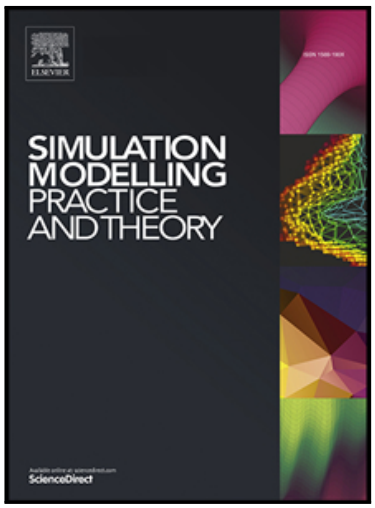
Quality of input data in emergency department simulations: Framework and assessment techniques

Peer-reviewed author version

VANBRABANT, Lien; MARTIN, Niels; RAMAEKERS, Katrien & BRAEKERS, Kris (2019) Quality of input data in emergency department simulations: Framework and assessment techniques. In: Simulation modelling practice and theory, 91, p. 83-101.

DOI: 10.1016/j.simpat.2018.12.002
Handle: http://hdl.handle.net/1942/27469



Quality of input data in emergency department simulations: framework and assessment techniques

Lien Vanbrabant, Niels Martin, Katrien Ramaekers, Kris Braekers

PII: S1569-190X(18)30183-7
DOI: https://doi.org/10.1016/j.simpat.2018.12.002
Reference: SIMPAT 1888

To appear in: *Simulation Modelling Practice and Theory*

Received date: 19 September 2018
Revised date: 30 November 2018
Accepted date: 3 December 2018

Please cite this article as: Lien Vanbrabant, Niels Martin, Katrien Ramaekers, Kris Braekers, Quality of input data in emergency department simulations: framework and assessment techniques, *Simulation Modelling Practice and Theory* (2018), doi: https://doi.org/10.1016/j.simpat.2018.12.002

Highlights

- The reliability of a complex ED simulation model depends on input data quality
- Electronic health record data used in ED simulations often contains quality issues
- Automated assessment techniques provide an efficient way to evaluate data quality
- Framework and assessment techniques are generalisable to other application domains
- Case study shows necessity and usefulness of our data quality assessment approach

Quality of input data in emergency department simulations: framework and assessment techniques

Lien Vanbrabant[a,b,*], Niels Martin[c], Katrien Ramaekers[a], Kris Braekers[a]

[a] *UHasselt, Research Group Logistics, Agoralaan Building D, 3590 Diepenbeek, Belgium*
[b] *Research Foundation Flanders (FWO), Egmontstraat 5, 1000 Brussel, Belgium*
[c] *UHasselt, Research Group Business Informatics, Agoralaan Building D, 3590 Diepenbeek, Belgium*

Abstract

Operations research techniques are widely used to analyse and optimise emergency department operations. The complex and stochastic nature of an emergency department makes simulation a suitable and frequently used technique. Simulation can provide valuable insights to hospital managers on how to improve the efficiency of an emergency department. However, the output of the simulation study is only as reliable as the input data used as basis for simulation modelling. As a result, high quality input data are essential for the construction of a realistic simulation model. This paper provides a data quality framework that categorises possible data quality problems in electronic healthcare records of emergency departments. Electronic healthcare records are a common source of input data for emergency department simulations, but often suffer from data quality issues. For the data quality problems identified in the framework, data quality assessment techniques are described. These techniques enable researchers and practitioners to identify and quantify the potential data quality issues present in input data. In order to facilitate data quality assessment, an implementation to automate this process is developed and applied to a real-life case study. This case study demonstrates the need for thorough and structured data quality assessment. Possible ways to deal with identified data quality problems are also described.

*Corresponding author
Email addresses: lien.vanbrabant@uhasselt.be (Lien Vanbrabant), niels.martin@uhasselt.be (Niels Martin), katrien.ramaekers@uhasselt.be (Katrien Ramaekers), kris.braekers@uhasselt.be (Kris Braekers)

Keywords: Data quality problems, Data quality assessment, Simulation, Emergency department, Electronic health records

1. Introduction

Emergency departments (EDs) are a crucial component in the healthcare system. They serve as one of the main access points to a hospital and provide treatment to a large variety of patients, with a major focus on critically ill patients [1]. Providing timely and high-quality care to these patients is the primary objective of an ED. However, EDs worldwide are faced with the problem of (over)crowding, which undermines their ability to fulfil these requirements [2]. Crowding is defined as a situation in which the demand for emergency services exceeds the capacity of the available resources in the ED [3]. Due to the risks associated to crowding, hospitals search for ways to alleviate the pressure on EDs [4, 5].

The challenge for EDs is to mitigate crowding by improving their efficiency without reducing the quality of care [6, 7]. Operations Research and Operations Management (OR/OM) techniques have been widely applied to model, analyse and optimise processes in healthcare organisations, including ED operations [8]. Because of the complex and stochastic environment, simulation techniques are frequently used and well-suited to investigate most healthcare systems. Examples of application areas for simulation in healthcare are outpatient clinics, intensive care units, EDs, inpatient units, and even whole hospital systems [9]. As a result, simulation is increasingly used to analyse ED operations and to examine possible process improvements [10].

A simulation study typically results in recommendations to hospital management on how to improve ED efficiency. These are only reliable if the simulation model is a good reflection of the actual ED operations [11]. Therefore, the basis of any simulation study is the construction of a realistic simulation model. When developing a simulation model, input data plays a critical role as it is used to determine the simulation model structure, key model parameters and for validation purposes [12, 13]. Given the 'garbage in, garbage out' principle, the quality of data used as input for the simulation model has a direct impact on the validity of the simulation study [13].

Input data collection and analysis is often identified as the most difficult, expensive and time-consuming part of a simulation study [14]. This is confirmed by Skoogh et al. [15], who state that input data collection and

analysis counts for up to 31% of the total simulation study time. Input data can be acquired from multiple sources, for example through interviews, observations, surveys or electronic health records (EHRs) [13]. The increased use of EHRs to store clinical data offers benefits to researchers, both for clinical and operations research [16]. Despite the wide availability of EHR data, the quality and suitability of this data for research purposes can be questioned [16, 17]. Examples of quality problems in EHRs are incomplete and inaccurate patient records. Especially for simulation studies, which often rely more on detailed and realistic inputs than other OR techniques, data quality issues may impede the validity of the results [13]. However, most research on simulation in EDs does not take data quality into account or lacks a description of the data cleaning process.

