid	How to select best performing hyperparameters in an automatic fashion?	When hyperparameters are specified in a *grid* of matrix form, the corresponding algorithm can exhaustively *search* for the best combinations that should be applied.
8. Assemblage	How can diverse ML models be combined in a way that better predictions are achieved?	By using for instance an *ensemble method* called *stacking* , a meta-learner can aggregate results of individual estimators.
9. Interactive Application	How to prepare visualizations for audiences allowing them to further adjust presented content reflecting their own interests?	Designing *interactive applications* permits users to drill-down the data while having graphics updated automatically.
10. Cloud	How to deploy DS solutions across the whole company and beyond and enhance them using third-party infrastructure?	The design of IT architecture can take advantage of *cloud computing* services which offer various on-demand capabilities that are not limited to DS.

problem and corresponding solution was aimed to be *single-pass readable* and it was summarized in one sentence form in Figure 15. Even though patterns were numbered, they should not be looked at to be in a specific order of use when compared to the ones by Alexander et al. (1977). Typically, for instance, data scientists may opt to utilize *Notebook* at the very end of the modelling process when they are required to present their results and enlighten stakeholders about achieved predictions and insights.

The pattern language began first with a tool that permits data scientists to document their acquired knowledge using a consistent and simple format for description. An additional advantage of *Notebook* is that when text and code snippets are executed, the input and the output of the journal is rendered in line, making it cohesive within one file. Therefore, not only putting a focus on the right information being conveyed to business leaders but allowing to document the whole DS as it has been pursued too.

The visualization and communication of outcomes is critical and developing *Interactive Applications* that tell engaging stories and let to dive deeper into the data by adjusting how graphics look like only supports managers in outlining a strategy of how to increase the value of the products and services offered. Subsequently, the *Cloud* comes on the stage when created applications need to be deployed and scaled up and down to possibly thousands of employees worldwide whereby leveraging the resources and other features of cloud vendors. Thus, avoiding managing own hardware infrastructure as it likely not a core competency of the company.

Continuing, at the core of the knowledge discovery are naturally data, and therefore it is necessary to store them efficiently so that they can be manipulated and various mainly statistical operations can be carried out. After accessing them from source systems, another pattern forms a foundation through creating a two-dimensional data structure in a tabular form. Consequently, the use of *Data Frames* enables to transform (*Tidy*) them with an objective to further simplify processing and their use within the targeted ecosystem of tools. Exemplary for *Prototyping* purposes, the outcome of which shall be impactful and reliable predictions for co-workers.

The *Leakage* is often consequential when modelling algorithms need to be applied because it can lead to conclusions which are statistically skewed and invalid, making them almost certainly not accurately reflecting the data at hand. In fact, missing data should be a prime source of a concern for any KDD professional and due to their impact, missingness needs to be dealt with early on during the information pre-processing. On the other hand, *Assemblage* and *Grid* leverage computing resources to optimize hyperparameters of each model as well as to create several ones and intelligently combine them to additionally improve the performance for stated aims and avoid overfitting where the ML prototype is not generalizable to the new input. Undoubtedly, to validate the power of such model, *Cross-Validation* is *the* technique to split data and calculating multiple predictions and test set errors on different data sets. As a consequence,

average them to provide final unbiased outlook on the prototype.

Going back to section 2.2.2.1 on the pattern related research, not surprisingly some of the design patterns identified in this thesis have been formalized and proposed by several researchers as well. Notably, the work of Heer and Agrawala (2006) is compelling as it describes for instance *Data Column* which is directly associated in this work to *Data Frames* by not only allowing to store data in a common format but permitting to administer various expressions on data themselves too – what the same authors have described in a separate pattern called *Expression*. Moreover, *Interactive Application* make it easy for users to dynamically change graphics, illustratively through filtering. On this matter, Heer and Agrawala (2006, p. 859) have described *Dynamic Query Binding* which automatically "*refine[s] a data view through direct manipulation*".

Categorization

Besides, it could have been observed that with an exception of *Notebook Design Pattern* all are mainly applicable to CRISP-DM stages of data understanding, preparation, modelling, evaluation and deployment. Unfortunately, due to a delimitation of providing R and Python code examples, it was largely avoided to examine patterns related to the business understanding even though a considerable amount of information was collected. Nonetheless, with regard to their grouping, patterns might be not only associated with specific CRISP-DM phases as illustrated in Figure A.24 but due to addressing different design "levels" they could be divided into three structural layers too (Dearden and Finlay 2006; Chen 2004):

(a) *Data Patterns*: *Data Frame*, *Tidy Data* and *Leakage*

(b) *Modelling Patterns*: *Prototyping*, *Cross-validation*, *Grid* and *Assemblage*

(c) *Infrastructure Patterns*: *Notebook*, *Interactive Application* and *Cloud*

Principally, *data patterns* consistently fetch, store and prepare analysis-ready information which is the primary object dealt with in DS. Next, *modelling patterns* take advantage of the previous group and enable to stepwise acquire better predictive models and with them previously not known wisdom. At last, *infrastructure patterns* support previous two categories by offering underlying tools that allow to deliver actionable insights, for example by developing on and deploying solutions to the cloud-based production systems.

As such, the above-mentioned three classes form a cohesive pattern language, in the DS context, where its composition and interrelationships shall make it easier to derive, create and use best approaches to frequent hurdles in the knowledge discovery.

5.1.2 Toolkit Matrix

The third study question of this research has asked which open source tools are available in order to solve common pitfalls that each pattern has outlined and attempted to iron out. As

a result of this inquiry and the conducted qualitative survey where thirty-two packages were sampled, Data Science R and Python Toolkit Matrix was visualized in Figure 13. This likely represents a valuable resource for researchers and practitioners because it can support them with gaining an overview of some of the essential applications for specific DS tasks. Particularly, when they might not be aware of them due to being new to the domain and coming from other languages.

Sub-ecosystems

While **chapter II** has discussed R and Python software ecosystem from rather a high-level perspective, the outcomes presented through DSTM also provide an extended and deeper perspective into available tools. In fact, during the course of DS pattern discovery, it was observed that five packages are a part of two collections that form sub-ecosystems in both programming environments, namely the previously mentioned R's *tidyverse*[4] (`tibble` and `tidyr`) and Python's *SciPy*[5] (`pandas`, `Jupyter` and `Numpy`).

These sub-ecosystems, to which academicians have devoted so far only limited attention, provide utilities for DS with an objective to simplify importing, processing, modelling and visualizing raw information. At the same time, individual libraries available on CRAN, PyPI or currently in the development for instance on *GitHub* are strongly interlinked with each other by possessing typical ecosystem characteristics. Namely, in pursuing a common philosophy around API design and having a consistent implementation of internal components due to adopting specific conventions that allow to "*guess how another different component*" might work (Bryan and Wickham 2017, p. 785). Building upon interoperability within the respective universe of different packages sharing common objectives, group of developers and related infrastructure, two "*cohesive system[s]*" were created examined next (Bryan and Wickham 2017, p. 785).

SciPy

Because Python was developed as a general-purpose language and not being used for DS intentions alone, it has aimed to offer fundamental concepts for many plausible use cases, in diverse areas including web development. Therefore, the community of users had to build capabilities for DS from the ground up as they were not present in Python itself and hence SciPy ecosystem was set up by a reason of lacking an efficient array manipulation or plotting of graphics (Millman and Aivazis 2011). Subsequently, due to the rise of its popularity, over the years it has become a gateway to "*open-source software for mathematics, science, and engineering*" – the scientific computing – for which it has developed a comprehensive knowledge discovery toolkit (Perez et al. 2011; Oliphant 2007). On the other hand, tidyverse came from a different perspective as the community around R – given its a priory focus on the analytical domain – already had a blueprint on top of which it could build the improvements.

Tidyverse

As mentioned previously in section 2.1, R has been in the development since 1993 and even

[4]https://www.tidyverse.org/ – Tidyverse
[5]https://www.scipy.org/ – SciPy

earlier when considering its predecessor S. Ever since it has tried to be correct first, fast second and maintain backwards compatibility with tools being created back in the 1980s and 1990s (Maechler 2018). Resulting from this, however, is that many of the past choices in terms of what would a developer expect from a specific function may seem nowadays rather cumbersome to handle – likewise called "*historical inefficiencies and idiosyncrasies*" by Ross et al. (2017). A typical example is R's native `data.frame()` where by default any strings are silently converted to data type factor, which represents categorical variables, unless provided with `stringsAsFactors = FALSE` parameter (Peng 2015). While it may be a desired behaviour in many contexts, in others it causes a confusion, specifically for newcomers judged by a number of questions on websites like *Stack Overflow* (Mount 2014). When compared to Python, this overcame similar challenges by introducing a completely new version 3.x as described in the **chapter II**.

For R, where the innovation happening these days mostly in the package universe, tidyverse reimagines its already existing features in a new way by providing a consistent approach in "*function syntax[, its arguments,] inputs and outputs*", thus "*allow[ing] data analyses to flow naturally from one task to the next*", for instance through the "%>%" pipe operator while benefiting from data being in the *Tidy* format (Ross et al. 2017, pp. 2-3). By contributing feature rich improvements that cannot be easily incorporated into the languages itself by reason of its heritage, it has been often taught from the beginning in the courses of statistics and DS due to supporting "*entire end-to-end workflow*" (McConville 2017; Rickert 2017).

Going back to the DSTM, together with two sub-ecosystems, it has shown that users can clearly take advantages of such (and not limited to) participated packages because they are extensively documented and supported by a large and strong community of developers around them. Hence, helping to overcome any potential challenges faced by the users and at the same time ensuring that there is a seamless integration and compatibility beyond the both sub-ecosystems.

In general, three remarks could be made. For one, some packages like `pandas` and `mlr` are in fact universal and can be used for multiple DS design patterns, see Figure A.24 where it was attempted to link and categorize libraries with CRISP-DM framework and its six phases. Indeed, these tools are not only suited of being used with their primarily associations (*Data Frame* and *Prototyping*) but with other relevant ones as well, including *Tidy Data*, *Grid* or *Cross-validation*. As a result, beginners and experienced professionals should seek such comprehensive applications as they cover a wide range of tasks and permit to avoid jumping between different libraries that may not interact with each other in a simple manner.

Moreover, even though R's package ecosystem is approximately a tenth the size of Python's, it has shown to provide a richer set of DS-related capabilities for diverse KDD steps – especially when omitting *Usage Metrics* and *Maintenance* criteria explained in Table B.7. Nonetheless, both programming languages offer a large collection of DS utilities, with and without previously

mentioned two sub-ecosystems. Given that Python's core developers have aimed to build a language that is applicable in a broader IT landscape rather than just focusing on one particular area such as data analytics, *R Core Team* could independently pursue a path of developing an open source alternative to *Matlab* or *Wolfram Language* that supports primarily mathematicians. Thus, based on the gathered observations in the course of this thesis, the discussion of choosing between R and Python proves to be inappropriate as clearly R is more advanced for the data analysis, though it also offers little flexibility outside of its domain compared to Python and its ecosystem. Accordingly, the need to learn both of them, one after another, as explained by Theuwissen (2015, 2016).

Finally, while recording tools in the database and creating DSTM, the significance of *GitHub* was manifested as being *the* platform for the development and programmers' collaboration. Albeit arguably more feature rich alternatives like *GitLab* or *BitBucket* exist, all sampled packages have source code repository there and with a few exceptions make use (though to various degrees) of continues integration and deployment whereby software applications are automatically tested, build and released – in line with Agile practises (Biňas 2013; Krill 2016; Stolberg 2009). As such, data scientists may use *GitHub* not only for monitoring team's DS progress, developing new packages and sharing publicly the result of data analysis with the source code but when discovering new projects currently in the development too.

5.2 Limitations and Implications

Repercussions

The findings presented in this thesis add to the overall understanding of design patterns within the DS field – specifically where and how they can be applied to help distinct groups of stakeholders simplifying knowledge extraction from the diversity of data compiled. The implications of this examination are important for both researchers and practitioners.

Scientists

Firstly, the contribution of this study was by providing a systematic review of extant research done in the fields of SECO, DS and design pattern – all of which were put into a relationship. This served as a basis for a subsequent analysis and where missing elements were also identified such as very little studies dealing with Python software ecosystem and where more in depth investigations can be worthwhile. Additionally, the background research presented a solid evidence base for future inquiries as design patterns for DS were not yet constructed and to what this study has finally devoted an attention, attempted to fill the gap and added to the body of research by extending the knowledge of design patterns which may be applied in a new interdisciplinary domain. All this is particularly relevant for the community of scientists who might deliberate and discuss these ten design patterns and use deliverables to formulate new ones based on the already conducted exploration (Stol and Ali Babar 2010).

Professionals

Although any outcomes should be interpreted with caution, they give additional insights to various groups of stakeholders working directly with DS teams as well as aspiring data scientists themselves. For the target audience of this thesis, the formalized patterns help to better understand what best practises could be leveraged when conducting DS as they relate to problems that often come up and need to be tackled. Hence, pattern candidates shall educate and train for possible situations including in the modelling phase and case of overfitting and accordingly provide ideas and steps that may appropriately address them to minimize any negative impact. For experienced professionals, this work can further strengthen their expertise by (re)discovering practises and applications that they have not used so far, and thus equip them better for future occurrences and needs too.

Even though executives and UI/UX designers might not be interested in technical details of data mining and KDD, they can use these outcomes to become acquainted with some core principles behind the science of data as well. In fact, non-technically skilled managers should benefit from this information while navigating their organizations towards greater DS adoption and utilization. As a result, they can prepare for potential challenges which it brings, for instance in terms of proper project management and data governance.

Previously, it was mentioned that R and Python have two large sub-ecosystems of tools that provide well-established capabilities for DS. Through explored applications that support each pattern candidate, DSTM has the potential to give valuable guidelines for new (data science-oriented) programming languages such as *Julia* too. This in understanding of what kind of tasks it needs to be able to address, for example by offering advanced modelling capabilities within a modular API, in order to establish itself as *the* language for KDD purposes (Krishnakumar 2011). Therefore, it helps a wide group of stakeholders, including vendors, individual developers, investors, and other contributors, to decide in which software ecosystem to invest their time, skills and other resources (Hoving et al. 2013)

Constrains

This study, by discovering data science design pattern candidates, has established a starting point on which future research can base its findings. Though, in the following, it is necessary to judge the conducted qualitative research as this thesis was subject to several conceptual and methodological weaknesses. When these limitations are explored and devoted a renewed attention by scientists, it could additionally help to increase pattern language's quality. Indeed, in every step as documented in Figure 3, unavoidable subjective choices were made which resulted into bringing a source of uncertainty on the outcomes. To compensate for taken decisions, it was attempted to explain the situation and provide a broader perspective on the matter (Petrov 2016).

Golafshani (2003) has documented that depending on the type of study, authors had diverse views on what specific criteria should be of a concern in order to determine the quality

of the research (Long and Johnson 2000). Some scientists have claimed that credibility, trustworthiness and rigour are suitable lenses and of a paramount importance in the qualitative and quantitative research alike – all of which stem from the universally accepted terms of *reliability* and *validity* (Golafshani 2003; Long and Johnson 2000). With regard to the former one, because of a difficulty in demonstrating it, a revision has been made to capture dependability and consistency (Golafshani 2003). On the other hand, validity has been explained as to whether the study "*represents accurately those features (…) that it is intended to describe (…) or theorize*" (Long and Johnson 2000, p. 31). Coming next, some of the issues divided into three major categories are discussed.

Construct and Internal Validity While construct validity deals with threats that "*might arise during research design*", the internal validity investigates problems "*during [the subsequent] data extraction*" (Ampatzoglou et al. 2013, p. 13).

As noted by many researchers, being limited to the availability of data from which one may derive design patterns, the quality of selected literature sources is critically important. Overall, the work has shown to be able to derive ten candidates from a set of literature sources as opposed to directly for example interviewing domain experts. Although more qualitative and quantitative data could have been gathered, the outcomes might not have changed much as ultimately a decision was taken to formalize only an initial subset of ten design patterns. On the one hand, the advantage of the approach described in section 3.1.1.1 was that fewer restrictions and assumptions were placed on the research and on any data collected. Hence, allowing to assemble a reasonable sample according to the best efforts (Krishnakumar 2011; Coyne 1997). At the same time, the study benefited from not only searching manually but also utilizing snowballing technique where additional resources were collected in order to support patterns' reliability. On the other hand, the process is not fully reproducible and it could be argued that because it depended on information sources some of which were later deemed unreliable, it similarly lacks further power to be representative of the field too (Hajimia 2014).

While borrowing principles of SLR, the search has been initially done through key words that were deemed appropriate by author's best judgement. Yet, due to a general paucity of the relevant primary and secondary data, the search was extended on several occasions by incorporating various new terms such as "lessons learned" to raise the quality and volume of gathered literature. Later, the GIA was applied to extract phrases and concepts that would lead to the discoverability of DS pattern candidates. Though, it should be noted here that text labelling was not supported by an independent parallel review of the same documents by other researchers as otherwise it would gain an additional trustworthiness by possibly clarifying misinterpretations and conducting "*peer debriefings and stakeholder checks*" (Thomas 2006, p. 243). To increase

such validity, a cautionary opinion of a fellow researcher could bring a novel angle to the most important design patterns in the DS that shall be elucidated.

With regard to the toolkit matrix, even though it was attempted to discover relevant tools through criteria listed in Table B.7, author's a prior knowledge and any R and Python experiences might have negatively influenced the choice of tools too. This both in terms of their volume and relevancy, resulting into introducing selection and other cognitive biases into this work. As a consequence, unintentionally omitting applications that are of a great significance for the particular problem.

Transferability Likewise called *external validity*, transferability deals with "*generalizing the obtained results from the sample to the population*" (Ampatzoglou et al. 2013, p. 13).

Naturally, patterns shall be neutral and easily applicable to other related areas, environments or technical applications. In the context of this research, it could be reasonably stated that described design patterns are not specific to the DS per se because of their broad nature that covers a wide domain of business analytics and its engineering practises which are underpinned by the KDD process. Not surprisingly, given that DS was defined as an umbrella term that covers a multitude of concepts, one may also find illustrated pairs of a problem and solution to exist in areas like ML (*Grid* or *Cross-Validation*) or big data (*Cloud*). As stated previously, *data frames* are typically used for a variety of use cases across the fields.