This paper focuses on data quality assessment of input data extracted from the EHRs of a hospital. Since EHRs are increasingly used as input data in ED and other healthcare simulation studies, and data quality issues tend to be present, there is a need for a structured approach to assess the quality of this data. The contribution of this paper is fourfold. Firstly, data quality problems in EHRs are identified, and a comprehensive framework for categorising these problems is developed. Secondly, data quality assessment techniques are described to identify and, whenever meaningful, quantify the problems mentioned in the framework. Thirdly, the developed theoretical foundation is used to implement the data quality assessment techniques in the R programming language. The implemented functions facilitate data quality assessment and enable the identification of quality issues in a dataset prior to its use in a simulation model. Furthermore, the functions are defined in a generic way and bundled in an R-package, which standardises and facilitates the identification and quantification of data quality problems. Finally, the implemented functions are applied to a real-life dataset to show their functionality in assessing input data quality. The real-life dataset consists of data extracted from the EHRs of the ED of a Belgian university hospital. Even though the framework and assessment techniques are described and illustrated from the viewpoint of ED simulation, this does not preclude its use in other research contexts such as other OR studies in healthcare. Examples are the simulation of outpatient clinics for appointment scheduling research, or a simulation of inpatient units to investigate admission policies.

The remainder of this paper is structured as follows. Section 2 gives an overview of related literature on ED simulation and data quality. In Section 3, the developed data quality framework and the accompanying assessment

techniques are discussed. In addition, a description of the implementation is provided based on an illustrative example. The implemented functions are applied to a real-life case study in Section 4. Section 5 provides a discussion on how to deal with data quality problems. The paper ends with a conclusion and future research opportunities in Section 6.

2. Related literature

The related literature is divided in two subsections. In Section 2.1, the importance of simulation to investigate ED operations, and the role of input data in simulation research, are discussed. Section 2.2 gives an overview of existing general and healthcare-specific data quality frameworks, indicating the need for a new comprehensive data quality framework that covers data quality problems present when using EHR data for simulation purposes.

2.1. Related literature on ED simulation

Simulation is the preferred technique to model and analyse the complex, stochastic and dynamic nature of ED operations for several reasons [18, 8]. Firstly, a great level of detail can be taken into account, such as individual patient characteristics, making the assumptions in a simulation model less restrictive [10, 8]. Secondly, time-dependent and stochastic characteristics can be included [19]. These characteristics make a simulation model capable of approximating real-life behaviour, and allow for reliable what-if analyses. Thirdly, simulation makes it possible to investigate the simultaneous effect of different improvements by taking interdependencies into account. Since patient flow through the ED results from the interplay of many factors, this can lead to valuable insights [10]. Finally, it is possible to analyse the effect of different process changes on a set of ED performance measures such as length of stay, door to doctor time, patient throughput, etc., and to identify trade-offs between these key performance indicators (KPIs).

Gul and Guneri [20] and Paul et al. [21] published reviews on ED simulation. These review papers emphasise the widespread use of simulation as an effective technique to analyse ED operations and to identify solutions to alleviate the negative effects of ED crowding. More general review papers on OR in an ED context also mention simulation as an appropriate and widely applied method to investigate and improve ED operations [8]. Implementations of operational improvements based on simulation findings mostly confirm the simulation results. This shows that simulation findings can be

reliable and valuable. Examples of successful implementations can be found in Konrad et al. [22], Oh et al. [23] and Paul and Lin [5].

Although simulation is a suitable technique to investigate ED operations, the generated output is only valuable when the model closely resembles the real system. Consequently, the construction of a realistic simulation model forms the basis of any simulation study [10]. This requires the presence of accurate data on the real system, i.e. input data [14]. Input data can facilitate both the specification of the model structure and its parameters. An overview of key simulation model components (i.e. entities, activities, resources and the control-flow), and accompanying simulation parameters (e.g. entity arrival rates), is provided in Martin et al. [24]. The reliability and granularity level of the simulation model depend on the availability, level of detail and quality of the input data [25, 26]. A lack of high quality input data necessitates the use of a higher abstraction level or simplifying assumptions, which impairs the validity of simulation results [13, 14].

Several methods exist to assess input data quality in a simulation context, ranging from subjective to more objective methods, with validation runs of the simulation model being the most frequently used method [27, 28, 15]. However, these methods do not always assess data quality in an objective or unambiguous way. This paper identifies and structures common data quality problems with input data for ED simulation. This facilitates the data quality assessment process, as researchers have some guidance on which type of problems may be present. In addition, this paper extends the state-of-the-art on data quality assessment by introducing a more objective and automated tool to systematically determine the presence and extent of the possible data quality problems. Based on the findings of data quality assessment, researchers have to find ways to resolve the problems or minimise their effect [14].

2.2. Related literature on data quality frameworks

To support the classification of data quality issues, this section explores existing literature on the classification of data quality problems, both general and healthcare-specific. Table 1 gives an overview of these frameworks and the main classification basis used to categorise data quality problems. While the first part of the table contains the general frameworks, the three rows in grey refer to healthcare-specific frameworks.

Looking at the general frameworks, four different classifications for data quality problems can be distinguished: (i) fitness for use, (ii) single or multi-source, (iii) schema or instance level, and (iv) problem manifestation classifi-

Table 1: Overview of existing data quality frameworks and their main classification basis

	Main classification			
Framework	Fitness for use	Single or multi-source	Schema or instance level	Problem manifestation
Wang and Strong [29]	X			
Rahm and Do [30]		X	X	
Kim et al. [31]				X
Mueller and Freytag [32]				X
Barateiro and Galhardas [33]		X	X	
Oliveira et al. [34]		X		
Gschwandtner et al. [35]		X		
Bose et al. [26]				X
Kahn et al. [27]	X			
Weiskopf and Weng [17]	X			
Mans et al. [36]				X

cations. Data quality frameworks built around the concept of fitness for use focus on the fact that data can be unsuitable for the application, even if the data is accurate. These frameworks take the viewpoint of the end user into account. The framework of Wang and Strong [29], one of the first frameworks proposed in literature, is based on this concept.

The second classification present in literature is based on a distinction between single-source and multi-source problems. Single-source problems are concerned with only one dataset and multi-source problems with the integration of multiple datasets. Barateiro and Galhardas [33], Gschwandtner et al. [35] and Rahm and Do [30] classify data quality problems into single- and multi-source problems, while Oliveira et al. [34] additionally focus on the number of datasets to integrate.

Thirdly, Barateiro and Galhardas [33] and Rahm and Do [30] distinguish schema and instance level problems in addition to their classification based on single-source and multi-source problems. Schema level problems refer to data quality issues that emerge because of a poor data model design and a lack of enforcement of data entry rules. Instance level problems are data quality problems inherent to the data values.

Finally, the main classification of Kim et al. [31], Mueller and Freytag [32] and Bose et al. [26] is based on the possible types of data anomalies. Mueller and Freytag [32] divide data quality problems into syntactical anomalies, semantic anomalies and coverage anomalies. Kim et al. [31] also developed a comprehensive classification of dirty data based on the manifestation of quality problems. A main distinction is made between missing and not-

missing data. Not-missing data is broken down further into wrong data and not wrong, but unusable data. Bose et al. [26] recognise four problem categories: missing data, incorrect data, imprecise data and irrelevant data.

Even though the main classifier differs between the existing frameworks, most frameworks overlap in the final data problems identified. In some frameworks these final problems are very specific, such that they can be measured by specific tests (e.g. [33, 26, 35, 31, 34, 30]). Examples of those final problem types are missing values, spelling errors, duplicated records, values outside domain ranges, etc. Other frameworks define broad problem categories, like accuracy, completeness, believability, timeliness, etc. (e.g. [29]).

The previously discussed frameworks are general data quality frameworks, applicable and adjustable to nearly every research context. Focusing on data quality in healthcare, three frameworks have recently been developed. Mans et al. [36] apply the framework of Bose et al. [26] to a healthcare context. The classifications of Kahn et al. [27] and Weiskopf and Weng [17] are based on the framework of Wang and Strong [29]. The latter framework is adjusted to only incorporate data quality problems relevant in a healthcare context and especially in the reuse of EHR data for clinical research.

Since the approach used in the framework of Mans et al. [36] is rather general (i.e. a value is either missing, incorrect, imprecise or irrelevant) and the frameworks of Kahn et al. [27] and Weiskopf and Weng [17] focus on the reuse of EHR data in clinical research, there are still deficiencies with regard to the use of EHR data in OR contexts in general and simulation in particular. Also, the end categories of the healthcare-specific frameworks tend to be too general, with every category still containing a very diverse collection of problems. To be able to define quality assessment methods and to avoid overlooking problems that are not immediately recognisable in a dataset, a greater level of detail is necessary. In this regard, this paper develops a comprehensive data quality framework that covers data quality problems present when using EHR data for simulation purposes in general, with ED simulation as a specific application. The framework is constructed in such a way that the end categories refer to specific data quality problems that are observable in a dataset. By presenting a functional data quality assessment framework, this paper provides an important extension of the state-of-the-art in literature.

3. Data quality framework and assessment techniques

This section outlines the developed data quality framework and the accompanying assessment techniques. Subsection 3.1 provides a general overview of the developed data quality framework. The implementation and automation of the data quality assessment techniques is described in Subsection 3.2, and illustrated by means of a fictitious example in Subsection 3.3.

3.1. General overview of the data quality framework

The new data quality framework can be found in Figure 1. It is based on insights from the existing frameworks - both general and healthcare-specific. The framework focuses on data quality problems present in EHR data of an ED, which is intended for use in an OR context, especially simulation. It is assumed that the hospital is the only authorised user of the EHR system, so they compose a data file with the requested information for the simulation study. This file is made available to the researcher, which implies that a distinction between single or multi-source problems is superfluous. The design of the data gathering system is also not considered in this paper, as the focus is on the quality of existing EHR data as input to a simulation study. Consequently, only instance level problems are identified. Since the context is already defined, we decide to use a problem manifestation classification instead of a fitness for use classification.

The main classification used in the framework is based on the framework of Kim et al. [31]. The problem classes of Bose et al. [26] and Mans et al. [36] are also covered by the framework. Data quality problems are split into missing data and not-missing data. Not-missing data is further divided into wrong data and not wrong but not directly usable data. The name of the last subcategory is changed compared to the framework of Kim et al. [31], where this category is named not wrong but unusable data. With regard to the intended use, the new name covers the category's content better. The category contains data that is not wrong, but some data preprocessing is required to make it usable for the purpose at hand. The main classification is further subdivided until specific, non-overlapping data quality problems are identified. These problems are indicated in grey in Figure 1.

3.2. Data quality assessment: the DAQAPO-package

In order to identify and quantify data quality problems, assessment techniques are required. Instead of executing ad hoc tests on a dataset, a more

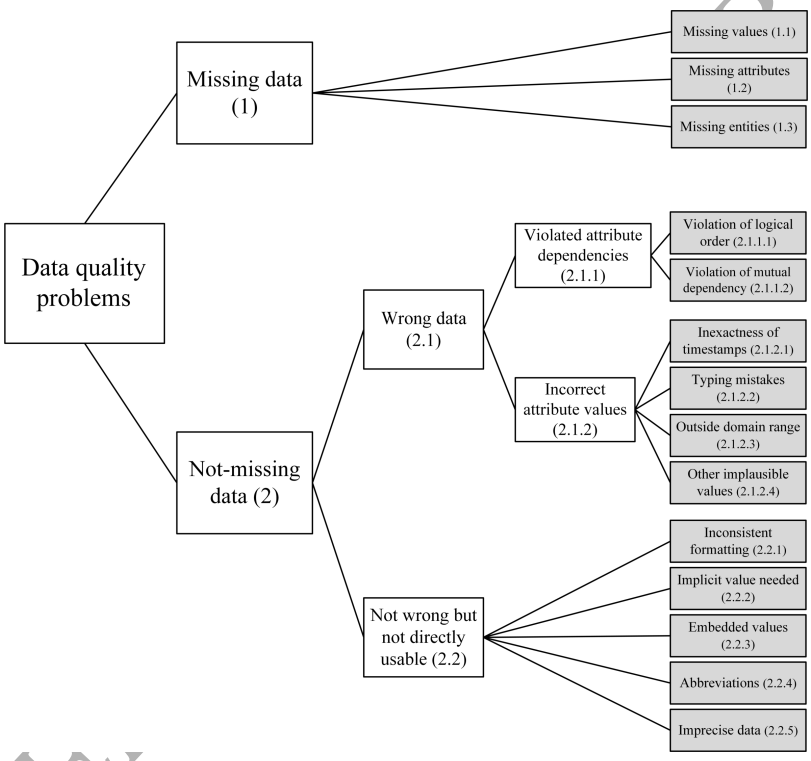


Figure 1: Data quality framework

systematic approach for data quality assessment is proposed in this paper. Generic data quality assessment techniques are developed for several quality problems in the framework.

The data quality assessment functions are implemented in R, which is a programming language that provides extensive functionalities for data manipulation and statistical analysis. All functions are bundled in an R-package called "DAQAPO": Data Quality Assessment for Process-Oriented data[1]. Packages are developed in R to make functionality and associated documentation easily accessible for its users. The functionality of the DAQAPO-package uses, amongst others, *dplyr* for data manipulations such as sorting and data summarisations, and *lubridate* to work with timestamps.