Going back to Gamma et al. (1994), authors have described design patterns with C++ examples. However, the notion of particular issue and answer to it can and has been applied in other languages, exemplary Java or Ruby as well (Freeman et al. 2009; Olsen 2007). Thus, patterns in this research might have been supplemented with other programming languages, *Matlab* to name one example. Nonetheless, usually with modifications as presented artefacts in a predefined form are just templates which have to be adjusted due to a variety of faced situations or technologies (Fowler 2002).

Confirmability As mentioned in section 3.1.1.3, not only candidates were outlined based on the data from the gathered literature but through additional search of their practical application it was ensured that they are indeed being used at various stages of KDD life-cycle and by data scientists themselves. Yet despite following 3D2P methodology, a triangulation of multiple methods for data collection, analysis and evaluation could potentially derive better findings and improve their confirmability.

Indeed, further research might repeat this work by meaningfully integrating other approaches used for pattern discovery. Along those listed by Inventado and Scupelli (2015), the knowledge of domain experts shared through in-depth interviews or writing workshops could plausibly

Domain Experts

prove to be the most relevant and valuable to have. Hence, increasing the obtained wisdom both in the volume as well as mainly in its quality – stemming from direct involvement of experienced professionals who were not, unfortunately, surveyed in the study. Accordingly, help these design pattern candidates to acquire a more confidential status in order to spread farther their use within the community of data scientists and beyond.

5.2.1 Future Research

In the future studies, among others, two notable paths can be pursued to establish a more complete DS pattern language.

Different Approach

As it was noted, a research limitation of this study is utilizing literature works rather than interviewing experts from the domain who could likely outline problems and solutions that they apply in a more reliable fashion. Consequently, based already on the conducted research, the next step would be to try to empirically confirm findings and thus enhance the quality of identified design patterns for example though a quantitative survey of DS experts from diverse industries and countries. Furthermore, if a choice had been made to use mixed methods methodology, expert interviews could have been combined with a literature review leading to supposedly further increasing the value of formalized design patterns.

Different Perspectives

Moreover, being limited by practical constraints, the thesis could not provide a comprehensive review of the whole DS landscape. This is a reason a decision was taken to describe only a number of pattern candidates linked to the domain. However, during the stages of *pattern prospecting* and *pattern mining* large amounts of various information were gathered and this may be used to uncover and formalize new problem and solution pairs. In fact, it was observed that some plausible patterns are related not only to *technical* aspects (and on which this study has tried to zoom in) but also to the *organizations* or the *environment* – in line with TOE framework, firm's contexts and forces influencing the adoption of any kind of technology (Baker 2012; DePietro et al. 1990; Oliveira and Martins 2011). As a result, additional pattern examples ought to include themes like how to acquire stakeholders' support for DS initiatives, how to tackle right analytical problems in right way that encourages their involvement and how to convert and educate business analysts to become data scientists (Halper 2016; Wujek et al. 2016; Shan et al. 2015). Therefore, extra research shall examine more closely collected data, seen in Figure A.22, from different perspectives and attempt to discover other design patterns when considering the whole life-cycle of DS.

Entanglement

At the high-level of perspective, data science design pattern may ease the complexity of interaction between *technical* systems (like R and Python which simplify conducting specific tasks) and *social* ones where humans participate in some organizational structure and use the technology to accomplish their goals (Orlikowski and Scott 2008; Whitworth 2009). At the

core of this *socio-technical system* theory is the idea that a process such as data analysis has to take into account not only technical factors that influence the functionality, usage and a wisdom gained from applying DS but a whole set of interdependent social aspects too which cannot be treated as separate parts in a complex system (Baxter and Sommerville 2011). However, in this thesis the focus was only on the technological domain that cannot be viewed independently due to complex linkages and mutual interactions between design patterns themselves and a developer who has to shape and adjust these templates (Fowler 2002). Thus, in the future research it could be interesting for example to formulate DS design patterns as seen from the social angle and considering expanding the above-mentioned TOE framework with a socially based individual dimension (Petrov 2016). Exemplary, make an inquiry of how identified patterns are applied differently, though being in similar circumstances. The reason for this might be the impact stemming from various organizational and individual aspects, from the cultural influence to Hofstede's power distance between employees (Hofstede 1993, 2011).

Multidimensionality

Besides, going back to section on delimitations of this work, further studies extending design pattern by presenting corresponding anti-patterns would be very interesting as well. Additionally, with recent advancements in the artificial intelligence it is clear that one will need to explore applications that support multidimensional data and associated patterns which this thesis has intentionally omitted when it was focusing only on the tabular ones. Illustratively, the work of Perez (2018) already attempts to put an attention on the concept of *deep learning* and design patterns that can be practically applied in the robotics or self-driving vehicles.

To summarize, what has been described in the **previous chapter** helps data scientists to become better equipped with technical and theoretical knowledge of processing data stored in various quality and granularity when producing business relevant, actionable results to themselves and other internal and external co-workers. At the same time, novel studies in the area of design patterns for analytical purposes shall not only use gathered data in Figure A.22 to develop new candidates but also try to qualitatively validate those described in this thesis for instance though expert interviews. Indeed, when this and other supplementary research is pursued and conducted, it can foster effective management of company's expectations in relation to the broader and inseparable "*technology, techniques, and people*" (Bhatt 2001, p. 68).

5.3 Conclusion

To conclude, given the scarcity of information in the current literature regarding *data science design patterns*, the quest of this work was to address this lack by exploring and describing their first, known to the author, systematic account. As a result, the inquiry has tackled frequently arising problems and their best solutions in the interdisciplinary field of DS. Essentially, the

goal was to qualitatively note a "*common solution, look for its core, and then write down the resulting pattern*" (Fowler 2002, p. 10; Schmidt et al. 1996).

Focusing on DS design patterns, a devised research gap led to formulating the main objective of uncovering solutions to identified issues and secondary aim of supplementing drafted artefacts with practical examples using utilities from R and Python ecosystem. Subsequently, this translated into three study questions, the first of which was to understand key terms.

As such, the work began with an exploration of software ecosystems of R and Python. Characterized by different actors, though appearing and collaborating as a group on the same technological platform, these were later surveyed in order to identify libraries and provide source code examples to each design pattern. Certainly, both environments offer a wide range of applications supporting researchers and practitioners in analysing data in different disciplines, with R having an edge by reason of its rather domain-specific focus. While two programming languages are also distinct, they share a large, often interacting, community of users and developers who participate in their common aspirations – be it for advancing the analysis and visualization of data or for development of web applications and system-level scripting.

Although DS may not have been a novelty because of absorbing other existing concepts like ML and big data, section 2.2 explained that it has become associated with loosely defined analytics where not only sound mathematical foundations are necessary but software engineering experience too. By cause of a diversity of conducted tasks, data scientists have been needed who combine a multitude of critical skills in their work such as the proficiency in statistics, deep domain know-how and among others the ability of programming with R and Python. Even though the field may have roots in the academia and research where different information has always been analysed, skilled data scientists are these days difficult to identify and come by. Therefore, for example, if business analysts want to advance their career and move into more technical field, the presented DS design patterns may benefit them by gaining an initial understanding of best practises that can solve common issues when acquiring insightful knowledge.

After key phrases were reviewed in detail, a research methodology was presented in the **chapter III** incorporating qualitative methods under the stewardship of the *data-driven design pattern production* framework. By using a systematic GIA for gathered data, major themes were mined from a final set of forty-one literature sources – all that in order to discover and formulate yet-to-be-verified design pattern candidates. As a result, based on the understandings of three concepts, the illustrated methodology helped to systematically accumulate data, analyse them and codify these patterns in a simple to understand template form. Unfortunately, while many potential candidates could have been found dealing for instance with data normalization, it was beyond the scope of this work to describe them all in a complete language. Not coincidently where more research remains to be pursued as there is no silver bullet to work out each

encountered problem. Hence, only a set of ten pieces was elucidated together with gaining a larger picture into R and Python ecosystems and how they can solve arising DS issues.

Overall, the results of this thesis should help professionals in the DS and associated fields to more easily document and share their wisdom and best practises through used vocabulary and benefiting from a specific form that was applied for their formalization. As discussed earlier, the artefacts could be categorized into three groups according to the primary subject matter being dealt with, namely data, ML prototype or infrastructure. Therefore, this classification shall further aid in decomposing DS problems into manageable parts and offer a matching tool for avoiding "*wasting time and resources reinventing the wheel*" both in terms of creating new duplicate tools as well as questionable, yet-to-be-verified approaches to a particular DS problem (Provost and Fawcett 2013b, p. 20).

When equipping ten design patterns in the **chapter IV** with a collected sample of thirty-two R and Python packages, some libraries were used for displaying source code examples too. In this manner, supporting target audience in showing a reliable set of tools capable of addressing outlined issues that occur when following a process methodology such as CRISP-DM. Within the sample, five tools were also identified stemming from two R and Python sub-ecosystems, named tidyverse and SciPy. Documented in **this chapter**, these aim at providing a complete end-to-end toolbox that might be beneficial for any emergent DS purpose and use cases, from managing (big) data using in-memory data frames to building predictive models that power business forecasts through applying comprehensive, well-known ML libraries.

The collected utilities from two (sub-)ecosystems for each design pattern permit to answer the third study question whereby creating Data Science R and Python Toolkit Matrix. From this map, undoubtedly, newcomers to DS can gain a starting point into R and Python environments by reason of visualizing some of the fundamental, arguably "most useful and popular", tools like `caret`, `pandas` or `Shiny`. Subsequently, it can lead researchers and practitioners to utilize most appropriate applications for a particular situation that is being faced in their work.

Indeed, designing IT architecture for analysing data requires significant human efforts whereby mistakes in the process are learned with the experience over a period of time. In order to avoid pitfalls, the knowledge of dedicated design patterns examined in this thesis, coupled with others like big data or cloud computing, can greatly simplify employees' working intentions and help to take educated decisions right from the beginning. Nonetheless, Fowler (2002, p. 13) writes that one should "*never forget that [they are] a starting point, not a final destination*". Even though design patterns give a common ground to "*communicate, document, and explore design alternatives*", it is always necessary to see them in the context of specific work assignments, usually within a constrained business structure and its available technology and resources (Gamma et al. 1994, p. 389). Thus, when tackling DS challenges, due to internal

systems and processes, data complexity or surrounding organizational environment, ten design patterns will nearly always require further adjustments to a particular case, applications used and possibly developer experiences and preferences too. As such, pattern candidates need to be thoroughly understood by practitioners before being applied. Otherwise there is a risk of running into a situation where methods and solutions are exercised without fully realizing why they are needed, what is their impact and how they contribute to the acquisition of the wisdom in the first place.

Overall, the described design patterns contribute in better understanding of how they can address and solve, together with source code examples that employ tools from R and Python software ecosystem, ten frequently arising DS problems. Almost certainly more research will be pursued, for instance by Morley (2019), however, as the outlined aims from the introduction were achieved, a conclusion is made here.

APPENDICES

APPENDIX A

Figures

Review of Software Ecosystems related to programming languages

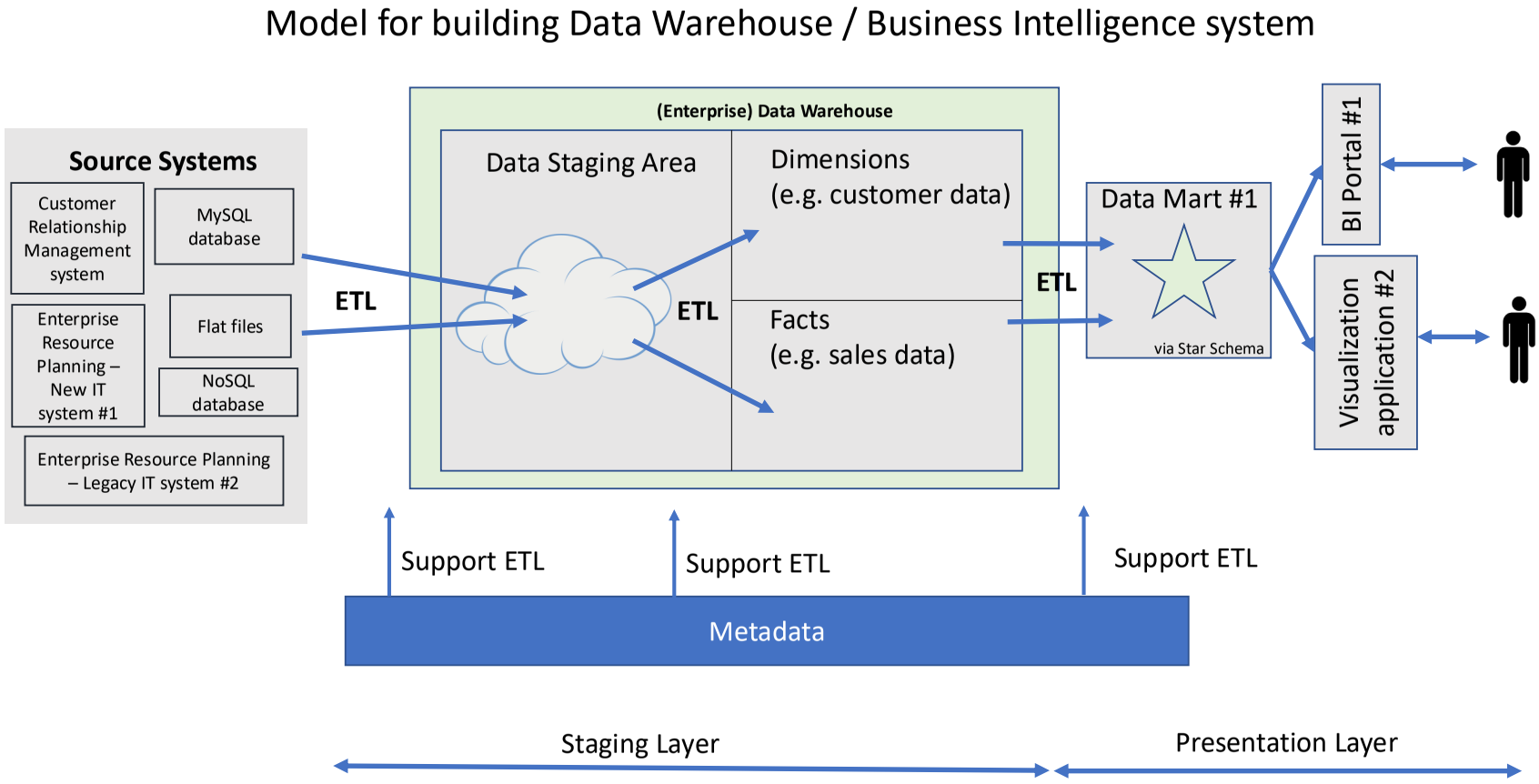
Article Name	Citations from Google Scholar (GS) / other	Authors	Reviewed Ecosystem	Description	Methodology
The Evolution of the R Software Ecosystem	GS: 52 / IEEE: 17	German et al. (2013)	R	Explore evolution characteristics of R's ecosystem through CRAN packages and mailing lists	Quantitative analysis of different sets of packages - Base, Recommended, Popular and Contributed ones
How do software ecosystems evolve? A quantitative assessment of the R ecosystem.	GS: 1	Plakidas et al. (2016)	R	Examine structure and evolution of R ecosystem to answer what makes it successful	Quantitative assessment of marketplace and community metrics are used to understand its state and health (data from CRAN and Bioconductor)
Evolution of the R software ecosystem: metrics, relationships, and their impact on qualities	GS, Scopus: 0	Plakidas et al. (2017)	R	Explorative analysis of R ecosystem which aims to generate a variety of metrics for an ecosystem-wide assessment	Mixed; examine evolution of dependency network and analyse R's ecosystem health (data from CRAN and Bioconductor)
On Clusters in Open Source Ecosystems	GS: 18	Syed and Jansen (2013)	Ruby	Explorative study where the goal is to find relationship characteristics within Ruby sub-communities	Mixed; use concept of modularity (from social network analysis) to expose clusters in ecosystem. Later, conduct an online survey to gain qualitative insights into clusters of developers
Steering Insight: An exploration of the Ruby Software Ecosystem	GS: 42 / Springer: 3	Kabbedijk and Jansen (2015)	Ruby	Present roles, elements, characteristics and relationships in the Ruby ecosystem	Quantitative analysis of social network data (from GitHub)
Python: Characteristics Identification of a Free Open Source Software Ecosystem	GS: 9 / Scopus: 6	Hoving et al. (2013)	Python	The goal is to identify which characteristics can be found in FOSS ecosystem. For this, authors decide to investigate Python ecosystem	Quantitative analysis of data from PyPI
An Empirical Comparison of Developer Retention in the RubyGems and npm Software Ecosystems	GS, Springer: 0	Constantinou and Mens (2017)	JavaScript (NPM) and Ruby	Compare RubyGems and npm ecosystems to analyse socio-technical activities	Quantitative; use survival analysis to identify factors which lead developers to abandon the project (GHTorrent data)
Others, not mentioned in section 2.1.3.1					
Python: an ecosystem for scientific computing	GS: 121	Perez et al. (2011)	Python	Authors narratively introduce Python and its ecosystem for scientific computing (incl. some of its libraries) and argue why it is advantageous compared to Matlab or C++	
Socio-Technical Congruence in the Ruby Ecosystem	GS: 9	Syeed et al. (2014)	Ruby	Ruby gems are investigated to verify whether they maintain high socio-technical congruence levels	Empirical study of Conway's Law
BioRuby: bioinformatics software for the Ruby programming language	GS: 149	Goto et al. (2010)	Ruby	Describe BioRuby software components that provide tools for bioinformatics written in Ruby	
On the Maintainability of CRAN Packages	GS: 24 / IEEE: 7	Claes et al. (2014)	R	Analyse packages and their dependencies in order to identify sources of errors and how much time is needed to fix them	Quantitative; conduct study per various flavours, i.e. devel-linux, release-macos, etc. (data from CRAN)

Figure A.16: Illustrates a literature database that is used in section 2.1.3.1 to provide a review of corresponding articles where surveys of different programming ecosystems have been conducted, see `Master_DS_DP.xlsx`.