An overview of the functions in the DAQAPO-package is provided in Table 2. The table contains the function names, their purpose and their parameters. Each function has a set of input parameters, and the values of these parameters are user-defined and context-specific. This enables users to customise the function to the specific dataset, process and application, improving the generalisability of the DAQAPO-package to applications other than ED simulation. An overview of the input parameters of each function can be found in the 'parameters'-column of Table 2, and detailed information on these parameters is provided in the Appendix. While the parameters in bold should be defined explicitly, the other ones are optional as default values are specified. The function's output highlights a potential data quality issue. Based on this information, the package's user will need to determine whether this constitutes an actual and sufficiently important issue. Whether data are of sufficient quality depends on the purpose and the context of the simulation study. Examples of determining factors are the desired level of detail captured by the simulation model, the presence of other input data, the specific ED and the attributes under study (some attributes are considered more important and introduce more bias in the model than others).

Table 2: Overview of implemented functions in the DAQAPO-package

Function	Purpose	Parameters
missing_values	Detect missing values at different levels of aggregation	**activity_log** level_of_aggregation

[1]The DAQAPO-package is publicly available at `https://github.com/nielsmartin/daqapo`.

	(overview, column or activity)	colname details filter_condition
incomplete_cases	Detect incomplete cases in terms of the activities which need to be recorded for a case	**activity_log** **activity_vector** details filter_condition
inactive_periods	Detect periods in which no information is recorded in the dataset	**activity_log** **threshold_in_minutes** timestamp only_consider_start_activity details filter_condition
activity_order	Detect violations in activity order	**activity_log** **activity_order** timestamp details filter_condition
attribute_dependency	Detect violations of attribute dependencies	**activity_log** **condition_vector1** **condition_vector2** details filter_condition
related_activities	Detect missing attribute registrations for a case which should be registered given the fact that another activity is registered	**activity_log** **activity1** **activity2** details filter_condition
conditional_activity_presence	Detect the presence of an activity which is required under a particular condition	**activity_log** **condition_vector** **activity_vector** details filter_condition
duration_outliers	Detect duration outliers	**activity_log** **activity_considered** bound_sd lower_bound upper_bound details filter_condition
time_anomalies	Detect negative or zero durations	**activity_log** anomaly_type details filter_condition
multi_registration	Detect the registration of	**activity_log**

	multiple activities for the same case or by the same resource at the same point in time	level_of_aggregation timestamp **threshold_in_seconds** details filter_condition
attribute_range	Detect violations of attribute range	**activity_log** **column** **domain_range** timestamp_format details filter_condition

3.3. Data quality issues and assessment techniques

This section describes the functionality of the data quality framework and DAQAPO-package using an illustrative example. Section 3.3.1 introduces the illustrative example used to explain the data quality problems (Figure 1) and implemented assessment functions (Table 2). All data quality problems of the framework are outlined in Sections 3.3.2 to 3.3.4. The numbers between brackets in Figure 1 are used in the discussion to refer to the structure of the framework. For each problem, the accompanying assessment technique(s) and - if applicable - the implemented function(s) are explained by use of the illustrative example. Coding details are not presented, but for each implemented function the purpose and obtained output are described.

3.3.1. Illustrative example

An artificial dataset is constructed representing an extract from the EHRs of an ED. For illustrative purposes, it is assumed that the patient flow only consists of registration, triage, clinical examination, radiological examination (optional), treatment and discharge.

The artificial dataset describes the process that six patients followed in an ED at a specific date. The dataset has the form of an activity log, where each line describes the execution of an activity (e.g. clinical examination) on a particular case (i.e. patient). For each activity execution, additional information can be recorded about the activity such as its start time, completion time, and the resource that executed the activity. Moreover, case attributes can be recorded in an activity log, which are characteristics of a particular case [36]. For an ED patient, this can be the age, sex, triage code, diagnosis, etc. When data is structured in another format, R provides extensive data manipulation functionalities to convert the data to an activity log format.

The artificial dataset can be found in Figure 2. Each patient has a unique *case_id* which identifies the patient on which the activity is executed. The second column indicates the activity, while the third column refers to the resource that performed the activity. Resource information is represented by the type of resource and an identification number. In column four and five, the start and complete timestamp of the activity are registered. The last three columns contain case attributes (*cattr*), namely the age, triage code and specialisation a patient is assigned to. The activity log is ordered based on the start times of the activities. As an example, the first row in the illustrative dataset indicates that patient 508 is registered in the ED by clerk 4 on 20/11/2017 at 10:14:12. Registration ended at 10:17:50. The patient was 89 years old, had triage code 3 and was assigned to urgency medicine (i.e. URG).

	case_id	activity	resource	start	complete	cattr_age	cattr_triagecode	cattr_specialisation
1	508	Registration	Clerk 4	2017-11-20 10:14:12	2017-11-20 10:17:50	89	3	URG
2	509	Registration	Clerk 3	2017-11-20 10:16:17	2017-11-20 10:16:17	24	4	TRAU
3	508	Triage	Nurse 27	2017-11-20 10:18:08	2017-11-20 10:25:48	89	3	URG
4	509	Triage	Nurse 27	2017-11-20 10:22:18	2017-11-20 10:26:18	24	4	TRAU
5	510	Triage	Nurse 27	2017-11-20 10:29:33	2017-11-20 10:32:05	62	1	URG
6	510	Clinical exam	Doctor 7	2017-11-20 10:35:22	2017-11-20 10:41:05	62	1	URG
7	511	Registration	Clerk 4	2017-11-20 10:40:14	2017-11-20 10:43:00	36	2	URG
8	510	Radiological exam	Radiologist 2	2017-11-20 10:43:20	2017-11-20 11:03:56	62	1	URG
9	509	Clinical exam	Doctor 2	2017-11-20 10:50:05	2017-11-20 10:57:48	24	4	TRAU
10	511	Triage	Nurse 21	2017-11-20 10:54:11	2017-11-20 10:51:17	36	2	TRAU
11	512	Triage	Nurse 21	2017-11-20 10:56:01	2017-11-20 11:00:31	2	4	PED
12	512	Registration	Clerk 4	2017-11-20 11:00:33	2017-11-20 11:03:15	2	4	PED
13	508	Clinical exam	Doctor 7	2017-11-20 11:12:01	2017-11-20 11:22:09	89	3	URG
14	510	Treatment	Doctor 7	2017-11-20 11:12:24	2017-11-20 11:27:47	62	1	URG
15	511	Clinical exam	Doctor 2	*NA*	2017-11-20 11:20:09	36	2	TRAU
16	513	Registration	Clerk 3	2017-11-20 11:21:38	2017-11-20 11:25:01	15	*NA*	URG
17	513	Triage	Nurse 21	2017-11-20 11:24:59	2017-11-20 11:32:05	15	*NA*	URG
18	508	Radiological exam	Radiologist 1	2017-11-20 11:25:16	2017-11-20 11:43:00	89	3	URG
19	512	Clinical exam	Doctor 5	2017-11-20 11:28:51	2017-11-20 11:36:08	2	4	PED
20	511	Radiological exam	Radiologist 1	2017-11-20 11:31:14	2017-11-20 11:54:29	36	2	TRAU
21	509	Treatment	Doctor 2	2017-11-20 11:40:11	2017-11-20 11:56:02	24	4	TRAU
22	509	Discharge	Clerk 3	2017-11-20 11:56:14	2017-11-20 11:58:02	24	4	TRAU
23	508	Treatment	Doctor 7	2017-11-20 12:20:02	2017-11-20 12:34:00	89	3	URG
24	512	Discharge	Clerk 3	2017-11-20 12:24:13	2017-11-20 12:26:55	2	4	PED
25	511	Treatment	Doctor 2	2017-11-20 12:25:11	2017-11-20 12:31:18	36	2	TRAU
26	511	Discharge	Clerk 3	2017-11-20 12:33:29	2017-11-20 12:36:06	36	2	TRAU
27	508	Discharge	Clerk 3	2017-11-20 12:39:32	2017-11-20 12:43:00	89	3	URG