A review of surveys of tools

Article Name	Citations from Google Scholar (GS) / other	Authors	Domain	Description	Methodology (search/selection/assessment)	# of Tools surveyed
A survey of tools for the analysis of quantitative PCR (qPCR) data	GS: 49 / Scopus: 22	Pabinger (2014) et al.	Biology	Survey FOSS applications for analysis of quantitative polymerase-chain-reaction (qPCR) data	According to 10 criteria such as whether tool supports error propagation leading to assigning "+" (yes) or "--" (no)	27 (9 of which are R-based and 1 Python-based)
A survey of current software for network analysis in molecular biology	GS: 115 / BioMed: 73	Thomas and Bonchev (2010)	Biology	Survey software for network analysis and conduct tests to compare features and capabilities reflecting real-life scenarios	Qualitatively assess two scenarios	1 FOSS (Cytoscape) and 2 commercial applications (Pathway Studio, IPA)
A survey of visualization tools for biological network analysis	GS: 150	Pavlopoulos et al. (2008)	Biology	Survey visualization tools for biological networks	Selected to cover wide range of functionalities, e.g. for visualization of pathways, gene networks, etc.	5 FOSS and 4 proprietary tools analysed according to different criteria such as compatibility with different data, etc.
Data mining tools	GS: 108	Mikut and Reischl (2011)	Computer Science/Information Technology	Give a detailed overview of commercial and FOSS tools. First introduce 7 criteria for tools' comparison and later classify them into 9 categories	Acknowledge using Excel database of tools from C. Giraud-Carrier	195
A survey of data mining and knowledge discovery software tools	GS: 562	Goebel and Gruenwald (1999)	Computer Science/Information Technology	Survey commercial and FOSS off-the-shelf applications for KDD and data mining according to general and data mining characteristics, and database connectivity	Each application is analysed according to the developed classification scheme	43
A survey of open source tools for machine learning with big data in the Hadoop ecosystem	GS: 79 / Scopus: 35	Landset et al. (2015)	Computer Science/Information Technology	Discuss 3 processing paradigms from perspective of 5 ML engines that implement them and look at associated ML libraries	Qualitatively assign a rating to 4 frameworks based on literature and documentation review - not on experimental/quantitative assessments	5 (MapReduce, Spark, Flink, Storm, H2O) processing engines and 4 ML frameworks (Mahout, Mlib, H2O, Samoa)
Survey of Real-time Processing Systems for Big Data	GS: 46	Liu et al. (2014)	Computer Science/Information Technology	Present FOSS that supports real-time big data processing in relation to Hadoop	6 distinct characteristics and features are described and compared	10 (e.g. Spark, Kafka, Impala)
Cloud monitoring: A survey	GS: 368	Aceto et al. (2013)	Computer Science/Information Technology	Survey commercial and FOSS monitoring tools for cloud computing	Assign "Yes" or "No" to each tool according to 12 criteria such as accuracy, etc.	9 commercial and 8 FOSS platforms, 11 services
Others, not mentioned in 2.1.3.2						
Big Data technologies: A survey	GS: 2	Oussous et al. (2017)	Computer Science/Information Technology	Review technologies developed for big data for each system layer (e.g. data storage, processing, etc.). Focus on FOSS applications	Descriptive in nature, compare different properties of selected tools (e.g. HDFS vs. HBase)	Apache's Hadoop family (HDFS, HBase, MapReduce, Pig, etc.), R is mentioned for performing advanced analytics
A survey on platforms for big data analytics	GS: 124	Singh and Reddy (2014)	Computer Science/Information Technology	Surveys hardware platforms to assess them based on metrics such as scalability, etc. Moreover, present a case study implementing k-means algorithm on them	Qualitatively assign "stars" according to different criteria such as fault tolerance and real-time processing	3 horizontal scaling platforms (Spark, MapReduce, TCP/IP) and 4 vertical scaling platforms (MapReduce, GPUs, FPGA, Multicore CPUs)
Big Data: A Survey	GS: 940	Mao et al. (2014)	Computer Science/Information Technology	Provide background and big picture on big data. For data generation, acquisition, storage and analysis, discuss applicable technologies	Qualitatively describe different characteristics	Unspecified - many different applications are mentioned for each of 4 phases (R is "data mining and analysis tool")

Figure A.17: Illustrates a literature database that is used in section 2.1.3.2 to provide a summary of articles surveying different software applications, see `Master_DS_DP.xlsx`.



Cloud Icon – courtesy of Balin (2009) – public license - https://commons.wikimedia.org/wiki/File:Cartoon_cloud.svg
Person Icon – courtesy of Sperry (2016) – MIT license - http://ionicons.com/

Figure A.18: The figure draws upon Thornthwaite and Ginnebaugh (2012) and Kimball et al. (2008) and presents a typical architectural design of building data warehouse/business intelligence system, see `DW_BI_Structure.pdf`. Data are first extracted from the source systems such as (non-)relational databases or flat files (for instance comma-separated values documents) into the staging area. Then, they are transformed for building the *dimension* and *fact* tables. At last, they are loaded for example into data marts and OLAP cubes in order to present them in the diverse BI frond-end applications.

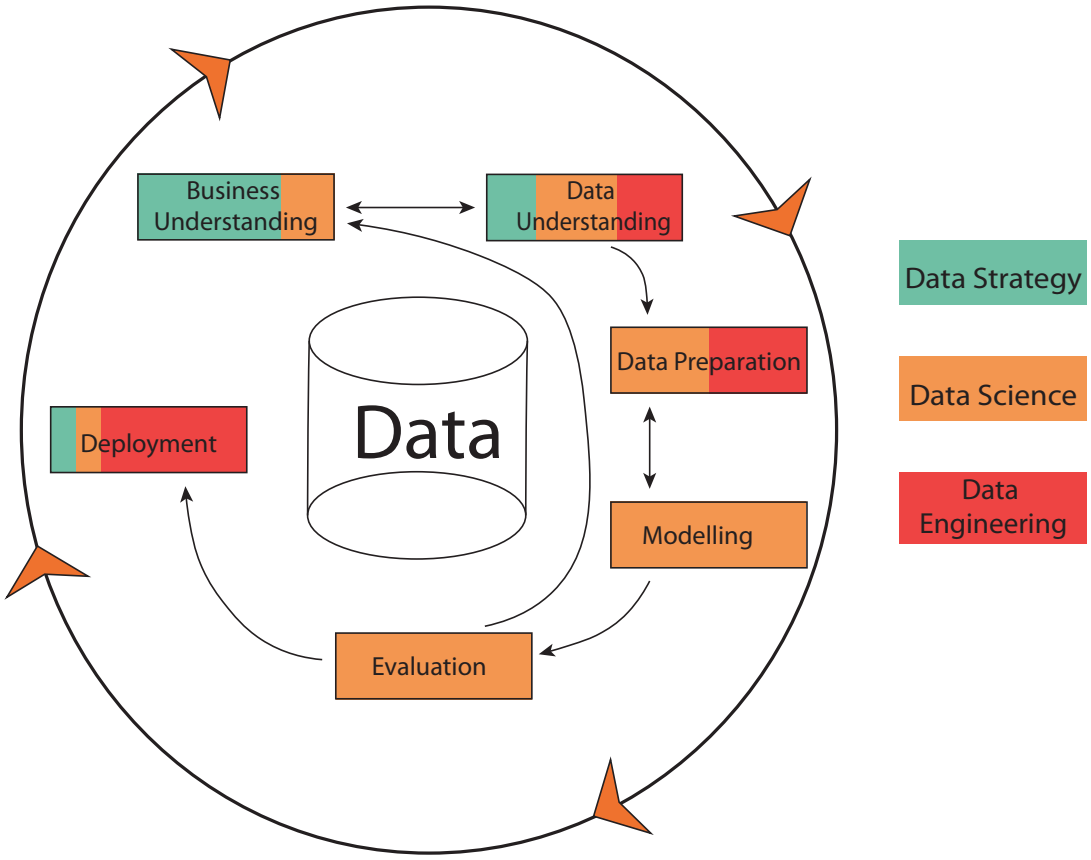


Figure A.19: Illustrates six iterative phases of CRISP-DM methodology and the prevalence of *data strategy*, *data science* and *data engineering*. All this with regard to each CRISP-DM stage where the portion of data strategy tries to bridge the gap between *what* and *why*. On the other hand, the part of data science combines business and technology skills for answering targeted questions, for instance through model building. Finally, data engineering works on the end-to-end analytical platform where data are acquired, later processed by means of ETL pipeline and presented to stakeholders. Inspired by Chapman et al. (2000) and Samani et al. (2016).

Data Science/Big Data studies with relation to design patterns

Article Name	Citations from Google Scholar (GS) / other	Authors	Type	Domain	Description	Pattern Usage (e.g. with source code, examples, practical applications)
Big data architecture and patterns - https://www.ibm.com/developerworks/analytics/library/bd-archpatterns1/index.html		Masore et al. (2013)	A series of web articles	Big Data - architecture	5-part series of articles details pattern-based approach to building big data architecture	
Mosaic Data Science Design Patterns - http://www.mosaicdatascience.com/resources/data-science-design-patterns/		Mosaic (2014)	A series of web articles	Data Science	5-part series of articles details typical DS tasks/problems e.g. handling missing values or their normalization	x
BigDataPatterns.org		Arcitura (Erl, Khattak and Buhler - 2015)	Website	Big Data	Describe IT artefacts (mechanisms; e.g. compression engine) and set of individual and compound patterns	
The Patterns Of Big Data	-	Hopkins et al. (2013)	Forrester Research Presentation (see related report "Deliver On Big Data Potential With A Hub-And-Spoke Architecture"	Big Data - architecture	Through 11 interviews, 4 big data patterns have been identified and described	x
MapReduce Design Patterns	GS: 112	Miner & Shook (2012)	Book	Hadoop/MapReduce paradigm	Present a set of 6 different sub-categories of patterns that can be used for MapReduce	x
Visual Thinking Design Patterns	GS: 2	Ware et al. (2013)	Journal Article	Visualization	Find and describe 20 visual thinking design patterns to support construction of efficient visualizations	
Software Design Patterns for Information Visualization	GS: 208	Heer and Agrawala (2006)	Journal Article	Visualization	Present 12 design patterns that have been observed in existing visualization frameworks	x
Towards Design Patterns for Dynamic Analytical Data Visualization	GS: 38	Chen (2004)	Journal Article	Visualization	9 visualization design patterns, organized according to Gamma et al. (1994), are presented and discussed	x
Systematic Literature Review / Systematic Mapping Study						
Research state of the art on GoF design patterns: A mapping study	GS: 44	Ampatzoglou et al. (2013)	Journal Article	General - focus on GoF's patterns	Provide an overview of research efforts on GoF's design patterns until the end of 2010. Categorize 120 research studies into 5 subtopics such as pattern's formalization or detection and report that patters can enhance one quality attribute in expense of another.	
The state of the art on design patterns: A systematic mapping of the literature	GS: 2	Bafandeh Mayvan et al. (2017)	Journal Article	General	Having a broader scope of inspecting 637 articles, identify and quantify 6 research topics between 1995 and 2015. Conclude that more research is needed for pattern evaluation and specification.	
Pattern Languages in HCI: A Critical Review	GS: 243	Dearden & Finlay (2006)	Journal Article	Human Computer Interaction	Examine the concept of pattern language in human-computer interaction (HCI). Authors explain what patterns are, how are they used and what values they embody for HCI, for example in their development or use.	
Others						
Big Data Architecture Patterns - http://bigr.io/architecture/		Hardman (2016)	Website/PDF document	Big Data - architecture	6 big data, architectural patterns such as "lambda" or "data lake" are explored, based on a case study	x
A big data analytics design patterns to select customers for electricity theft inspection	Scopus: 0	Leal & Boldt (2016)	Journal Article	Big Data - general	Develop a solution architecture to select suspicious power-grid customers engaged in the electricity theft	x
Enrichment Patterns for Big Data	GS: 7	Holley et al. (2014)	Journal Article	Big Data - general	Define Data Evolution Framework that leads authors to detail data enrichment process and related 6 design patterns	
Patterns of business intelligence systems use in organizations	GS: 3	Arnott et al. (2017)	Journal Article	BI/Decision support	Conduct a case study research (cross-case analysis done with interviews) into 8 BI systems and 86 decisions supported by these systems. As a result, develop framework to describe BI use patterns using Gory & Morton framework.	
Introducing Design Patterns to Knowledge Processing Systems in the context of Big Data and Cloud Platforms	GS: 0	Nadschlager (2017)	Journal Article	Knowledge Processing Systems (KPS)	Conduct empirical study to name 12 design patterns candidates which are presented for development of KPS	
Layered Software Patterns for Data Analysis in Big Data Environment	GS: 0 / Scopus: 0	Hakeem (2016)	Journal Article	Big Data - architecture	Propose four-stage software model for big data problems based on layered patterns	

Figure A.20: Illustrates a literature database that is used in section 2.2.2.1. It provides a short summary of articles and internet sources that were reviewed, see `Master_DS_DP.xlsx`.

Protocol for Thesis: Discovering Data Science Design Patterns with Examples from R and Python Software Ecosystem

(After Final *Kick-Off* and How it Looks at the *Very End*)

1. Goal: Knowing how to approach my thesis regarding problem statement, research questions and methodology:
   a. Find and read guidelines on graduate thesis, LaTeX (a first-time user) and technical best practises
   b. Search articles on *design patterns*, *data science* and *software ecosystem* (for SECO: follow Jansen, largely)
   c. Clarify thesis structure and the research → Top-down research/writing approach selected

2. Write **chapter 1** (10 sharp):
   a. Introduce motivation & structure
   b. Establish *briefly* methodology and delimitations
      • Not a goal to replace these with chapters 3/5 at this stage!
   c. Make necessary graphic (structure) in Visio and table (RQ: 1,2 & 3) in LaTeX!
   d. Check for style & grammar AND context & research purpose/closing gaps must be clearly explained
   e. No summary in this chapter

3. Write **chapter 2** (40 sharp):
   a. *SECO*:
      • Brief explanation of it including definition – interconnect with R and Python
        - For SECO, need to include previous research too
      • How to survey tools/software in the research? → Need for related research on this and R's and Python's ecosystem
        - Create LaTeX table providing key characteristics of two programming languages as well
   b. *Data Science*:
      • Start with historical aspects and how it evolved from BI + statistics to DS
        - Similar LaTeX table between BI (1.0) and DS (2.0)
        - Characterized through "more than ever" big data, data scientists and (agile) iterative methodologies
      • *Design Patterns*
        - Consider historical development (but do not duplicate the introduction), over DP's use & application in related areas of big data, ML, deep learning, etc. → Establish a connection between DS and DP
   c. Summary of individual sections and their outcomes
   d. Graphics: CRISP-DM in Illustrator; BI/DW in PowerPoint; tables in LaTeX
      • Decide where to put them and when & how to introduce them!
   e. Revise all text for clarity & meaning & logic
   f. "Ensure that there is a flow of narrative that explains why each topic is being discussed" (Evans, 2014)

4. Write **chapter 3** (50 sharp) – Research Approach has to be as detailed as possible because it will be my guideline:
   a. Begin first with philosophy/qualitative methods and why them
   b. Explain 3D2P methodology and use of GIA (+ GTM; not right at the beginning, though)
   c. PP: Borrow core underlying principles of SLR to:
      • Gather relevant DS literature (iteratively & continuously throughout the whole research)
      • Practical screen & quality appraisal at last
      • Finish when all sources have been coded in the spreadsheet - data extraction / "discovery"
   d. PM: concept matrix
      • Synthesis/mining
      • Qualitative survey of R and Python tools from their ecosystem:
        - Purposeful sampling because of the high quantity of existing ones → be transparent and take into consideration section with surveying the tools → establish criteria in LaTeX table
   e. PW: guidelines for pattern writing and following pattern form
      • Mention pattern evaluation too
   f. Summary about major aspects
   g. Once all sources collected, re-iterate on this to further improve
      • Graphics: Make methodology in BPMN (3rd party tool) + statistics (in Word & Excel)

5. Do heavy lifting: At same time, start gathering & analysing relevant sample of literature & code it according to chapter 3
   a. Create 1 Excel DB that will capture all the literature information/codes
   b. Finish gathering and determinate a complete dataset (incl. formatting)
      • Needs to be in line with methodology
      • Once done, make a broad outline of numerous design patterns → fixed at 10
      • Note limitations of research
   c. Once a pattern candidate is established, make 2nd Excel DB and survey plausible R and Python tools from the ecosystem
      • Go over all applications and understand what each does and how they support DS ← see Table criteria
      • Document them in XMind - for R & Python (consider "native" approaches as well; all in 1 picture)
        - Need to mention DSTM in the discussion too, own section
   d. Go back: Rework, as necessary, the chapter 3

6. Write **chapter 4** - Back to formalizing DS DP using explained structure with R and Python code examples
   a. After trial-and-error, do not use illustration pictures, just code examples at the end
   b. List sequentially 10 design patterns
   c. Ensure there is a coherence between DPs which can help forming a DS pattern language
      • 4th step of pattern writing – patterns can be found/known variations have to be exemplified
      • Code samples should be in common notebooks .rmd/.ipynb
   d. Graphics: showcase & explain DSTM
   e. Summary – no deeper details as this is done in chapter 5

7. Write **chapter 5** – finish with:
   a. Discussion of results and ensuring it still targets my audience
      • DS patterns:
        - Relationship with a pattern language – talk about each individual pattern and then them as a whole
        - Categorize into 3 groups: data, modelling, infrastructure
      • DSTM:
        - Describe SciPy & tidyverse ecosystem which leads to 3 observations: universality, R/Python's size, GitHub vs others
   b. Implications and limitations
      • Future research – 2 paths and consider how design patterns relate to TOE(I) framework/sociotechnical system
      • Should be related only to future research not to e.g. SECOs individually
   c. Conclusion: what were the objectives, how and what was achieved

Other rules that were followed:

- Margin notes are made only where it is deemed to fit the purpose
- Over 4 items → list of (a) (b) (c) (d) bullet points with some exceptions already at 3
- (some) Statistical/Latex Guidelines:
  - Microtype package for protrusion
  - New term/highlighting: \emph{}
  - It is - - and not simple –
  - American quotes " " not British, etc. ones; but British spelling
  - Escape dots, e.g.\
  - 2 fonts: one for normal text + one for code listings
  - Use \dots within quotations

Figure A.21: Illustrates how does the thesis protocol look at the very end, see `Thesis_protocol.pdf`.