Figure 2: Artificial dataset used in the illustrative example

3.3.2. Missing data

The first part of the data quality framework concerns missing data (1). This is data that is missing in a field while it should not be missing [31]. Missing data are a very common and inevitable problem [37]. The fact that some

data values are missing can have two important negative effects. Firstly, it can lead to biased estimates for simulation parameters and statistics such as central tendency, dispersion or correlation, especially when the missingness is related to the (unknown) value of the attribute itself or another attribute in the dataset. Secondly, missing data reduces the statistical power of the data analysis results, because there are less cases available for the analysis [38]. There are three types of missing data: values, attributes and entities.

Missing values
Missing values (1.1) are mandatory attribute values that are missing for certain patients. The presence and quantity of missing values seems straightforward to identify, but an important consideration has to be made. In case null values are not possible for an attribute, every n.a., zero or empty field indicates a missing value. If missing values are not consistently represented, all representations have to be defined before quality assessment. In the other case, if null values are possible for an attribute, a distinction has to be made between missing and null values. A possible way to do this is by identifying dependencies with other attributes. For example, in an ED context, a bed request is not assigned if the patient is discharged home, otherwise this value is missing in the EHRs.

In the DAQAPO-package, the function *missing_values* is developed in order to detect missing values in a dataset. This function makes no distinction between missing and null values. When null values are possible, the *missing_values* function can be used in combination with a function to detect mutual dependency (see Section 3.3.3) in order to distinguish null values from missing values. The output of the *missing_values* function provides statistics on missing values in the dataset. Figure 3 shows the output obtained in R when applying the function to the complete artificial dataset. The absolute and relative number of missing values is defined for each column in the dataset. As can be seen, one start time and two triage code attributes are missing. Furthermore, the specific records in the activity log that contain missing values are shown. In the illustrative example, the triage code is missing for patient 513 and the start time of the clinical exam is missing for patient 511. Instead of identifying all missing values in the dataset (overview level), the function can also be applied at the activity level to determine the missing values for each activity as shown in Figure 4. This way activities with insufficient data registration can be identified easily.

An important consequence of missing values is the existence of incom-

```
Selected level of aggregation: overview

*** OUTPUT ***
Absolute number of missing values per column:

case_id              0
activity             0
resource             0
start                1
complete             0
cattr_age            0
cattr_triagecode     2
cattr_specialisation 0

Relative number of missing values per column (expressed as percentage):

case_id              0.000000
activity             0.000000
resource             0.000000
start                3.703704
complete             0.000000
cattr_age            0.000000
cattr_triagecode     7.407407
cattr_specialisation 0.000000

Overview of activity log rows which are incomplete:
   case_id      activity resource              start           complete cattr_age cattr_triagecode cattr_specialisation
15     511 Clinical exam Doctor 2               <NA> 2017-11-20 11:20:09        36              2                 TRAU
16     513  Registration  Clerk 3 2017-11-20 11:21:38 2017-11-20 11:25:01        15             NA                  URG
17     513        Triage Nurse 21 2017-11-20 11:24:59 2017-11-20 11:32:05        15             NA                  URG
```

Figure 3: Detailed output of the *missing_values* function at the overview level in the illustrative example

```
Selected level of aggregation: activity

*** OUTPUT ***
Absolute number of missing values per column (per activity):
# A tibble: 6 x 8
            activity case_id resource start complete cattr_age cattr_triagecode cattr_specialisation
               <chr>   <int>    <int> <int>    <int>     <int>            <int>                 <int>
1     Clinical exam       0        0     1        0         0                0                     0
2         Discharge       0        0     0        0         0                0                     0
3 Radiological exam       0        0     0        0         0                0                     0
4      Registration       0        0     0        0         0                1                     0
5         Treatment       0        0     0        0         0                0                     0
6            Triage       0        0     0        0         0                1                     0

Relative number of missing values per column (per activity, expressed as percentage):
# A tibble: 6 x 8
            activity case_id resource start complete cattr_age cattr_triagecode cattr_specialisation
               <chr>   <dbl>    <dbl> <dbl>    <dbl>     <dbl>            <dbl>                 <dbl>
1     Clinical exam       0        0   0.2        0         0        0.0000000                     0
2         Discharge       0        0   0.0        0         0        0.0000000                     0
3 Radiological exam       0        0   0.0        0         0        0.0000000                     0
4      Registration       0        0   0.0        0         0        0.2000000                     0
5         Treatment       0        0   0.0        0         0        0.0000000                     0
6            Triage       0        0   0.0        0         0        0.1666667                     0
```

Figure 4: Detailed output of the *missing_values* function on the activity level in the illustrative example

```
*** OUTPUT ***
It was checked whether the activities Clinical exam - Discharge - Registration - Treatment - Triage are present for cases.
These activities are present for 3 ( 50 %) of the cases and are not present for 3 ( 50 %) of the cases.

Note: this function only checks the presence of activities for a particular case, not the completness of these entries in the
activity log or the order of activities.

For cases for which the aforementioned activities are not all present, the following activities are recorded (ordered by decre
asing frequency of occurrence):
# A tibble: 3 x 3
                                  activity_list     n case_ids
                                          <chr> <int>    <chr>
1 Clinical exam - Discharge - Registration - Triage     1      512
2                 Clinical exam - Treatment - Triage     1      510
3                           Registration - Triage     1      513
```

Figure 5: Detailed output of the *incomplete_cases* function in the illustrative example

plete records. When a large number of records is incomplete, it becomes more difficult to reconstruct the exact patient flow through the ED. So besides assessing the missingness for every attribute separately, the amount of incomplete patient records is also an important measure.

The *incomplete_cases* function quantifies the number of incomplete patient paths in a dataset. For example, if registration, triage, clinical exam, treatment and discharge are standard activities that need to be executed on every patient, we can check if all patients undergo these activities. The output of applying this function on the artificial dataset is presented in Figure 5. From this figure, it follows that the required activities are not recorded for three patients. The output also shows which of the required activities have been executed for these patients.

Missing attributes

Missing attributes (1.2) are attributes needed as input to the simulation model which are not present in the data file. The difference with the previous category is that the values of these attributes are missing for every patient in the dataset. The attributes are either excluded from the extracted data file or they are not recorded in the EHRs.

Whether attributes are missing depends (partially) on the application, i.e. the level of detail at which (a part of) the ED needs to be modelled, compared to the level of detail in the registered data. If attributes are missing, the severity of this quality problem is contingent on the derivability of the attribute values from other data or the possibility to deduce a good estimate based on on-field observations or surveys. Given these considerations, it is possible to assess the presence of this quality problem, but measuring the extent of the problem is a subjective evaluation by the user of the data based on the specific application. For this reason, no assessment function is developed for missing attributes.

Missing entities

The last type of missing data are missing entities (1.3). Normally, every arriving patient is registered in the EHRs of an ED. However, because of a technical failure, human fault or error in the data extraction process, it is possible that patients (i.e. entities) are missing in the data file. When missing entities are a recurring problem, they may for example result in the estimation of incorrect arrival patterns based on the dataset. Since missing entities are not registered, the number of missing entities is not deductible from the data file. Nevertheless, it may be possible to identify the presence of the quality problem in case it is characterised by extended time periods without arrivals or registrations in the data file. However, caution is advisable when identifying this data quality problem. First of all, the acceptable time between arrivals may be season-, day- or hour-dependent. For example, a three hour time period without arrivals may indicate a problem during daytime, while at night this might be a common observation. Secondly, as exceptionally long periods without arrivals may occur in reality, an extended time period between arrivals does not necessarily indicate a data quality problem. The presence of an extended time period without arrivals only indicates that a possible data quality problem might exist, which should be verified before any conclusions on missing entities can be made.

In order to reveal extended time periods without arrivals, the *inactive_periods* function is developed. Figure 6 contains the output of applying the function. In the illustrative example, registration is the first activity every patient undergoes. Consequently, the start timestamps of registration are considered as these best approximate arrival times. The maximum time period that may elapse between consecutive arrivals is user-defined, and in Figure 6 it is set at 20 minutes. The results show that the time between consecutive arrivals exceeds the threshold of 20 minutes three times.

In addition to considering the time between consecutive arrivals, the *inactive_periods* function also provides the possibility to investigate the time between consecutive activities. In that case, the function identifies periods in which nothing is recorded in the system for an extended period of time. In an activity log, this actually defines if rows are missing, while investigating arrivals determines if cases (i.e. patients) are missing. Both specifications of inactive periods may indicate a technical failure of the system.

```
Selected timestamp parameter value: start

*** OUTPUT ***
Specified threshold of 20 minutes is violated 3 times.

Threshold is violated in the following periods:
        period_start          period_end time_gap
1 2017-11-20 10:16:17 2017-11-20 10:40:14 23.95000
2 2017-11-20 10:40:14 2017-11-20 11:00:33 20.31667
3 2017-11-20 11:00:33 2017-11-20 11:21:38 21.08333
```

Figure 6: Detailed output of the *inactive_periods* function applied to arrivals in the illustrative example

3.3.3. Wrong data

The first of two not-missing data categories is wrong data (2.1). This category can be further subdivided into violated attribute dependencies and incorrect attribute values. Violated attribute dependencies (2.1.1) are data values that cannot be identified as wrong without information about other attribute values. This category contains two data quality problems: violation of logical order and violation of mutual dependency. Incorrect attribute values (2.1.2) are data values that are wrong on their own, without violating their relation with other attributes. The four problem types in this category are inexactness of timestamps, typing mistakes, values outside domain ranges and other implausible values.

Violation of logical order

The first quality problem in the violated attribute dependencies category (2.1.1) is violation of logical order (2.1.1.1). Violation of logical order implies a problem with the timestamps of successive activities such that the patient flow based on timestamps in the data file is not correct compared to the regular patient flow. Since patient flow is site- and context-specific, the first step is to define the order of all activities in the regular patient flow. After that, investigating the timestamps of sequential activities enables the identification of this quality problem. If the regular patient flow depends on patient characteristics, a filter condition can be passed to the function. This way the dataset can be divided in subsets and a separate order for each subgroup of patients can be specified.

The detection of activity order violations is automated in the DAQAPO-package with the function *activity_order*. The output provides statistics on the number of cases that violate the prespecified order. Figure 7 illustrates the detailed output of applying the function in order to test if registration is always executed before triage. The results indicate that the order is not respected for two cases in the illustrative example. The details show that

```
Selected timestamp parameter value: both

*** OUTPUT ***
It was checked whether the activity order Registration - Triage is respected.
This activity order is respected for 4 ( 66.66667 %) of the cases and not for 2 ( 33.33333 %) of the cases.

Note: cases for which not all activities are recorded, will be considered as cases for which an incorrect order is registered.


Time overlap is detected between the following consecutive activities (ordered by decreasing frequency of occurrence):
# A tibble: 1 x 2
             overlapping     n
                   <chr> <int>
1 Registration and Triage     1

For cases for which the aformentioned activity order is not respected, the following order is detected (ordered by decreasing
frequeny of occurrence):
# A tibble: 2 x 3
          activity_list     n case_ids
                  <chr> <int>    <chr>
1               Triage      1      510
2 Triage - Registration     1      512
```

Figure 7: Detailed output of the *activity_order* function in the illustrative example

for patient 510 only triage is recorded, and for patient 512 triage is executed before registration. Furthermore, based on both start and completion timestamps, the function can detect overlap between registration and triage. As can be seen in the output, overlap between registration and triage is detected once in the data file.

Violation of mutual dependency

The second problem type related to attribute dependencies is violation of mutual dependency (2.1.1.2). Two attributes are called mutually dependent if the value of one attribute impacts the value of the other attribute. An example is the fact that a patient aged under 16 will be assigned to a paediatrician. A violation of a mutual dependency is present if two mutually dependent attributes have contradicting values. Mutual dependency can be identified in three ways: between attributes, between activities, and between an attribute and an activity. These three approaches are explained below.