Gathered literature for formalizing Data Science Design Patterns

Original (=1) or Snowballed (=2) ?	Which underwent quality appraisal (3rd step)	Subjective Quality & Relevancy (very good, good, fair, poor)	Title	Year	Author(s)	Type of Source	Category / Sub-Category / Individual / Notes	Data					
								Ability to access a variety of data sources, e.g. DB, formats (JSON, CSV) - seamless integration	Locate, gather and understand data				
								import, export - reading from different raw formats/tools, csv, excel, disk, - databases	data scraping/web crawler	Data accessibility confidentially/collection: Identify + catalogue + document sources/available data (online/offline) -> create data dictionary	right data (int/ext., life cycle-3V; storage: what/how; understand it, learned vs. original)	insufficient data - acquire/crowdsourcing new external data from dedicated services (outsourcing)	(open, external) data -e.g. customer feedback/history data
2		very good	Patterns (and Anti-Patterns) for Developing Machine Learning Systems	2008	Gordon Rios	Presentation						x	x
1		very good	Best Practices in Data Mining	2003	Richard Boire, Paul Tyndall, Greg Carriere, Rob Champion	White Paper		x			x	x	x
2		very good	Patterns for Research in Machine Learning	2012	Ali Eslami	Blog Post	Including HN discussion to Ali Eslami's post https://news.ycombinator.com/item?id=4384317			x			
1		very good	Rules of Machine Learning:Best Practices for ML Engineering	2016	Martin Zinkevich	Report	Including HN discussion to Martin Zinkevich's document -- https://news.ycombinator.com/item?id=13414776			x	x	x	x
1		very good	Best Practices for Machine Learning Applications	2016	B Wujek, P Hall, F Güneş	Article				x	x		x
1		very good	Machine Learning essentials: Best practices, categories and misconceptions	2017	Gabriela Motroc	Article	Includes other 2 sources as well						
1		very good	Data Science and Big Data: Enterprise Paths to Success - Best Practices Report	2016	Fern Halper	Report		x		x	x	x	x
1		very good	The Practical Guide to Managing Data Science at Scale	2017	Domino Data Lab, Inc.	White Paper	pdf document			x	x	x	x
2		very good	The Data Science Handbook	2017	Field Cady	Book		x	x	x	x	x	
2		very good	Python Data Science Handbook	2016	Jake VanderPlas	Book		x					
2		very good	The Data Science Handbook: Advice and Insights from 25 Amazing Data Scientists	2015	Carl Shan, Henry Wang, Max Song, and William Chen	Book		x	x	x	x	x	x

Figure A.22: Illustrates an excerpt from a concept matrix that is used for discovering DS design pattern candidates in **chapter IV**. The full matrix is provided in the supplement, see `Master_DS_DP.xlsx`.

Tools's name	For what language is it?	Description	License	Name of Maintainer/Company	Code Repository	If source code repository on Github, number of "Stars" and "Forks"	Is it tested e.g. with Travis-CI?	Page on CRAN/PyPI	Dedicated Website for community/Online Documentation
Jupyter	Requires Python - supports several including R, SQL and Ruby	A web-based notebook environment for interactive computing	BSD 3-clause	Fernando Perez/Steering Council	https://github.com/jupyter/notebook	>= 3532 stars / 1121 forks	Yes	https://pypi.python.org/pypi/notebook	https://jupyter-notebook.readthedocs.io/en/stable/
RMarkdown	Requires R - supports several including Python, SQL and Ruby	Dynamic Documents for R	GPL-3	Yihui Xie (RStudio)	https://github.com/rstudio/rmarkdown	>= 1008 stars / 506 forks	Yes	https://cran.r-project.org/package=rmarkdown	http://rmarkdown.rstudio.com/
Pandas	Python	Powerful data structures for data analysis, time series,and statistics	BSD 3-clause	Wes McKinney	https://github.com/pandas-dev/pandas	>= 12 247 stars / 4 774 forks	Yes	https://pypi.python.org/pypi/pandas/	https://pandas.pydata.org/
blaze	Python	NumPy and Pandas interface to Big Data	BSD 3-clause	Continuum Analytics	https://github.com/blaze/blaze	>= 2236 stars / 307 forks	Yes	https://pypi.python.org/pypi/blaze	http://blaze.pydata.org/
data.table	R	data.table: Extension of 'data.frame'	Mozilla Public License 2.0 (MPL)	Matt Dowle (H2O.io)	https://github.com/Rdatatable/data.table/	>= 1262 stars / 659 forks	Yes	https://cran.r-project.org/package=data.table	http://r-datatable.com
tibble	R	tibble: Simple Data Frames	MIT license	Kirill Müller	https://github.com/tidyverse/tibble	>= 168 stars / 60 forks	Yes	https://cran.r-project.org/package=tibble	http://tibble.tidyverse.org/
Numpy	Python	NumPy: array processing for numbers, strings, records, and objects.	BSD 3-clause	Travis Oliphant / Numpy Developers	https://github.com/numpy/numpy	>= 6359 stars / 2567 forks	Yes	https://pypi.python.org/pypi/numpy	http://www.numpy.org/
tidyr	R	Easily Tidy Data - evolution of reshape2	MIT license	Hadley Wickham	https://github.com/tidyverse/tidyr/issues	>= 556 stars / 221 forks	Yes	https://cran.r-project.org/package=tidyr	http://tidyr.tidyverse.org/
reshape2	R	Flexibly Reshape Data: A Reboot of the Reshape Package	MIT license	Hadley Wickham	https://github.com/hadley/reshape	>= 167 stars / 45 forks	Yes	https://cran.r-project.org/package=reshape2	https://www.rdocumentation.org/packages/reshape2/
missingno	Python	Missing data visualization module for Python.	MIT license	Aleksey Bilogur	https://github.com/ResidentMario/missingno	>= 831 stars / 81 forks	No	https://pypi.python.org/pypi/missingno/	See GitHub
fancyimpute	Python	Multivariate imputation and matrix completion algorithms implemented in Python	Apache License 2.0	Alex Rubinsteyn	https://github.com/iskandr/fancyimpute	>= 288 stars / 51 forks	Yes	https://pypi.python.org/pypi/fancyimpute	See GitHub
mice	R	Multivariate Imputation by Chained Equations	>= GPL-2	Stef van Buuren	https://github.com/stefvanbuuren/mice	>= 46 stars / 26 forks	No	https://cran.r-project.org/package=mice	http://stefvanbuuren.github.io/mice/
VIM	R	Visualization and Imputation of Missing Values	>= GPL-2	Matthias Templ	https://github.com/statistikat/VIM	>= 28 stars / 7 forks	Yes	https://cran.r-project.org/package=VIM	https://www.rdocumentation.org/packages/VIM
Shogun	Multiplatform, including R and Python	Unified and efficient machine learning library	GPL-3	Sören Sonnenburg & Gunnar Rätsch	https://github.com/shogun-toolbox/shogun	>= 1932 stars / 826 forks	Yes	https://pypi.python.org/pypi/shogun-ml	http://www.shogun-toolbox.org/
mlr	R	Machine Learning in R	BSD 2-clause	Bernd Bischl	https://github.com/mlr-org/mlr	>= 884 stars / 262 forks	Yes	https://cran.r-project.org/package=mlr	http://mlr-org.github.io/
scikit-learn	Python	A set of python modules for machine learning and data mining	BSD 3-clause	Andreas Mueller	https://github.com/scikit-learn/scikit-learn	>= 25 378 stars / 12 964 forks	Yes	https://pypi.python.org/pypi/scikit-learn/	http://scikit-learn.org/stable/
caret	R	Classification and Regression Training	>= GPL-2	Max Kuhn (RStudio)	https://github.com/topepo/caret	>= 759 stars / 379 forks	Yes	https://cran.r-project.org/package=caret	http://topepo.github.io/caret/
ml-ensemble	Python	Machine Learning Ensemble Library	MIT license	Sebastian Flennerhag	https://github.com/flennerhag/mlens	>= 127 stars / 25 forks	Yes	https://pypi.python.org/pypi/mlens	http://ml-ensemble.com/
Mlxtend	Python	Machine Learning Library Extensions	BSD 3-clause	Sebastian Raschka	https://github.com/rasbt/mlxtend	>= 1295 stars / 332 forks	Yes	https://pypi.python.org/pypi/mlxtend/	http://rasbt.github.io/mlxtend/
caretEnsemble	R	Ensembles of Caret Models	MIT license	Zachary A. Deane-Mayer	https://github.com/zachmayer/caretEnsemble	>= 171 stars / 65 forks	Yes	https://cran.r-project.org/package=caretEnsemble	https://www.rdocumentation.org/packages/caretEnsemble
SuperLearner	R	Prediction model ensembling method	GPL-3	Eric Polley (Mayo Clinic)	https://github.com/ecpolley/SuperLearner	>= 96 stars / 36 forks	Yes	https://cran.r-project.org/package=SuperLearner	https://www.rdocumentation.org/packages/SuperLearner/
mlrMBO	R	Toolbox for Bayesian Optimization and Model-Based Optimization in R	MIT license	Bernd Bischl	https://github.com/mlr-org/mlrMBO	>= 74 stars / 24 forks	Yes	https://cran.r-project.org/package=mlrMBO	https://mlr-org.github.io/mlrMBO/
Bokeh	Python	Interactive plots and applications in the browser from Python	BSD 3-clause	Bryan Van de Ven	https://github.com/bokeh/bokeh	>= 7200 stars / 1805 forks	Yes	https://pypi.python.org/pypi/bokeh	https://bokeh.pydata.org/en/latest/
Plot.ly Dash	Python	Python framework for building analytical web applications	MIT license	Chris Parmer	https://github.com/plotly/dash	>= 3240 stars / 290 forks	Yes	https://pypi.python.org/pypi/dash	https://plot.ly/dash/
Shiny	R	Web Application Framework for R	GPL-3	Winston Chang (RStudio)	https://github.com/rstudio/shiny	>= 2640 stars / 1324 forks	Yes	https://cran.r-project.org/package=shiny	https://shiny.rstudio.com/
Boto3	Python	Amazon Web Service SDK for Python	Apache License 2.0	Amazon Web Services	https://github.com/boto/boto3	>= 2962 stars / 592 forks	Yes	https://pypi.python.org/pypi/boto3	https://boto3.readthedocs.io/en/latest/
azure	Python	Microsoft Azure Client Libraries for Python	MIT license	Laurent Mazuel/Microsoft	https://github.com/Azure/azure-sdk-for-python	>= 493 stars / 410 forks	Yes	https://pypi.python.org/pypi/azure/	https://docs.microsoft.com/en-us/python/azure/
google-cloud	Python	Python idiomatic client for Google Cloud Platform services.	Apache License 2.0	Google Cloud Platform	https://github.com/GoogleCloudPlatform/google-cloud-python	>= 1216 stars / 543 forks	Yes	https://pypi.python.org/pypi/google-cloud	https://google-cloud-python.readthedocs.io/en/latest/
rsconnect	R	Interface for RPubs, shinyapps.io, and RStudio Connect	GPL-2	JJ Allaire (RStudio)	https://github.com/rstudio/rsconnect	>= 33 stars / 33 forks	Yes	https://cran.r-project.org/package=rsconnect	https://www.rdocumentation.org/packages/rsconnect/
cloudml	R	R interface to Google CloudML	Apache License 2.0	Javier Luraschi (RStudio)	https://github.com/rstudio/cloudml	>= 25 stars / 8 forks	Yes	In development on GitHub	https://tensorflow.rstudio.com/tools/cloudml/
doAzureParallel	R	Allows users to submit parallel workloads in Microsoft Azure Cloud	MIT license	Brian Hoang (Microsoft)	https://github.com/Azure/doAzureParallel	>= 65 stars / 28 forks	Yes	In development on GitHub	See GitHub
AzureML	R	R interface to AzureML	MIT license	Rich Calaway (Microsoft)	https://github.com/RevolutionAnalytics/AzureML	>= 39 stars / 18 forks	Yes	https://cran.r-project.org/package=AzureML	https://www.rdocumentation.org/packages/AzureML/

Figure A.23: Illustrates seventeen R and fifteen Python software tools that were identified and some of which were used for design pattern code examples. Only two R packages (`cloudml` and `doAzureParallel`) are as of March 2018 in development on *GitHub*. See the database in `Master_DS_DP.xlsx` as well.

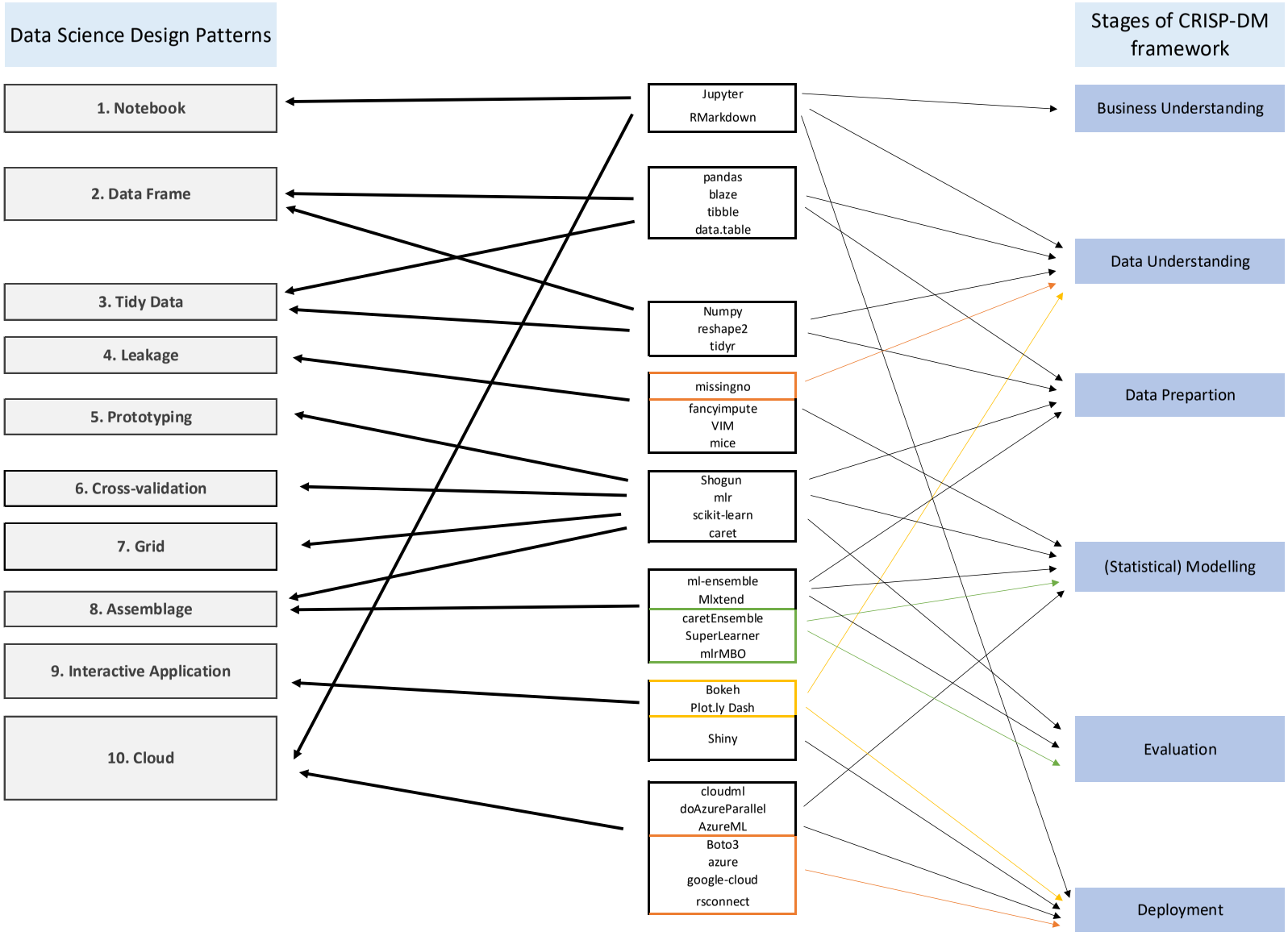


Figure A.24: Illustrates links between formalized DS design patterns, identified software tools and their association with stages of CRISP-DM framework, see `Master_DS_DP.xlsx`.

APPENDIX B

Tables

Table B.6: Presents steps by which information sources are collected and processed. For more, see the methodology in Figure 3 too.

Step	Source	Description	Used for ...	Total # of resources
Keywords (initial search)	Scopus, Web of Science, Google (Scholar)	Material, including grey literature, where following keywords were found: "(best/good) approach(es)/advice(s)/practise(s)/ solution(s)/recommendation(s)", "data science/ML/BI/big data", "data analytics/analysis/mining", "(abstract/common/frequent) problem(s)/mistake(s)", "lesson(s) learned", "(design) pattern(s)"	Inclusion	26
Practical Screen	———	Documents must have a clear purpose and addressing objectives of the study.	Exclusion	
Snowballing	Sample of 26 resources	Additional information is collected by means of inspecting identified documents and extending an initial sample through iteratively snowballing other studies.	Inclusion	66
Quality assurance	Sample of 66 resources	A number of documents, which were classified as "poor" and "fair" on a four-step scale (very good, good, fair and poor), are excluded.	Exclusion	41

Table B.7: Presents, in order of importance, criteria which are used for tool's inclusion and exclusion in the study.

#	Criteria	Type	Description	Used for ...
1	Release on CRAN or PyPI	Qualitative	At the point of survey, included packages should be released on either central repository – hence being open sourced in nature. If a package has not yet been published due to being in development, upon author's own discretion it might be considered as well.	Inclusion
2	Purpose	Qualitative	A package is excluded if it does not have a clear purpose and addressing the needs of each data science design pattern.	Exclusion
3	Website	Qualitative	A package is excluded if it does not have a dedicated website (including a public source code repository) where users can find more information, for example documentation.	
4	Usage Metrics	Quantitative	A package is excluded if does not have a recognizable number of downloads or other metrics supporting being used in the community, made available for instance by `Rdocumentation.org`, `anaconda.org` or `Depsy.org`.	
5	Maintenance	Quantitative	A package is excluded if it seems to be abandoned in terms of changes to the source code repository – in fact it should have a publicly released version during a time period since January 2016 until March 2018.	

BIBLIOGRAPHY

REFERENCES

Aceto, G., Botta, A., De Donato, W., & Pescapè, A. (2013). Cloud monitoring: A survey. *Computer Networks*, *57*(9), 2093–2115. doi:10.1016/j.comnet.2013.04.001

Agarwal, R. & Dhar, V. (2014). Big Data, Data Science, and Analytics: The Opportunity and Challenge for IS Research. *Information Systems Research*, *25*(3), 443–448. doi:10.1287/isre.2014.0546

Aguinis, H., Forcum, L. E., & Joo, H. (2012). Using Market Basket Analysis in Management Research. *Journal of Management*, *39*(7), 1799–1824. doi:10.1177/0149206312466147

Alexander, C., Ishikawa, S., & Silverstein, M. (1977). *A pattern language: towns, buildings, construction*. Oxford University Press. Retrieved from https://www.patternlanguage.com/bookstore/pattern-language.html

Ampatzoglou, A., Charalampidou, S., & Stamelos, I. (2013). Research state of the art on GoF design patterns: A mapping study. *Journal of Systems and Software*, *86*(7), 1945–1964. doi:10.1016/j.jss.2013.03.063

Arcitura. (2018). Big Data Patterns. *For more see Erl, Khattak & Buhler's Big Data Fundamentals: Concepts, Drivers & Techniques (2016)*. The Prentice Hall Service Technology Series from Thomas Erl. Retrieved from www.bigdatapatterns.org

Arel, I., Rose, D., & Coop, R. (2009). DeSTIN: A Scalable Deep Learning Architecture with Application to High-Dimensional Robust Pattern Recognition. In *Proc. of the aaai 2009 fall symposium on biologically inspired cognitive architectures*. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.232.9693

Arlot, S. & Celisse, A. (2010). A survey of cross-validation procedures for model selection. *Statistics Surveys*, *4*, 40–79. doi:10.1214/09-SS054

Arnott, D., Lizama, F., & Song, Y. (2017). Patterns of business intelligence systems use in organizations. *Decision Support Systems*, *97*, 58–68. doi:10.1016/j.dss.2017.03.005

Asay, M. (2016). Exponential growth of R's open source community threatens commercial competitors. Retrieved from http://www.techrepublic.com/article/exponential-growth-of-rs-open-source-community-threatens-commercial-competitors/

Ayankoya, K., Calitz, A., & Greyling, J. (2014). Intrinsic Relations between Data Science, Big Data, Business Analytics and Datafication. In *Proceedings of the southern african institute for computer scientist and information technologists annual conference 2014* (pp. 192–198). doi:10.1145/2664591.2664619

Azur, M. J., Stuart, E. A., Frangakis, C., & Leaf, P. J. (2011). Multiple imputation by chained equations: What is it and how does it work? *International Journal of Methods in Psychiatric Research*, *20*(1), 40–49. doi:10.1002/mpr.329

Bafandeh Mayvan, B., Rasoolzadegan, A., & Ghavidel Yazdi, Z. (2017). The state of the art on design patterns: A systematic mapping of the literature. *Journal of Systems and Software*, *125*, 93–118. doi:10.1016/j.jss.2016.11.030

Baker, J. (2012). The Technology-Organization-Environment Framework. In *Information systems theory* (pp. 231–245). Springer. doi:10.1007%2F978-1-4419-6108-2_12

Baraldi, A. N. & Enders, C. K. (2010). An introduction to modern missing data analyses. *Journal of School Psychology*, *48*(1), 5–37. doi:10.1016/j.jsp.2009.10.001

Barbosa, O., Santos, R., Alves, C., Werner, C., & Jansen, S. (2013). A systematic mapping study on software ecosystems from a three-dimensional perspective. In *Software ecosystems: Analyzing and managing business networks in the software industry* (pp. 59–81). Edward Elgar Publishing. doi:10.4337/9781781955635.00011

Basili, V. R., Caldiera, G., & Rombach, H. D. (1994). *The goal question metric approach*. http://www.cs.umd.edu/~mvz/handouts/gqm.pdf and http://onlinelibrary.wiley.com/doi/10.1002/0471028959.sof142/abstract.

Bauer, E., Kohavi, R., Chan, P., Stolfo, S., & Wolpert, D. (1999). An Empirical Comparison of Voting Classification Algorithms: Bagging, Boosting, and Variants. *Machine Learning*, *36*. doi:10.1023/A:1007515423169

Baxter, G. D. & Sommerville, I. (2011). Socio-technical systems: From design methods to systems engineering. *Interacting with Computers*, *23*(1), 4–17. doi:10.1016/j.intcom.2010.07.003

Bengtsson, H. (2017). Milestone: 12000 packages on CRAN. Retrieved from https://stat.ethz.ch/pipermail/r-devel/2017-December/075231.html

Bergin, J., Eckstein, J., Volter, M., Sipos, M., Wallingford, E., Marquardt, K., … Manns, M. L. (2012). *Pedagogical Patterns: Advice For Educators*. CreateSpace Independent Publishing Platform. Retrieved from http://oro.open.ac.uk/34138/

Bergstra, J. & Bengio, Y. (2012). Random search for hyper-parameter optimization. *The Journal of Machine Learning Research*, *13*(1), 281–305. Retrieved from https://dl.acm.org/citation.cfm?id=2188395

Bhatt, G. D. (2001). Knowledge management in organizations: examining the interaction between technologies, techniques, and people. *Journal of Knowledge Management*, *5*(1). doi:10.1108/13673270110384419

Biňas, M. (2013). Version Control System in CS1 Course: Practical Experience. In *11th ieee international conference on emerging elearning technologies and applications*. IEEE. doi:10.1109/ICETA.2013.6674398

Blaha, M. (2010). *Patterns of data modeling*. CRC Press. Retrieved from https://www.crcpress.com/Patterns-of-Data-Modeling/Blaha/p/book/9781439819890

Blanzieri, E. & Bryl, A. (2008). A survey of learning-based techniques of email spam filtering. *Artificial Intelligence Review*, *29*(1), 63–92. doi:10.1007/s10462-009-9109-6

Boehmke, B. C. (2016). *Data Wrangling with R*. Springer. doi:10.1007/978-3-319-45599-0

Boire, R., Tyndall, P., Carriere, G., & Champion, R. (2003). *Best Practices in Data Mining* (tech. rep. No. August). Retrieved from https://www.the-cma.org/disciplines/analytics/archive/best-practices-in-data-mining

Bonchev, D. & Thomas, S. (2010). A survey of current software for network analysis in molecular biology. *Human Genomics*, *4*(5), 353. doi:10.1186/1479-7364-4-5-353

Bosch, J. (2009). From Software Product Lines to Software Ecosystems. In *Proceedings of the 13th international software product line conference* (pp. 111–119). Carnegie Mellon University. Retrieved from http://dl.acm.org/citation.cfm?id=1753235.1753251

Breck, E., Cai, S., Nielsen, E., Salib, M., & Sculley, D. (2016). What's your ML Test Score? A rubric for ML production systems. In *Reliable machine learning in the wild - nips 2016 workshop*. Retrieved from https://research.google.com/pubs/pub45742.html

Bruce, P. & Bruce, A. (2017). *Practical Statistics for Data Scientists*. O'Reilly Media. Retrieved from http://shop.oreilly.com/product/0636920048992.do

Brunner, R. J. & Kim, E. J. (2016). Teaching Data Science. *Procedia Computer Science*, *80*, 1947–1956. doi:10.1016/j.procs.2016.05.513

Bryan, J. & Wickham, H. (2017). Data Science: A Three Ring Circus or a Big Tent? *Journal of Computational and Graphical Statistics*, *26*(4), 784–785. doi:10.1080/10618600.2017.1389743

Bühlmann, P. (2012). Bagging, boosting and ensemble methods. In J. Gentle, W. Härdle, & Y. Mori (Eds.), *Handbook of computational statistics: Concepts and methods* (2nd, pp. 985–1022). Springer. doi:10.1007/978-3-642-21551-3_33

Cady, F. (2017). *The Data Science Handbook*. Wiley. Retrieved from https://www.wiley.com/en-us/The+Data+Science+Handbook-p-9781119092940

Cameron, R. (2009). A sequential mixed model research design: Design, analytical and display issues. *International Journal of Multiple Research Approaches*, *3*(2), 140–152. doi:10.5172/mra.3.2.140

Cao, L. (2016). Data Science: Nature and Pitfalls. *IEEE Intelligent Systems*, *31*(5). doi:10.1109/MIS.2016.86

Cao, L. (2017). Data Science: A Comprehensive Overview. *ACM Computing Surveys*, *50*(3), 1–42. doi:10.1145/3076253

Carbone, A., Jensen, M., & Sato, A.-H. (2016). Challenges in data science: a complex systems perspective. *Chaos, Solitons & Fractals*, *90*, 1–7. doi:10.1016/j.chaos.2016.04.020

Carr, D. (1999). Guidelines for designing information visualization applications. In *Proceedings of the 1999 ericsson conference on usability engineering*. Retrieved from http://urn.kb.se/resolve?urn=urn:nbn:se:ltu:diva-39533

Carson, D., Gilmore, A., Perry, C., & Gronhaug, K. (2001). *Qualitative Marketing Research*. SAGE Publications. Retrieved from https://us.sagepub.com/en-us/nam/qualitative-marketing-research/book208811

Cass, S. & Diakopoulos, N. (2017). The 2017 Top Programming Languages. IEEE. Retrieved from http://spectrum.ieee.org/computing/software/the-2017-top-programming-languages

Chapman, P., Clinton, J., Kerber, R., Khabaza, T., Reinartz, T., Shearer, C., & Wirth, R. (2000). *Step-by-step data mining guide*. CRISP-DM consortium | IBM. Retrieved from https://www.the-modeling-agency.com/crisp-dm.pdf

Chen, H. (2004). Towards Design Patterns for Dynamic Analytical Data Visualization. *Visualization and Data Analysis*. doi:10.1117/12.539227

Chen, H., Chiang, R. H. L., & Storey, V. C. (2012). Business Intelligence and Analytics: From Big Data to Big Impact. *MIS Quarterly*, *36*(4), 1165–1188. Retrieved from http://dl.acm.org/citation.cfm?id=2481674.2481683

Christensen, H. B., Hansen, K. M., Kyng, M., & Manikas, K. (2014). Analysis and design of software ecosystem architectures - Towards the 4S telemedicine ecosystem. *Information and Software Technology*, *56*(11), 1476–1492. doi:10.1016/j.infsof.2014.05.002

Chu, R., Duling, D., & Thompson, W. (2007). *Best Practices for Managing Predictive Models in a Production Environment*. Retrieved from http://www2.sas.com/proceedings/forum2007/076-2007.pdf

Clarke, D. (2013). *Seven steps to success machine learning in practice*. Retrieved from http://daoudclarke.github.io/guide.pdf

Codd, E. F. (1970). A Relational Model of Data for Large Shared Data Banks. *Commun. ACM*, *13*(6), 377–387. doi:10.1145/362384.362685

Collier, K. (2011). *Agile Analytics: A Value-Driven Approach to Business Intelligence and Data Warehousing*. Addison-Wesley Professional. Retrieved from https://dl.acm.org/citation.cfm?id=2025246

Constantinou, E. & Mens, T. (2017). An empirical comparison of developer retention in the RubyGems and npm software ecosystems. *Innovations in Systems and Software Engineering*, *13*(2-3), 101–115. doi:10.1007/s11334-017-0303-4

Conway, D. (2013). The Data Science Venn Diagram. Retrieved from http://drewconway.com/zia/2013/3/26/the-data-science-venn-diagram

Corbin, J. M. & Strauss, A. (1990). Grounded theory research: Procedures, canons, and evaluative criteria. *Qualitative Sociology*, *13*(1), 3–21. doi:10.1007/BF00988593

Corea, F. (2016). *Big Data Analytics: A Management Perspective*. Springer. doi:10.1007/978-3-319-38992-9

Cortez, P. & Morais, A. (2007). A Data Mining Approach to Predict Forest Fires using Meteorological Data. In J. M. Neves, M. F. Santos, & J. M. Machado (Eds.), *New trends in artificial intelligence: Proceedings of the 13th portuguese conference on artificial intelligence* (pp. 512–523). Retrieved from http://hdl.handle.net/1822/8039

Coyne, I. T. (1997). Sampling in qualitative research. Purposeful and theoretical sampling; merging or clear boundaries? *Journal of Advanced Nursing*, *26*(3), 623–630. doi:10.1046/j.1365-2648.1997.t01-25-00999.x

Creswell, J. (2013). *Research Design: Qualitative, Quantitative, and Mixed Methods Approaches*. SAGE Publications. Retrieved from https://us.sagepub.com/en-us/nam/research-design/book237357

Cusumano, M. A. & Jansen, S. (2013). Defining software ecosystems: a survey of software platforms and business network governance. In S. Jansen, S. Brinkkemper, & M. Cusumano (Eds.), *Software ecosystems: Analyzing and managing business networks in the software industry* (Vol. 13). Edward Elgar Pub. Retrieved from https://www.elgaronline.com/view/9781781955628.00008.xml

DAMA. (2017). *DAMA-DMBOK: Data Management Body of Knowledge* (2nd). Technics Publications. Retrieved from https://dama.org/content/body-knowledge

Davenport, T. H. (2013). Analytics 3.0. *Harvard Business Review*. Retrieved from https://hbr.org/2013/12/analytics-30

Davenport, T. H. & Patil, D. (2012). Data Scientist: The Sexiest Job of the 21st Century. *Harvard Business Review*. Retrieved from https://hbr.org/2012/10/data-scientist-the-sexiest-job-of-the-21st-century

De Mauro, A., Greco, M., & Grimaldi, M. (2015). What is big data? A consensual definition and a review of key research topics. In *Aip conference proceedings - international conference on integrated information* (Vol. 1644, *1*, pp. 97–104). AIP Publishing. doi:10.1063/1.4907823

Dearden, A. & Finlay, J. (2006). Pattern Languages in HCI: A Critical Review. *Human-Computer Interaction*, *21*(1), 49–102. doi:10.1207/s15327051hci2101_3

Delibašić, B., Kirchner, K., & Ruhland, J. (2008). A Pattern Based Data Mining Approach. In C. Preisach, H. Burkhardt, L. Schmidt-Thieme, & R. Decker (Eds.), *Data analysis, machine learning and applications. studies in classification, data analysis, and knowledge organization* (pp. 327–334). Springer. doi:10.1007/978-3-540-78246-9_39

DePietro, R., Wiarda, E., & Fleischer, M. (1990). The context for change: Organization, technology and environment. In L. G. Tornatzky & M. Fleischer (Eds.), *The processes of technological innovation* (pp. 151–175). Lexington Books. Retrieved from https://books.google.com/books?id=EotRAAAAMAAJ

Dhar, V. (2013). Data Science and Prediction. *Communications of the ACM*, *56*(12), 64–73. doi:10.1145/2500499

Dhungana, D., Groher, I., Schludermann, E., & Biffl, S. (2010). Software Ecosystems vs. Natural Ecosystems: Learning from the Ingenious Mind of Nature. In *Proceedings of the fourth european conference on software architecture: Companion volume* (pp. 96–102). doi:10.1145/1842752.1842777

Dichev, C. & Dicheva, D. (2017). Towards Data Science Literacy. *Procedia Computer Science*, *108*, 2151–2160. doi:10.1016/j.procs.2017.05.240

Dick, M. (2014). Interactive Infographics and News Values. *Digital Journalism*, *2*(4), 490–506. doi:10.1080/21670811.2013.841368

Dietrich, D. (2014). Building Data Science Teams. Retrieved from https://www.slideshare.net/emcacademics/building-data-science-teams-31057129

Domino. (2017). *The Practical Guide to Managing Data Science at Scale*. Domino Data Lab, Inc. Retrieved from https://www.dominodatalab.com/resources/managing-data-science/

Donoho, D. (2017). 50 Years of Data Science. *Journal of Computational and Graphical Statistics*, *26*(4), 745–766. doi:10.1080/10618600.2017.1384734

Douglass, B. P. (2002). *Real-Time Design Patterns: Robust Scalable Architecture for Real-Time Systems*. Addison Wesley Professional. Retrieved from https://dl.acm.org/citation.cfm?id=557125

Dowle, M. & Srinivasan, A. (2018). data.table: Extension of data.frame. Retrieved from https://cran.r-project.org/package=data.table

Durao, F., Carvalho, J. F. S., Fonseka, A., & Garcia, V. C. (2014). A systematic review on cloud computing. *Journal of Supercomputing*, *68*(3), 1321–1346. doi:10.1007/s11227-014-1089-x

Edirisingha, P. (2012). Interpretivism and Postivism (Ontological and Epistemological Perspectives). Retrieved from https://prabash78.wordpress.com/2012/03/14/interpretivism-and-postivism-ontological-and-epistemological-perspectives/

Al-Eisawi, D. & Lycett, M. (2012). Business Intelligence - Definitions, managerial effects and aspects: A Systematic Literature Review. In *Proceedings of the 14th international conference on enterprise information systems* (pp. 209–214). doi:10.5220/0004005902090214

Eppler, M. J. & Mengis, J. (2004). The Concept of Information Overload - A Review of Literature from Organization Science, Accounting, Marketing, MIS, and Related Disciplines. *The Information Society*, *20*(5), 325–344. doi:10.1080/01972240490507974

Erl, T., Khattak, W., & Buhler, P. (2016). *Big Data Fundamentals: Concepts, Drivers & Techniques*. The Prentice Hall Service Technology Series from Thomas Erl. Prentice Hall. Retrieved from https://dl.acm.org/citation.cfm?id=2898954

Erl, T. (2015). *Cloud Computing Design Patterns*. The Prentice Hall Service Technology Series from Thomas Erl. Prentice Hall. Retrieved from https://dl.acm.org/citation.cfm?id=2810076

Eslami, A. (2012). Patterns for Research in Machine Learning. Retrieved from http://arkitus.com/patterns-for-research-in-machine-learning/