The identification of violated mutual dependency between attribute values is implemented with the function *attribute_dependency*. This dependency is evaluated at the row level since all patient attributes are spread over different columns in the activity log. In the illustrative example, the mutual dependency between age and specialisation is verified for patients aged under 16. First, all patients aged under 16 are identified by the function. For these patients, the second condition is tested. In the illustrative activity log, two patients are aged under 16. For patient 513, the mutual dependency is violated since this patient is 15 years old and assigned to the specialisation "URG". These findings are summarised in Figure 8.

A second function, *related_activities*, can be used to determine whether a mutual dependency between activities in an activity log is violated. This

```
*** OUTPUT ***
The following statement was checked: if condition(s) cattr_age <= 16 hold(s), then cattr_specialisation == "PED" should also h
old

This statement holds for 4 ( 66.66667 %) of the rows in the activity log for which the first condition(s) hold and does not ho
ld for 2 ( 33.33333 %) of these rows.

For the following rows, the first condition(s) hold(s), but the second condition does not:
  case_id     activity resource             start          complete cattr_age cattr_triagecode cattr_specialisation
1     513 Registration  Clerk 3 2017-11-20 11:21:38 2017-11-20 11:25:01        15              NA                  URG
2     513       Triage Nurse 21 2017-11-20 11:24:59 2017-11-20 11:32:05        15              NA                  URG
```

Figure 8: Detailed output of the *attribute_dependency* function in the illustrative example

```
*** OUTPUT ***
The following statement was checked: if Triage is recorded for a case, then Registration should also be recorded.
This statement holds for 5 ( 83.33333 %) of the cases and does not hold for 1 ( 16.66667 %) of the cases.

For the following cases, only Triage is recorded:
[1] 510
```

Figure 9: Detailed output of the *related_activities* function in the illustrative example

dependency is evaluated at the column level since all activities executed on a patient are spread over different rows in the activity log. The function detects whether an activity that should be registered given the presence of another activity, is recorded in the data file. For example, the registration activity has to be present in the dataset if triage is registered. From the output in Figure 9, it follows that for patient 510 the registration activity is missing while triage is recorded.

A third possibility is a mutual dependency between an activity and an attribute in an activity log. For instance, if a triage code is recorded, the activity triage should be executed. The presence of a triage code can be identified at the level of a row, but to determine whether the activity triage is present the dataset has to be inspected over rows. This can be accomplished with the function *conditional_activity_presence*. The output in Figure 10 is obtained by applying the function to the artificial activity log in order to test if all patients with a triage code have actually undergone triage. For all patients in the illustrative example that are assigned a triage code, the activity triage is recorded.

```
*** OUTPUT ***
The following statement was checked: if condition(s) !is.na(cattr_triagecode) hold(s), then activity/activities Triage should
be recorded

The condition(s) hold(s) for 5 cases. From these cases:
- the specified activity/activities is/are recorded for 5 cases ( 100 %)
- the specified activity/activities is/are not recorded for 0 cases ( 0 %)
```

Figure 10: Detailed output of the *conditional_activity_presence* function in the illustrative example

Inexactness of timestamps

The next four data quality issues belong to the category of incorrect at-

```
Outliers are detected for activity Triage with lower bound 0 and upper bound 4 .

A total of 4 is detected ( 66.66667 % of the activity executions)
For the following rows, outliers are detected:
  case_id activity resource             start          complete cattr_age cattr_triagecode cattr_specialisation
1     508   Triage Nurse 27 2017-11-20 10:18:08 2017-11-20 10:25:48        89                3                 URG
2     511   Triage Nurse 21 2017-11-20 10:54:11 2017-11-20 10:51:17        36                2                TRAU
3     512   Triage Nurse 21 2017-11-20 10:56:01 2017-11-20 11:00:31         2                4                 PED
4     513   Triage Nurse 21 2017-11-20 11:24:59 2017-11-20 11:32:05        15               NA                 URG
   duration
1  7.666667
2 -2.900000
3  4.500000
4  7.100000
```

Figure 11: Detailed output of the *duration_outliers* function in the illustrative example

tribute values (2.1.2). The inexactness of timestamps problem (2.1.2.1) involves timestamps that are recorded imprecisely. There are two main causes of inexact timestamps. First of all, physicians give low priority to administrative tasks. They tend to postpone administrative tasks and bundle them for several patients. As a result, the timestamps in the EHRs can be an inaccurate representation of the execution time of the activity. Secondly, if timestamps are not recorded automatically by the system when a file is saved, input mistakes can lead to incorrect timestamps.

The problem is not always detectable, but four possible assessment methods exist. Firstly, by calculating durations of activities or between activities, outliers can be identified (see Hair [39] for more information on outlier detection). Outliers give an indication that one of the timestamps used in the calculations may be wrong. Secondly, negative or zero durations may also indicate inexact timestamps. Thirdly, resource information can be used as an indication of inexact timestamps. If the number of activities executed by one resource at more or less the same time is unrealistic, there is a high probability that the resource bundled the administrative tasks for several patients. Finally, if multiple activities are registered at more or less the same time for a patient, this may also indicate inexact timestamps.

All assessment methods are implemented in the DAQAPO-package. The function *duration_outliers* can be used to determine whether activity duration outliers are present. Outliers can be determined based on two conditions: a self-defined lower and upper bound, or a maximal deviation from the mean in terms of standard deviations. The results of applying the *duration_outliers* function to determine outliers in the triage duration can be found in Figure 11. A lower- and upper bound of zero and four minutes are specified, respectively. For four patients in the illustrative example, the triage duration falls outside this range and one of these durations is even negative.

A second function to identify inexact timestamps is called *time_anomalies*. This function detects all negative or zero durations in the activity log. Figure

```
Selected anomaly type: zero

*** OUTPUT ***
For 1 rows in the activity log ( 3.703704 %), an anomaly is detected.
The anomalies are spread over the activities as follows:
# A tibble: 1 x 2
      activity     n
         <chr> <int>
1 Registration     1


Anomalies are found in the following rows:
  case_id     activity resource             start          complete cattr_age cattr_triagecode cattr_specialisation
1     509 Registration  Clerk 3 2017-11-20 10:16:17 2017-11-20 10:16:17        24                4                  TRAU
  duration
1        0
```

Figure 12: Detailed output of the *time_anomalies* function in the illustrative example

```
Selected level of aggregation: resource
Selected timestamp parameter value: start

*** OUTPUT ***
Multi-registration is detected for 1 of the 9 resources ( 11.11111 %) These resources are:
[1] "Doctor 7"

For the following rows in the activity log, multi-registration is detected:
  case_id      activity resource             start          complete cattr_age cattr_triagecode cattr_specialisation
1     508 Clinical exam Doctor 7 2017-11-20 11:12:01 2017-11-20 11:22:09        89                3                  URG
2     510     Treatment Doctor 7 2017-11-20 11:12:24 2017-11-20 11:27:47        62                1                  URG
```

Figure 13: Detailed output of the *multi_registration* function on the resource level in the illustrative example

12 presents the results of detecting zero durations in the illustrative example. As can be seen, the registration of patient 509 has a duration of zero minutes according to the activity log, which is impossible in practice.