Evelson, B. (2017). *It's Time To Upgrade Business Intelligence To Systems Of Insight*. Forrester Research. Retrieved from https://www.forrester.com/report/Its+Time+To+Upgrade+Business+Intelligence+To+Systems+Of+Insight/-/E-RES122481

Everitt, B. S. & Hothorn, T. (2006). *A Handbook of Statistical Analyses Using R*. Chapman & Hall/CRC. Retrieved from https://dl.acm.org/citation.cfm?id=1213890

Ezust, A. & Ezust, P. (2011). *Introduction to Design Patterns in C++ with Qt* (2nd). Prentice Hall. Retrieved from https://www.ics.com/designpatterns/book/main.html

Fabian, R. (2013). Design Patterns. Retrieved from http://www.dataorienteddesign.com/dodmain/node16.html

Fehling, C., Leymann, F., Retter, R., Schupeck, W., & Arbitter, P. (2014). *Cloud Computing Patterns*. Springer. doi:10.1007/978-3-7091-1568-8

Finkelstein, A., Brinkkemper, S., & Jansen, S. (2009). A sense of community: A research agenda for software ecosystems. In *31st international conference on software engineering* (pp. 187–190). doi:10.1109/ICSE-COMPANION.2009.5070978

Fiřtík, Z. (2017). *Vizualizace výsledků data miningu prostřednictvím BI nástrojů*. University of Economics, Prague. Retrieved from https://vskp.vse.cz/69656_vizualizace_vysledku_data_miningu_prostrednictvim_bi_nastroju

Fournier-Viger, P. (2013). An Introduction to Frequent Pattern Mining. Retrieved from http://data-mining.philippe-fournier-viger.com/introduction-frequent-pattern-mining/

Fowler, M. (2002). *Patterns of Enterprise Application Architecture* (1st). Addison-Wesley Professional. Retrieved from https://martinfowler.com/books/eaa.html

Fowler, M. (2006). Writing Software Patterns. Retrieved from https://www.martinfowler.com/articles/writingPatterns.html

Franco-Bedoya, O., Ameller, D., Costal, D., & Franch, X. (2017). Open source software ecosystems: A Systematic mapping. *Information and Software Technology*, *91*, 160–185. doi:10.1016/j.infsof.2017.07.007

Freeman, E., Robson, E., Bates, B., & Sierra, K. (2009). *Head First Design Patterns: A Brain-Friendly Guide* (1st). O'Reilly Media. Retrieved from http://shop.oreilly.com/product/9780596007126.do

Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). *Design Patterns - Elements of Reusable Object-Oriented Software*. Addison-Wesley Professional. Retrieved from https://dl.acm.org/citation.cfm?id=186897

García-Laencina, P. J., Sancho-Gómez, J., & Figueiras-Vidal, A. R. (2010). Pattern classification with missing data: a review. *Neural Computing and Applications*, *19*(2), 263–282. doi:10.1007/s00521-009-0295-6

Geist, K., Geist, E., & Kuznik, K. (2012). The patterns of music: Young children learning mathematics through beat, rhythm, and melody. *YC Young Children*, *67*(1), 74–79. Retrieved from https://eric.ed.gov/?id=EJ975497

Gelman, A. & Hill, J. (2006). *Data Analysis Using Regression and Multilevel/Hierarchical Models*. Cambridge University Press. Retrieved from http://www.stat.columbia.edu/~gelman/arm/

Gentile, B. (2015). The Evolution of Business Intelligence. Retrieved from https://channels.theinnovationenterprise.com/articles/the-evolution-of-business-intelligence

German, D. M., Adams, B., & Hassan, A. E. (2013). The Evolution of the R Software Ecosystem. In *17th european conference on software maintenance and reengineering* (pp. 243–252). IEEE. doi:10.1109/CSMR.2013.33

Gershon, N. (1998). Visualization of an imperfect world. *IEEE Computer Graphics and Applications*, *18*(4), 43–45. doi:10.1109/38.689662

Giridhar, C. (2016). *Learning Python Design Patterns* (2nd). Packt Publishing. Retrieved from http://shop.oreilly.com/product/9781785888038.do

Goebel, M. & Gruenwald, L. (1999). A survey of data mining and knowledge discovery software tools. *ACM SIGKDD Explorations Newsletter*, *1*(1). Retrieved from http://dl.acm.org/citation.cfm?id=846172

Golafshani, N. (2003). Understanding Reliability and Validity in Qualitative Research. *The Qualitative Report*, *8*(4), 597–606. Retrieved from http://nsuworks.nova.edu/tqr/vol8/iss4/6/

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press. Retrieved from http://www.deeplearningbook.org

Grolemund, G. & Wickham, H. (2018). *R for Data Science: Import, Tidy, Transform, Visualize, and Model Data*. O'Reilly Media. Retrieved from http://r4ds.had.co.nz/

Guerrero, L. A. & Fuller, D. A. (2001). A pattern system for the development of collaborative applications. *Information and Software Technology*, *43*(7), 457–467. doi:10.1016/S0950-5849(01)00154-9

Guetterman, T. C., Fetters, M. D., & Creswell, J. W. (2015). Integrating quantitative and qualitative results in health science mixed methods research through joint displays. *The Annals of Family Medicine*, *13*(6), 554–561. doi:10.1370/afm.1865

Guha, R. & Al-Dabass, D. (2010). Impact of Web 2.0 and Cloud Computing Platform on Software Engineering. In *International symposium on electronic system design* (pp. 213–218). doi:10.1109/ISED.2010.48

Gurjar, Y. S. & Rathore, V. S. (2013). Cloud Business Intelligence - Is What Business Need Today. *International Journal of Recent Technology and Engineering*, *1*(6). Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.675.8516&rep=rep1&type=pdf

Gutierrez-Osuna, R. (2002). LECTURE 13: Cross-validation. Texas A&M University. Retrieved from https://www.cs.tau.ac.il/~nin/Courses/NC05/pr_l13.pdf

Hajimia, H. (2014). Research Method - Sampling. Retrieved from https://slideshare.net/hafizahhajimia/research-method-sampling

Hakeem, H. (2017). Layered software patterns for data analysis in big data environment. *International Journal of Automation and Computing*, 1–11. doi:10.1007/s11633-016-1043-x

Halper, F. (2016). *Data Science and Big Data: Enterprise Paths to Success*. TDWI. Retrieved from https://tdwi.org/research/2016/12/best-practices-report-data-science-and-big-data/asset.aspx

Hansen, P. B. (1995). *Studies in computational science: parallel programming paradigms*. Prentice Hall. Retrieved from https://dl.acm.org/citation.cfm?id=526904

Harper, J. (2014). Distinguishing Analytics, Business Intelligence, Data Science. Retrieved from http://www.dataversity.net/distinguishing-analytics-business-intelligence-data-science/

Hartigh, E. d., Tol, M., & Visscher, W. (2006). The health measurement of a business ecosystem. In *Proceedings of the european network on chaos and complexity research and management practice meeting* (pp. 1–39). Retrieved from https://www.researchgate.net/publication/288583566_Measuring_the_health_of_a_business_ecosystem

Hasan, H., Ahmad, S., Osman, B. M., Sapri, S., & Othman, N. (2017). A comparison of model-based imputation methods for handling missing predictor values in a linear regression model: A simulation study. *AIP Conference Proceedings*, *1870*. doi:10.1063/1.4995930

Hastie, T., Tibshirani, R., & Friedman, J. (2017). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. doi:10.1007/978-0-387-84858-7

Hattotuwagama, C. K., Doytchinova, I. A., Guan, P., & Flower, D. R. (2008). In silico QSAR-based predictions of class I and class II MHC epitopes. In *Immunoinformatics* (pp. 63–89). doi:10.1007/978-0-387-72968-8_4

Heer, J. M. & Agrawala, M. (2006). Software Design Patterns for Information Visualization. *IEEE Transactions on Visualization and Computer Graphics*, *12*(5). doi:10.1109/TVCG.2006.178

Heijden, G. J. M. G. v. d., T. Donders, A. R., Stijnen, T., & Moons, K. G. M. (2006). Imputation of missing values is superior to complete case analysis and the missing-indicator method in multivariable diagnostic research: A clinical example. *Journal of Clinical Epidemiology*, *59*(10), 1102–1109. doi:10.1016/j.jclinepi.2006.01.015

Heinze, J. (2014). History of Business Intelligence. Retrieved from https://www.betterbuys.com/bi/history-of-business-intelligence/

Hejdánek, M. (2016). *Návrh controllingové koncepce s využitím systému Business intelligence*. University of Economics, Prague. Retrieved from https://vskp.vse.cz/70950_navrh_controllingove_koncepce_svyuzitim_systemu_business_intelligence

Henke, N., Bughin, J., Chui, M., Manyika, J., Saleh, T., Wiseman, B., & Sethupathy, G. (2016). *The age of analytics: Competing in a data-driven world*. McKinsey. Retrieved from https://www.mckinsey.com/business-functions/mckinsey-analytics/our-insights/the-age-of-analytics-competing-in-a-data-driven-world

Hofstede, G. (1993). Cultural constraints in management theories. *Academy of Management Perspectives*, *7*(1), 81–94. doi:10.5465/AME.1993.9409142061

Hofstede, G. (2011). Dimensionalizing Cultures: The Hofstede Model in Context. *Online Readings in Psychology and Culture*, *2*(1). doi:10.9707/2307-0919.1014

Hohpe, G. & Woolf, B. (2003). *Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions*. Addison-Wesley Professional. Retrieved from https://dl.acm.org/citation.cfm?id=940308

Holley, K., Sivakumar, G., & Kannan, K. (2014). Enrichment Patterns for Big Data. In *2014 ieee international congress on big data* (pp. 796–799). IEEE. doi:10 . 1109 / BigData . Congress.2014.127

Hopkins, B., Owens, L., & Keenan, J. (2013). *The Patterns Of Big Data: A Data Management Playbook Toolkit*. Forrester Research. Retrieved from https://www.forrester.com/report/The+Patterns+Of+Big+Data/-/E-RES96101

Hornik, K. (2016). Frequently Asked Questions on R. Retrieved from https://cran.r-project.org/doc/FAQ/R-FAQ.html

Horton, N. J. & Kleinman, K. P. (2007). Much Ado About Nothing: A Comparison of Missing Data Methods and Software to Fit Incomplete Data Regression Models. *The American Statistician*, *61*(1). doi:10.1198/000313007X172556

Horvath, T. (2011). *Business Analytics - CRISP-DM*. University of Hildesheim. Retrieved from https://www.ismll.uni-hildesheim.de/lehre/ba-12ss/script/ba03.pdf

Hoving, R., Slot, G., & Jansen, S. (2013). Python: Characteristics identification of a free open source software ecosystem. In *7th ieee international conference on digital ecosystems and technologies (dest)* (pp. 13–18). doi:10.1109/DEST.2013.6611322

Humble, C. (2012). Oracle and the Java Ecosystem. Retrieved from https://www.infoq.com/articles/oracle-java-ecosystem

Hutter, F., Lücke, J., & Schmidt-Thieme, L. (2015). Beyond Manual Tuning of Hyperparameters. *KI-Künstliche Intelligenz*, *29*(4). doi:10.1007/s13218-015-0381-0

Iansiti, M. & Levien, R. (2004). *The keystone advantage: what the new dynamics of business ecosystems mean for strategy, innovation, and sustainability*. Harvard Business Press. Retrieved from http://www.hbs.edu/faculty/Pages/item.aspx?num=16920

Ihaka, R. (2009). *The R Project: A Brief History and Thoughts About the Future*. The University of Auckland. Retrieved from https://www.stat.auckland.ac.nz/~ihaka/downloads/Massey.pdf

Ihaka, R. & Gentleman, R. (1996). R: A Language for Data Analysis and Graphics. *Journal of Computational and Graphical Statistics*, *5*(3), 299–314. Retrieved from http://www.jstor.org/stable/1390807

Inventado, P. S. & Scupelli, P. (2015). Data-driven Design Pattern Production: A Case Study on the ASSISTments Online Learning System. *Proceedings of the 20th European Conference on Pattern Languages of Programs*, 1–13. doi:10.1145/2855321.2855336

Inventado, P. S. & Scupelli, P. (2016). Design Patterns for Math Problems and Learning Support in Online Learning Systems. In *Proceedings of the 10th travelling conference on pattern languages of programs - vikingplop* (pp. 1–16). doi:10.1145/3022636.3022644

Jagadish, H. V., Gehrke, J., Labrinidis, A., Papakonstantinou, Y., Patel, J. M. J., Ramakrishnan, R., & Shahabi, C. (2014). Big Data and Its Technical Challenges. *Communications of the ACM*, *57*(7), 86–94. doi:10.1145/2611567

Jain, A. K. (2010). Data clustering: 50 years beyond K-means. *Pattern Recognition Letters*, *31*(8). doi:10.1016/j.patrec.2009.09.011

Jansen, H. (2010). The logic of qualitative survey research and its position in the field of social research methods. doi:10.17169/fqs-11.2.1450

Jansen, S. (2014). Measuring the health of open source software ecosystems: Beyond the scope of project health. *Information and Software Technology*, *56*(11), 1508–1519. doi:10 . 1016/j.infsof.2014.04.006

Jifa, G. & Lingling, Z. (2014). Data, DIKW, Big Data and Data Science. *Procedia Computer Science*, *31*, 814–821. doi:10.1016/j.procs.2014.05.332

Joly, S. (2016). A Brief History of Business Intelligence | BI Software Insight. Retrieved from https://www.pyramidanalytics.com/blog/business-intelligence-history

Joshua, J., Alao, D., Okolie, S., & Awodele, O. (2013). Software Ecosystem: Features, Benefits and Challenges. *International Journal of Advanced Computer Science and Applications*, *4*(8), 11. doi:10.14569/IJACSA.2013.040833

Jurney, R. (2013). *Agile Data Science: Building Data Analytics Applications with Hadoop*. O'Reilly Media. Retrieved from http://shop.oreilly.com/product/0636920025054.do

Kabbedijk, J. & Jansen, S. (2011). Steering Insight: An Exploration of the Ruby Software Ecosystem. In *Software business: Second international conference, icsob* (pp. 44–55). doi:10.1007%2F978-3-642-21544-5_5

Kane, M. J., Emerson, J., & Weston, S. (2013). Scalable Strategies for Computing with Massive Data. *Journal of Statistical Software*, *55*(14), 1–19. Retrieved from http://www.jstatsoft.org/v55/i14/

Karpievitch, Y. V., Dabney, A. R., & Smith, R. D. (2012). Normalization and missing value imputation for label-free LC-MS analysis. *BMC Bioinformatics*, *13*(16). doi:10.1186/1471-2105-13-S16-S5

Kimball, R., Ross, M., Thornthwaite, W., Mundy, J., & Becker, B. (2008). *The Data Warehouse Lifecycle Toolkit* (2nd). Wiley. Retrieved from http://dl.acm.org/citation.cfm?id=1571714

Kimble, C. & Milolidakis, G. (2015). Big Data and Business Intelligence: Debunking the Myths. *Global Business and Organizational Excellence*, *35*(1), 23–34. doi:10.1002/joe.21642

Klakegg, O. J. (2015). Ontology and Epistomology. In B. Pasian (Ed.), *Designs, methods and practices for research of project management* (Chap. 5, p. 520). Routledge. doi:10.4324/9781315270197

Klein, B. (2018). History of Python. Retrieved from http://www.python-course.eu/python3_history_and_philosophy.php

Kohavi, R. (1995). A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection. In *Proceedings of the 14th international joint conference on artificial intelligence - volume 2* (pp. 1–7). Retrieved from https://dl.acm.org/citation.cfm?id=1643047

Kolfschoten, G., Lukosch, S., Verbraeck, A., Valentin, E., & Vreede, G. J. d. (2010). Cognitive learning efficiency through the use of design patterns in teaching. *Computers and Education*, *54*(3), 652–660. doi:10.1016/j.compedu.2009.09.028

Koren, Y. (2009). The bellkor solution to the netflix grand prize, 1–10. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.162.2118

Koziolek, H. (2008). Goal, question, metric. In I. Eusgeld, F. Freiling, & R. Reussner (Eds.), *Dependability metrics - lecture notes in computer science* (Vol. 4909, pp. 39–42). Springer. doi:10.1007/978-3-540-68947-8_6

Krawatzeck, R., Zimmer, M., & Trahasch, S. (2013). Agile Business Intelligence - Definition, Maßnahmen und Herausforderungen. *HMD Praxis der Wirtschaftsinformatik*, *50*(2), 56–63. doi:10.1007/BF03340796

Krill, P. (2016). Enterprise repo wars: GitHub vs. GitLab vs. Bitbucket. Retrieved from https://www.infoworld.com/article/3123244/application-development/enterprise-repo-wars-github-vs-gitlab-vs-bitbucket.html

Krishnakumar, V. (2011). Types of Research. Retrieved from https://www.slideshare.net/vaisalik/types-of-research

Kuhn, M. & Johnson, K. (2013). Applied predictive modeling. *Applied Predictive Modeling*, 1–600. doi:10.1007/978-1-4614-6849-3

Landset, S., Khoshgoftaar, T. M., Richter, A. N., & Hasanin, T. (2015). A survey of open source tools for machine learning with big data in the Hadoop ecosystem. *Journal of Big Data*, *2*(1), 24. doi:10.1186/s40537-015-0032-1

Larson, D. & Chang, V. (2016). A review and future direction of agile, business intelligence, analytics and data science. *International Journal of Information Management*, *36*(5), 700–710. doi:10.1016/j.ijinfomgt.2016.04.013

Lavrač, N., Motoda, H., Fawcett, T., Holte, R., Langley, P., & Adriaans, P. (2004). Introduction: Lessons Learned from Data Mining Applications and Collaborative Problem Solving. *Machine Learning*, *57*(1-2). doi:10.1023/B:MACH.0000035516.74817.51

Leal, A. G. & Boldt, M. (2016). A big data analytics design patterns to select customers for electricity theft inspection. In *Transmission & distribution conference and exposition-latin america, 2016 ieee pes* (pp. 1–6). doi:10.1109/TDC-LA.2016.7805663

Lee, C.-H. (2015). How to Choose Between Learning Python or R First. Retrieved from https://blog.udacity.com/2015/01/python-vs-r-learn-first.html

Li, Y. R. (2009). The technological roadmap of Cisco's business ecosystem. *Technovation*, *29*(5), 379–386. doi:10.1016/j.technovation.2009.01.007

Lim, A. & Tjhi, W. (2015). *R High Performance Programming*. Packt Publishing. Retrieved from https://www.packtpub.com/application-development/r-high-performance-programming

Little, R. J. A. & Rubin, D. B. (2002). *Statistical Analysis with Missing Data* (2nd). Wiley. Retrieved from https://www.wiley.com/en-us/Statistical+Analysis+with+Missing+Data%2C+2nd+Edition-p-9780471183860

Liu, S., Maljovec, D., Wang, B., Bremer, P.-T., & Pascucci, V. (2015). Visualizing High-Dimensional Data: Advances in the Past Decade. *Eurographics Conference on Visualization (EuroVis) - STARs*. doi:10.2312/eurovisstar.20151115

Liu, X., Iftikhar, N., & Xie, X. (2014). Survey of Real-time Processing Systems for Big Data. *Proceedings of the 18th International Database Engineering & Applications Symposium*, 356–361. doi:10.1145/2628194.2628251

Long, T. & Johnson, M. (2000). Rigour, reliability and validity in qualitative research. *Clinical Effectiveness in Nursing*, *4*(1), 30–37. doi:10.1054/cein.2000.0106

Loukides, M. (2011). *What is data science?* O'Reilly Media. Retrieved from https://www.oreilly.com/ideas/what-is-data-science

Luhn, H. P. (1958). A Business Intelligence System. *IBM Journal of Research and Development*, *2*(4), 314–319. doi:10.1147/rd.24.0314

Lutz, M. (2013). *Learning Python* (5th). O'Reilly Media. Retrieved from http://shop.oreilly.com/product/0636920028154.do

Ma, D. (2016). R vs Python for Data Science: Summary of Modern Advances. Retrieved from https://elitedatascience.com/r-vs-python-for-data-science

Maaten, L. v. d., Postma, E., & Herik, J. v. d. (2009). *Dimensionality reduction: A comparative review*. Technical Report TiCC TR 2009-005. Retrieved from https://www.tilburguniversity.edu/upload/59afb3b8-21a5-4c78-8eb3-6510597382db_TR2009005.pdf

Maclin, R. & Opitz, D. (1999). Popular Ensemble Methods: An Empirical Study. *Journal Of Artificial Intelligence Research*, *11*. doi:10.1613/jair.614

Maechler, M. (2018). [Rd] Fixed BLAS tests for external BLAS library. Retrieved from https://stat.ethz.ch/pipermail/r-devel/2018-January/075343.html

Maechler, M. & Bates, D. (2018). Matrix: Sparse and Dense Matrix Classes and Methods. Retrieved from https://cran.r-project.org/package=Matrix

Malina, M. A., Nørreklit, H. S., & Selto, F. H. (2011). Lessons learned: advantages and disadvantages of mixed method research. *Qualitative Research in Accounting & Management*, *8*(1), 59–71. doi:10.1108/11766091111124702

Manikas, K. (2016). Revisiting software ecosystems Research: A longitudinal literature study. *Journal of Systems and Software*, *117*, 84–103. doi:10.1016/j.jss.2016.02.003

Manikas, K. & Hansen, K. M. (2013). Software ecosystems - A systematic literature review. *Journal of Systems and Software*, *86*(5), 1294–1306. doi:10.1016/j.jss.2012.12.026

Mao, S., Chen, M., & Liu, Y. (2014). Big Data: A Survey. *Mobile Networks and Applications*, *19*(2), 171–209. doi:10.1007/s11036-013-0489-0

Marston, S., Li, Z., Bandyopadhyay, S., Zhang, J., & Ghalsasi, A. (2011). Cloud computing - The business perspective. *Decision Support Systems*, *51*(1), 176–189. doi:10.1016/J.DSS.2010.12.006

Matavire, R. & Brown, I. (2011). Profiling grounded theory approaches in information systems research. *European Journal of Information Systems*, *22*(1), 119–129. doi:10.1057/ejis.2011.35

Mattson, T. G., Sanders, B., & Massingill, B. (2004). *Patterns for Parallel Programming*. Pearson Education. Retrieved from https://dl.acm.org/citation.cfm?id=1406956

McAfee, A. & Brynjolfsson, E. (2012). Big Data: The Management Revolution. *Harvard business review*, (10). Retrieved from https://hbr.org/2012/10/big-data-the-management-revolution

McCleary, L. (2002). Using multiple imputation for analysis of incomplete data in clinical research. *Nursing Research*, *51*(5), 339–343. doi:10.1097/00006199-200209000-00012

McClure, S. (2015). Data Science and Big Data: Two very Different Beasts. Retrieved from http://www.kdnuggets.com/2015/07/data-science-big-data-different-beasts.html

McConville, K. (2017). Teaching the Tidyverse in the Second Semester, Undergraduate Statistics Course. Swarthmore College. Retrieved from https://www.user2017.brussels/uploads/mcconville_user2017_Poster_2017_06_16.pdf

Mens, T. & Grosjean, P. (2014). ECOS: Ecological Studies of Open Source Software Ecosystems. In *Ieee conference on software maintenance, reengineering, and reverse engineering* (pp. 403–406). doi:10.1109/CSMR-WCRE.2014.6747205

Merritt-Holmes, M. (2016). 10 differences between Data Science and Business Intelligence. Retrieved from http://www.itproportal.com/2016/08/18/10-differences-between-data-science-and-business-intelligence/

Meszaros, G. & Doble, J. (1997). MetaPatterns: A Pattern Language for Pattern Writing. *Pattern languages of program design*, 1–36. Retrieved from http://xunitpatterns.com/~gerard/plopd3-pattern-writing-patterns-paper.pdf

Microsoft. (2009). *Microsoft Application Architecture Guide*. Patterns & Practices. Microsoft Press. Retrieved from https://msdn.microsoft.com/en-us/library/ee658109.aspx

Mikut, R. & Reischl, M. (2011). Data mining tools. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, *1*(5), 431–443. doi:10.1002/widm.24

Millman, K. J. & Aivazis, M. (2011). Python for scientists and engineers. *Computing in Science and Engineering*, *13*(2), 9–12. doi:10.1109/MCSE.2011.36

Min, J. & Lee, Y. (2005). Bankruptcy prediction using support vector machine with optimal choice of kernel function parameters. *Expert systems with applications*, *28*(4), 603–614. doi:10.1016/j.eswa.2004.12.008

Miner, D. & Shook, A. (2012). *MapReduce Design Patterns*. O'Reilly Media. Retrieved from http://shop.oreilly.com/product/0636920025122.do

Mittag, N. (2013). *Imputations: Benefits, Risks and a Method for Missing Data*, CERGE-EI. Retrieved from http://home.cerge-ei.cz/mittag/papers/Imputations.pdf

Montanaro, S. (2012). Why is Python a dynamic language and also a strongly typed language. Retrieved from https://wiki.python.org/moin/Why%20is%20Python%20a%20dynamic%20language%20and%20also%20a%20strongly%20typed%20language

Moore, J. F. (1999). *The Death of Competition - Leadership & Strategy in the Age of Business Ecosystems*. Harper Paperbacks. Retrieved from https://www.harpercollins.com/9780887308505/the-death-of-competition

Morley, T. (2019). *Data Science Design Patterns*. Pearson Professional. Retrieved from http://www.pearsoned.co.nz/9780134000053

Mosaic. (2014). Mosaic Data Science Resources | Data Science Design Patterns. Retrieved from http://www.mosaicdatascience.com/resources/data-science-design-patterns/

Mount, J. (2014). Factors are not first-class citizens in R. Retrieved from http://www.win-vector.com/blog/2014/09/factors-are-not-first-class-citizens-in-r/

Muenchen, R. A. (2017). Why R is Hard to Learn. Retrieved from http://r4stats.com/articles/why-r-is-hard-to-learn/

Muenchen, R. A. (2018). The Popularity of Data Science Software. Retrieved from http://r4stats.com/articles/popularity/

Myers, M. D. (1997). Qualitative Research in Information Systems. *MISQ Discovery*, *21*(2), 241–242. Retrieved from http://www.qual.auckland.ac.nz/

Mysore, D., Shrikant, K., & Jain, S. (2013). Big data architecture and patterns. Retrieved from http://www.ibm.com/developerworks/library/bd-archpatterns1/

Nadschlager, S. (2017). Introducing Design Patterns to Knowledge Processing Systems in the Context of Big Data and Cloud Platforms. In *28th international workshop on database and expert systems applications* (pp. 47–51). doi:10.1109/DEXA.2017.26

Nagi, S. & Bhattacharyya, D. K. (2013). Classification of microarray cancer data using ensemble approach. *Network Modeling Analysis in Health Informatics and Bioinformatics*, *2*(3), 159–173. doi:10.1007/s13721-013-0034-x

Negash, S. & Gray, P. (2008). Business Intelligence. In *Handbook on decision support systems 2: Variations* (pp. 175–193). Springer. doi:10.1007/978-3-540-48716-6_9

Nguyen, H. & Hyde, P. V. D. (2015). Talk Data to Me: Data Visualization Best Practices. Tableau. Retrieved from https://www.youtube.com/watch?v=GnMSjSWDQNk

Nichols, T. E., Das, S., Eickhoff, S. B., Evans, A. C., Glatard, T., Hanke, M., … Yeo, B. T. (2017). Best practices in data analysis and sharing in neuroimaging using MRI. *Nature Neuroscience*, *20*(3), 299–303. doi:10.1038/nn.4500

Norvig, P. (1996). *Design patterns in dynamic programming*. Harlequin Inc. Retrieved from http://www.norvig.com/design-patterns/design-patterns.pdf

Noy, C. (2008). Sampling Knowledge: The Hermeneutics of Snowball Sampling in Qualitative Research. *International Journal of Social Research Methodology*, *11*(4), 327–344. doi:10.1080/13645570701401305

Okoli, C. & Schabram, K. (2011). *A Guide to Conducting a Systematic Literature Review of Information Systems Research*. Retrieved from https://papers.ssrn.com/sol3/papers.cfm?abstract_id=1954824

Oliphant, T. E. (2007). Python for scientific computing. *Computing in Science and Engineering*, *9*(3), 10–20. doi:10.1109/MCSE.2007.58

Oliveira, T. & Martins, M. (2011). Literature Review of Information Technology Adoption Models at Firm Level. *Electronic Journal of Information Systems Evaluation*, *14*(1), 110–121. Retrieved from http://www.ejise.com/issue/download.html?idArticle=705

Olsen, R. (2007). *Design Patterns in Ruby*. Addison-Wesley Professional. Retrieved from https://dl.acm.org/citation.cfm?id=1349728

Olson, D. & Delen, D. (2008). *Advanced data mining techniques*. Springer. doi:10.1007/978-3-540-76917-0

O'Neil, C. & Schutt, R. (2013). *Doing Data Science*. O'Reilly Media. Retrieved from http://shop.oreilly.com/product/0636920028529.do

Opara-Martins, J., Sahandi, R., & Tian, F. (2015). Critical review of vendor lock-in and its impact on adoption of cloud computing. In *International conference on information society* (pp. 92–97). doi:10.1109/i-Society.2014.7009018

Orlikowski, W. J. & Scott, S. V. (2008). Sociomateriality: Challenging the Separation of Technology, Work and Organization. *The Academy of Management Annals*, *2*(1), 433–474. doi:10.1080/19416520802211644

Ousterhout, J. (1998). Scripting: higher level programming for the 21st Century. *Computer*, *31*(3), 23–30. doi:10.1109/2.660187

Pabinger, S., Dander, A., Fischer, M., Snajder, R., Sperk, M., Efremova, M., … Trajanoski, Z. (2014). A survey of tools for variant analysis of next-generation genome sequencing data. *Briefings in Bioinformatics*, *15*(2), 256–278. doi:10.1093/bib/bbs086

Pahl, C. (2015). Containerization and the PaaS Cloud. *IEEE Cloud Computing*, *2*(3), 24–31. doi:10.1109/MCC.2015.51

Pan, D. (1998). *The application of design patterns in knowledge inference engine*. University of Calgary. Retrieved from https://prism.ucalgary.ca/bitstream/1880/26071/1/34985Pan.pdf

Patil, D. (2013). Stanford Seminar - The Things I Wish I Knew - Lessons Learned from Making Data Product. Retrieved from https://www.youtube.com/watch?v=YqQ86NDeGpo%0A

Patton, M. Q. (2015). *Qualitative evaluation and research methods* (4th). SAGE Publications. Retrieved from https://us.sagepub.com/en-us/nam/qualitative-research-evaluation-methods/book232962

Pavlopoulos, G. A., Wegener, A.-L., & Schneider, R. (2008). A survey of visualization tools for biological network analysis. *BioData Mining*, *1*(1), 12. doi:10.1186/1756-0381-1-12

Peltoniemi, M. & Vuori, E. (2004). Business ecosystem as the new approach to complex adaptive business environments. In *Proceedings of ebusiness research forum* (Vol. 2, pp. 267–281). Retrieved from https://www.researchgate.net/publication/228985086_Business_Ecosystem_as_the_New_Approach_to_Complex_Adaptive_Business_Environments

Peng, R. (2015). stringsAsFactors: An unauthorized biography. Retrieved from https://simplystatistics.org/2015/07/24/stringsasfactors-an-unauthorized-biography/

Perez, C. E. (2018). *Deep Learning Patterns*. Retrieved from http://www.deeplearningpatterns.com/doku.php?id=start

Perez, F., Granger, B. E., & Hunter, J. D. (2011). Python: An Ecosystem for Scientific Computing. *Computing in Science and Engg. 13*(2), 13–21. doi:10.1109/MCSE.2010.119

Persson, S. (2004). *Qualitative Methods in Software Engineering*, Lund University. Retrieved from http://fileadmin.cs.lth.se/serg/old-serg-dok/docs-masterthesis/59_Rep.5520.Persson.pdf

Peters, T. (2004). The Zen of Python. Retrieved from https://www.python.org/dev/peps/pep-0020/

Petrov, D. (2016). *A literature study on barriers organizations face when adopting open-source software*. Aarhus Univeristy. Retrieved from https://dmpe.github.io/PapersAndArticles/LiteratureStudyOnOpenSourceSoftware/

Plakidas, K., Schall, D., & Zdun, U. (2017). Evolution of the R software ecosystem: Metrics, relationships, and their impact on qualities. *Journal of Systems and Software*, *132*, 119–146. doi:10.1016/j.jss.2017.06.095

Plakidas, K., Stevanetic, S., Schall, D., Ionescu, T. B., & Zdun, U. (2016). How do software ecosystems evolve? A quantitative assessment of the R ecosystem. In *Proceedings of the 20th international systems and software product line* (pp. 89–98). doi:10.1145/2934466.2934488

Poslek, K. (2015). Overview of the JavaScript ecosystem - An introduction to the vast world of JavaScript. Retrieved from https://medium.com/@bojzi/overview-of-the-javascript-ecosystem-8ec4a0b7a7be

Provost, F. & Fawcett, T. (2013a). Data Science and its Relationship to Big Data and Data-Driven Decision Making. *Big Data*, *1*(1), 51–59. doi:10.1089/big.2013.1508

Provost, F. & Fawcett, T. (2013b). *Data Science for Business: What You Need to Know about Data Mining and Data-Analytic Thinking*. O'Reilly Media. Retrieved from http://shop.oreilly.com/product/0636920028918.do

Pukhovskaya, A. (2014). Green Issues in Education Research. *International Journal of Sales, Retailing and Marketing*, *3*(2). Retrieved from http://www.ijsrm.com/ijsrm/Current_&_Past_Issues_files/IJSRM3-2.pdf

Python Core Team. (2018a). 2to3 - Automated Python 2 to 3 code translation. Retrieved from https://docs.python.org/2.7/library/2to3.html

Python Core Team. (2018b). Graphic User Interface FAQ. Retrieved from https://docs.python.org/3/faq/gui.html

Python Core Team. (2018c). Python: A dynamic, open source programming language. Python Software Foundation. Retrieved from https://www.python.org/

Python Core Team. (2018d). The Python Language Reference. Retrieved from https://docs.python.org/3/reference/index.html
Python Core Team. (2018e). The Python Standard Library. Retrieved from https://docs.python.org/3/library/index.html
R Core Team. (2018a). An Introduction to R. Retrieved from ftp://cran.r-project.org/pub/R/doc/manuals/r-release/R-intro.html
R Core Team. (2018b). R: A Language and Environment for Statistical Computing. Vienna, Austria. Retrieved from http://www.r-project.org/
R Core Team. (2018c). R Language Definition. Retrieved from ftp://cran.r-project.org/pub/R/doc/manuals/r-release/R-lang.html
R Core Team. (2018d). What is R? Retrieved from https://www.r-project.org/about.html
Ramalho, L. (2015). *Fluent Python: Clear, Concise, and Effective Programming*. O'Reilly Media. Retrieved from http://shop.oreilly.com/product/0636920032519.do
Raschka, S., Julian, D., & Hearty, J. (2016a). *Python: Deeper Insights into Machine Learning*. Packt Publishing. Retrieved from https://www.packtpub.com/big-data-and-business-intelligence/python-deeper-insights-machine-learning
Raschka, S., Julian, D., & Hearty, J. (2016b). *Python: Deeper Insights into Machine Learning*. Packt Publishing. Retrieved from https://www.packtpub.com/big-data-and-business-intelligence/python-deeper-insights-machine-learning
Refaeilzadeh, P., Tang, L., & Liu, H. (2017). *Cross-Validation - Encyclopedia of Database Systems* (L. L. & Ö. M.T., Eds.). Springer. doi:10.1007/978-0-387-39940-9
Rickert, J. (2010). Learning R. Revolutions. Retrieved from http://blog.revolutionanalytics.com/2010/06/learning-r.html
Rickert, J. (2017). What is the tidyverse? R Views. Retrieved from https://rviews.rstudio.com/2017/06/08/what-is-the-tidyverse/
Robinson, D. (2017). The Impressive Growth of R. Retrieved from https://stackoverflow.blog/2017/10/10/impressive-growth-r/
Roe, C. (2013). The Emergence of Data Science: Data Management's New Pioneers. Retrieved from http://www.dataversity.net/the-emergence-of-data-science-data-managements-new-pioneers/
Ross, Z., Wickham, H., & Robinson, D. (2017). Declutter your R workflow with tidy tools. *PeerJ Preprints*. doi:10.7287/peerj.preprints.3180v1
Rossum, G. v. (2009). A Brief Timeline of Python. Retrieved from http://python-history.blogspot.de/2009/01/brief-timeline-of-python.html
Rostcheck, D. (2016a). Data science as a professional career. Retrieved from https://www.slideshare.net/DavidRostcheck/data-science-as-a-professional-career
Rostcheck, D. (2016b). Data Science? Business Intelligence? What's the difference? Retrieved from https://www.linkedin.com/pulse/data-science-business-intelligence-whats-difference-david-rostcheck
Rowley, J. (2007). The wisdom hierarchy: representations of the DIKW hierarchy. *Journal of information science*, *33*(2), 163–180. doi:10.1177/0165551506070706
Saint Joseph's University. (2017). Business Intelligence Analyst vs. Data Scientist: Understanding the Difference. Retrieved from http://online.sju.edu/graduate/masters-business-intelligence/resources/articles/data-science-or-business-intelligence

Salingaros, N. (2018). Some Notes on Christopher Alexander. University of Texas at San Antonio. Retrieved from http://zeta.math.utsa.edu/~yxk833/Chris.text.html

Samani, A., Oliveros, S., & Huang, C. (2016). Working Effectively in Data Science Teams. Retrieved from https://svds.com/working-effectively-in-data-science-teams/

Samuel, A. L. (1959). Some Studies in Machine Learning Using the Game of Checkers. *IBM Journal of Research and Development*, *3*(3). doi:10.1147/rd.441.0206

Sandelowski, M. (1995). Sample size in qualitative research. *Research in Nursing & Health*, *18*(2), 179–183. doi:10.1002/nur.4770180211

Sanner, M. F. (1999). Python: a programming language for software integration and development. *Journal of Molecular Graphics and Modelling*, *17*(1), 57–61. Retrieved from https://www.ncbi.nlm.nih.gov/pubmed/10660911

Saunders, M., Lewis, P., & Thornhill, A. (2015). *Research Methods for Business Students* (7th). Pearson. Retrieved from http://catalogue.pearsoned.co.uk/educator/product/Research-Methods-for-Business-Students/9781292016627.page

Schafer, J. L. & Graham, J. W. (2002). Missing data: Our view of the state of the art. *Psychological Methods*, *7*(2), 147–177. doi:10.1037/1082-989X.7.2.147

Schmarzo, B. (2014). Business Intelligence Analyst or Data Scientist? What's the Difference? Retrieved from https://blog.dellemc.com/en-us/business-intelligence-analyst-data-scientist-whats-difference/

Schmidt, D. C., Fayad, M., & Johnson, R. E. (1996). Software Patterns. *Commun. ACM*, *39*(10), 37–39. doi:10.1145/236156.236164

Sculley, D., Holt, G., Golovin, D., Davydov, E., Phillips, T., Ebner, D., … Dennison, D. (2015). Hidden Technical Debt in Machine Learning Systems. In *Advances in neural information processing systems* (pp. 1–9). Retrieved from https://dl.acm.org/citation.cfm?id=2969519

Seaman, C. (2013). Using Qualitative Methods in Empirical Studies of Software Engineering. Retrieved from http://ccsl.ime.usp.br/files/QualMethods%20minicourse%20USP-1.pdf

Seaman, C. (1999). Qualitative methods in empirical studies of software engineering. *IEEE Transactions on Software Engineering*, *25*(4), 557–572. doi:10.1109/32.799955

Segel, E. & Heer, J. (2010). Narrative visualization: Telling stories with data. *IEEE Transactions on Visualization and Computer Graphics*, *16*(6), 1139–1148. doi:10.1109/TVCG.2010.179

Shah, F. (2014). The History of BI: The 2000's and Now. Retrieved from http://dataconomy.com/2014/07/the-history-of-bi-the-2000s-and-now/

Shan, C., Wang, H., Chen, W., & Song, M. (2015). *The Data Science Handbook: Advice and Insights from 25 Amazing Data Scientists*. The Data Science Bookshelf. Retrieved from http://www.thedatasciencehandbook.com/

Shollo, A. & Galliers, R. D. (2016). Towards an understanding of the role of business intelligence systems in organisational knowing. *Information Systems Journal*, *26*(4), 339–367. doi:10.1111/isj.12071

Small, M. L. (2011). How to Conduct a Mixed Methods Study: Recent Trends in a Rapidly Growing Literature. *Annual Review of Sociology*, *37*(1), 57–86. doi:10.1146/annurev.soc.012809.102657

Smith, D. (2009). Batch mode in R: a primer. Revolutions. Retrieved from http://blog.revolutionanalytics.com/2009/06/batch-mode-in-r-a-primer.html

Smith, D. (2011). How does Data Science impact Business Intelligence? Revolutions. Retrieved from http://blog.revolutionanalytics.com/2011/05/how-does-data-science-impact-business-intelligence.html

Smith, D. (2013a). A detailed guide to memory usage in R. Revolutions. Retrieved from http://blog.revolutionanalytics.com/2013/11/a-detailed-guide-to-memory-usage-in-r.html

Smith, D. (2013b). Statistics vs Data Science vs BI. Revolutions. Retrieved from http://blog.revolutionanalytics.com/2013/05/statistics-vs-data-science-vs-bi.html

Smith, D. (2016). Over 16 years of R Project history. Revolutions. Retrieved from http://blog.revolutionanalytics.com/2016/03/16-years-of-r-history.html

Smith, D. (2017). CRAN now has 10,000 R packages. Here's how to find the ones you need. Revolutions. Retrieved from http://blog.revolutionanalytics.com/2017/01/cran-10000.html

Smith, J. W., Everhart, J., Dickson, W., Knowler, W., & Johannes, R. (1988). Using the ADAP Learning Algorithm to Forecast the Onset of Diabetes Mellitus. *Proceedings of the Annual Symposium on Computer Application in Medical Care*, 261–265. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2245318/ and https://www.kaggle.com/uciml/pima-indians-diabetes-database.

Spinellis, D. (2001). Notable design patterns for domain-specific languages. *Journal of Systems and Software*, *56*(1), 91–99. doi:10.1016/S0164-1212(00)00089-3

Stol, K.-J. & Ali Babar, M. (2010). Challenges in Using Open Source Software in Product Development: A Review of the Literature. In *Proceedings of the 3rd international workshop on emerging trends in free/libre/open source software research and development* (pp. 17–22). doi:10.1145/1833272.1833276

Stolberg, S. (2009). Enabling agile testing through continuous integration. In *Agile conference* (pp. 369–374). doi:10.1109/AGILE.2009.16

Stone, A., Takabi, H., Joshi, J. B. D., & Ahn, G.-J. (2010). Security and privacy challenges in cloud computing environments. *IEEE Security & Privacy*, *8*(6), 24–31. doi:10.1109/MSP.2010.186

Stufft, D. (2016). Publicly Queryable Statistics. Retrieved from https://mail.python.org/pipermail/distutils-sig/2016-May/028986.html

Suri, H. (2011). Purposeful Sampling in Qualitative Research Synthesis. *Qualitative Research Journal*, *11*(2), 63–75. doi:10.3316/QRJ1102063

Sutton, C. (2012). Principles of Research Code. Retrieved from http://www.theexclusive.org/2012/08/principles-of-research-code.html

Swanson, I. (2016). Data Scientists vs. BI Analysts: What's the Difference? Retrieved from http://www.cmswire.com/analytics/data-scientists-vs-bi-analysts-whats-the-difference/

Syed, S. & Jansen, S. (2013). On clusters in open source ecosystems. In *5th international workshop on software ecosystems* (Vol. 13). Retrieved from http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.416.322

Symonds, J. E. & Gorard, S. (2010). The death of mixed methods: research labels and their casualties. *The British Educational Research Association*, 1–19. doi:10.1080/09500790.2010.483514

Taft, D. K. (2015). One-Third of BI Pros Spend Up to 90% of Time Cleaning Data. Retrieved from http://www.eweek.com/database/one-third-of-bi-pros-spend-up-to-90-of-time-cleaning-data

Takahashi, M. & Ito, T. (2012). Multiple Imputation of Turnover in EDINET Data: Toward the Improvement of Imputation for the Economic Census. *Work Session on Statistical Data Editing, UNECE*. Retrieved from https://www.unece.org/fileadmin/DAM/stats/documents/ece/ces/ge.44/2012/35_Japan.pdf

Teddlie, C. & Yu, F. (2007). Mixed Methods Sampling: A Typology with Examples. *Journal of Mixed Methods Research*, *1*(1), 77–77. doi:10.1177/2345678906292430

Theuwissen, M. (2015). R vs Python for Data Science: The Winner is …. Retrieved from http://www.kdnuggets.com/2015/05/r-vs-python-data-science.html

Theuwissen, M. (2016). R or Python? Consider learning both. Retrieved from http://www.kdnuggets.com/2016/03/r-python-learning-both-datacamp.html

Thomas, D. R. (2006). A General Inductive Approach for Analyzing Qualitative Evaluation Data. *American Journal of Evaluation*, *27*(2), 237–246. doi:10.1177/1098214005283748

Thornthwaite, W. & Ginnebaugh, M. (2012). Microsoft Data Warehouse Business Intelligence Lifecycle - The Kimball Approach. Retrieved from https://www.slideshare.net/markginnebaugh/microsoft-data-warehouse-business-intelligence-lifecycle-kimball

Tippmann, S. (2014). Programming tools: Adventures with R. *Nature*, *517*(7532), 109–110. doi:10.1038/517109a

Trochim, W. M. K. (2006). Deduction and Induction | Qualitative Data. Retrieved from https://www.socialresearchmethods.net/

Trujillo, J. & Maté, A. (2012). Business intelligence 2.0: A general overview. In *Lecture notes in business information processing 96* (pp. 98–116). doi:10.1007/978-3-642-27358-2_5

Trujillo, J., Palomar, M., Gomez, J., & Song, I. Y. (2001). Designing data warehouses with OO conceptual models. *Computer*, *34*(12). doi:10.1109/2.970579

Tutfe, E. (1983). *The Visual Display of Quantitative Information*. Graphics Press. Retrieved from https://www.edwardtufte.com/tufte/books_vdqi

Tutunea, M. F. (2015). Business Intelligence Solutions for Mobile Devices - An Overview. *Procedia Economics and Finance*, *27*, 160–169. doi:10.1016/S2212-5671(15)00985-5

Ulrich, W. (2006). The art of observation: Understanding pattern languages. *Journal of Research Practice*, *2*(1), 9. Retrieved from http://jrp.icaap.org/index.php/jrp/article/view/26/46

VanderPlas, J. (2016). *Python Data Science Handbook*. O'Reilly Media. Retrieved from https://jakevdp.github.io/PythonDataScienceHandbook/index.html

Vanwinckelen, G. & Blockeel, H. (2012). On estimating model accuracy with repeated cross-validation. In *21st belgian-dutch conference on machine learning* (pp. 39–44). Retrieved from https://lirias.kuleuven.be/handle/123456789/346385

Vasconcelos, J. B. d. & Rocha, Á. (2017). Special section on data science and business intelligence. *International Journal of Information Management*, *37*(6), 716–717. doi:10.1016/j.ijinfomgt.2017.07.014

Venners, B. (2003). The Making of Python - A Conversation with Guido van Rossum. Retrieved from http://www.artima.com/intv/pythonP.html

Walkowiak, S. (2016). *Big Data Analytics with R*. Packt Publishing. Retrieved from https://www.packtpub.com/big-data-and-business-intelligence/big-data-analytics-r

Wall, P. (2018). Mixed methods research. Retrieved from http://resourcecentre.foodrisc.org/mixed-methods-research_185.html

Waller, M. A. & Fawcett, S. E. (2013). Data Science, Predictive Analytics, and Big Data: A Revolution That Will Transform Supply Chain Design and Management. *Journal of Business Logistics*, *34*(2), 77–84. doi:10.1111/jbl.12010

Walt, S. V. D., Colbert, S. C., & Varoquaux, G. (2011). The NumPy array: A structure for efficient numerical computation. *Computing in Science and Engineering*, *13*(2), 22–30. doi:10.1109/MCSE.2011.37

Ward, M. O., Grinstein, G., & Keim, D. (2010). *Interactive Data Visualization: Foundations, Techniques, and Applications*. CRC Press. Retrieved from https://www.crcpress.com/Interactive-Data-Visualization-Foundations-Techniques-and-Applications/Ward-Grinstein-Keim/p/book/9781482257373

Ware, C., Wright, W., & Pioch, N. (2013). Visual Thinking Design Patterns. https://ccom.unh.edu/vislab/VTDP_web_pages/VTDP_HomePage.html and https://uncharted.software/assets/visual-thinking-design-patterns.pdf.

Watson, H. J. & Wixom, B. H. (2007). The Current State of Business Intelligence. *Computer*, *40*(9), 96–99. doi:10.1109/MC.2007.331

Watson, H. J., Wixom, B. H., Hoffer, J. A., Anderson-Lehman, R., & Reynolds, A. M. (2006). Real-Time business intelligence: Best practices at continental airlines. *Information Systems Management*, *23*(1), 7–18. doi:10.1201/1078.10580530/45769.23.1.20061201/91768.2

Wayner, P. (2017). Python vs. R: The battle for data scientist mind share. Retrieved from http://www.infoworld.com/article/3187550/data-science/python-vs-r-the-battle-for-data-scientist-mind-share.html

Webster, J. & Watson, R. T. (2002). Analyzing the Past to Prepare for the Future: Writing a Literature Review. *MIS Quarterly*, *26*(2), xiii–xxiii. Retrieved from https://dl.acm.org/citation.cfm?id=2017162

Weisstein, E. W. (2018). Fractal. Retrieved from http://mathworld.wolfram.com/Fractal.html

Wellhausen, T. & Fiesser, A. (2011). How to write a pattern? In *Proceedings of the 16th european conference on pattern languages of programs - europlop* (pp. 1–9). doi:10.1145/2396716.2396721

Whitmore, J. (2016). Jupyter Notebook Best Practices for Data Science. Retrieved from https://www.svds.com/tbt-jupyter-notebook-best-practices-data-science/

Whitworth, B. (2009). A Brief Introduction to Sociotechnical Systems. *Encyclopedia of Information Science and Technology*, 394–400. doi:10.4018/978-1-60566-026-4.ch066

Wickham, H. (2013). The RStudio CRAN mirror. Retrieved from https://blog.rstudio.com/2013/06/10/rstudio-cran-mirror/

Wickham, H. (2014). Tidy Data. *Journal of Statistical Software*, *59*(10). doi:10.18637/jss.v059.i10

Widjaja, J. T. (2015). What is the difference between a data scientist and a business intelligence analyst? Retrieved from https://www.quora.com/What-is-the-difference-between-a-data-scientist-and-a-business-intelligence-analyst/answer/Jason-T-Widjaja

Wilkerson, T. (2016). CISB594 - Business Intelligence - Part 1. Retrieved from http://slideplayer.com/slide/7377905/

Wilson, G., Bryan, J., Cranston, K., Kitzes, J., Nederbragt, L., & Teal, T. K. (2017). Good enough practices in scientific computing. *PLoS Computational Biology*, *13*(6). doi:10.1371/journal.pcbi.1005510

Wilson, S. (2008). Patterns of Personal Learning Environments. *Interactive Learning Environments*, *16*(1), 17–34. doi:10.1080/10494820701772660

Witten, I. H., Frank, E., Hall, M. A., & Pal, C. J. (2016). *Data Mining: Practical Machine Learning Tools and Techniques* (4th). The Morgan Kaufmann Series in Data Management Systems. Morgan Kaufmann. Retrieved from https://www.cs.waikato.ac.nz/ml/weka/book.html

Wittern, E. (2016). An Analysis of the JavaScript Package Ecosystem npm. Retrieved from http://www.apiful.io/intro/2016/06/01/npm-analysis.html

Woźniak, M., Grańa, M., & Corchado, E. (2014). A survey of multiple classifier systems as hybrid systems. *Information Fusion*, *16*(1), 3–17. doi:10.1016/j.inffus.2013.04.006

Wujek, B., Hall, P., & Güneş, F. (2016). *Best Practices for Machine Learning Applications*. SAS. Retrieved from https://support.sas.com/resources/papers/proceedings16/SAS2360-2016.pdf

Yang, P., Hwa Yang, Y., B. Zhou, B., & Y. Zomaya, A. (2010). A Review of Ensemble Methods in Bioinformatics. *Current Bioinformatics*, *5*(4), 296–308. Retrieved from http://www.ingentaconnect.com/content/ben/cbio/2010/00000005/00000004/art00006

Zeutschler, T. (2016). IT Applications in Business Analytics. Hochschule Dusseldorf. Retrieved from https://wiwi.hs-duesseldorf.de/personen/thomas.zeutschler/Documents/HSD_W_ITAiBA_Zeutschler_SS2016_Lecture2_CRSIP_DM.pdf

Zhang, A., Fawaz, N., Ioannidis, S., & Montanari, A. (2012). Guess Who Rated This Movie: Identifying Users Through Subspace Clustering. In *28th conference on uncertainty in artificial intelligence*. Retrieved from https://arxiv.org/abs/1208.1544

Zhang, C. & Ma, Y. (2012). *Ensemble Machine Learning: Methods and Applications*. Springer. Retrieved from http://www.springer.com/de/book/9781441993250

Zhang, Q., Cheng, L., & Boutaba, R. (2010). Cloud computing: State-of-the-art and research challenges. *Journal of Internet Services and Applications*, *1*(1), 7–18. doi:10.1007/s13174-010-0007-6

Zhang, Z. (2016). Missing data imputation: focusing on single imputation. *Annals of translational medicine*, *4*(1), 9. doi:10.3978/j.issn.2305-5839.2015.12.38

Zinkevich, M. (2016). *Rules of Machine Learning: Best Practices for ML Engineering*. Retrieved from http://martin.zinkevich.org/rules_of_ml/rules_of_ml.pdf

Zumel, N. & Mount, J. (2014). *Practical data science with R*. Manning Publications. Retrieved from https://www.manning.com/books/practical-data-science-with-r