The third and fourth assessment method are automated with the *multi_registration* function. This function identifies the registration of multiple activities for the same case (case level) or by the same resource (resource level) within a user-defined timespan. Figure 13 displays the results of applying the *multi_registration* function at the resource level with a threshold of 60 seconds. In the illustrative example, doctor 7 probably bundled the registration of the clinical exam of patient 508 and the treatment of patient 510. When the function is applied at the case level with a threshold of 20 seconds, the output in Figure 14 is generated. A data quality problem is identified for four cases.

Typing mistakes

The second problem type in the incorrect attribute values category are typing mistakes (2.1.2.2). This quality issue focuses on text fields. Typing mistakes in numerical or categorical fields are included in one of the other categories, or may be unidentifiable (e.g. triage code 3 instead of 4 is registered). For example, a typing mistake in a timestamp field can manifest itself as a violation of logical order, inexact timestamp, or value outside the domain range.

As typing mistakes result in inexistent words, they can be identified using

```
Selected level of aggregation: case
Selected timestamp parameter value: both

*** OUTPUT ***
Multi-registration is detected for 4 of the 6 cases ( 66.66667 %) of the cases. These cases are:
[1] 508 509 512 513

For the following rows in the activity log, multi-registration is detected:
  case_id     activity resource             start           complete cattr_age cattr_triagecode cattr_specialisation
1     508 Registration  Clerk 4 2017-11-20 10:14:12 2017-11-20 10:17:50        89               3                  URG
2     508       Triage Nurse 27 2017-11-20 10:18:08 2017-11-20 10:25:48        89               3                  URG
3     509    Treatment Doctor 2 2017-11-20 11:40:11 2017-11-20 11:56:02        24               4                 TRAU
4     509    Discharge  Clerk 3 2017-11-20 11:56:14 2017-11-20 11:58:02        24               4                 TRAU
5     512       Triage Nurse 21 2017-11-20 10:56:01 2017-11-20 11:00:31         2               4                  PED
6     512 Registration  Clerk 4 2017-11-20 11:00:33 2017-11-20 11:03:15         2               4                  PED
7     513 Registration  Clerk 3 2017-11-20 11:21:38 2017-11-20 11:25:01        15              NA                  URG
8     513       Triage Nurse 21 2017-11-20 11:24:59 2017-11-20 11:32:05        15              NA                  URG
```

Figure 14: Detailed output of the *multi_registration* function on the case level in the illustrative example

an online dictionary, a spell check in data processing software or methods related to natural language processing. This makes the identification of typing mistakes a very time-consuming task. In addition, the severity of the problem depends not only on the number of typing mistakes, but also on the ease of identification and correction. For these reasons, no technique is provided to identify and quantify this problem.

Outside domain range

Values outside the domain range (2.1.2.3) concern timestamps, numerical and categorical values that are impossible given the range of possible values. Timestamps and numerical attributes with a value smaller than the minimum or larger than the maximum acceptable value lie outside the domain range. For categorical attributes, this implies that the recorded value is no element of the possible value set.

The function *attribute_range* checks whether the values of an attribute or timestamp fall within the range of possible values. Figure 15 contains the output of testing whether all triage codes fall within the range [1:5]. The output indicates that two rows in the artificial activity log have a triage code outside this range. The detailed information shows that these values are NAs.

Other implausible values

The last problem type is a residual category for wrong data values that do not fit in one of the previous categories (2.1.2.4). As EHR-data is hospital-specific and can be used for multiple OR purposes, a data quality problem not explicitly included in the framework may appear. Since this is the residual category, it is not possible to define a general assessment method for these problems. If there is a suspicion of a quality problem not captured in the previous categories, this can be checked and the number of implausible values

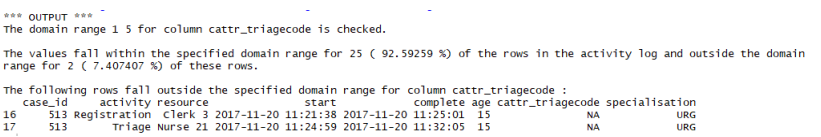


Figure 15: Detailed output of the *attribute_range* function in the illustrative example

can be calculated.

3.3.4. Not wrong but not directly usable data

The second not-missing data category is not wrong but not directly usable data (2.2). This category contains problems with data values which are not wrong, but not suitable for use in their current form. Data preprocessing is required to transform the raw data values into usable data. This category is subdivided into five specific problem types: inconsistent formatting, implicit value needed, embedded values, abbreviations and imprecise data.

The simplicity of identifying and resolving these problems is relevant information. Therefore, possible approaches to identify the problems, and solution methods to convert the data values into values usable for the application, are suggested in this subsection. Conversely, quantifying the problems is not very useful, given the fact that this category only contains solvable problems. Also, the suitability of data depends on the specific application, which makes it difficult to define and implement generic assessment methods. Therefore, no assessment techniques are provided to quantify the data quality problems in this category.

Inconsistent formatting

The first data quality problem in this category is inconsistently formatted data (2.2.1). Inconsistent formatting means that the format of data values varies within one attribute or among attributes of the same type. The problem can arise in different forms. The same representation can be used for different values (e.g. n.a. for both zero and empty fields) or different representations can be used for the same value (e.g. n.a. and 0 both indicate the absence of a value). Another possibility relates to data types instead of data values, namely the use of a different coding among attributes of the same data type (e.g. dates are presented as DD-MM-YY or YY-MM-DD).

To identify inconsistently formatted data, attributes of the same data type should be investigated. Additionally, data values within one attribute

should be inspected on consistency. If the problem is present, the inconsistent values have to be transformed in order to obtain a consistent representation throughout the data file. For standard data formats (e.g. dates, times, names), this can be done by changing the cell properties with a data manipulation tool. In case of application domain specific coding, such as the International Classification of Diseases (ICD) codes in a healthcare context, reformatting is more complex. Expert assistance will be required in that case.

Implicit value needed
Implicit value needed (2.2.2) implies that an attribute value that is inherent to a patient or activity is not present in the data file. The value is absent because the action is not executed or not all details of the action are registered in the EHRs. However, as this category deals with inherent attributes, the value can be assigned afterwards even without registration of the information. For example, start and/or completion timestamps of an activity are registered while the duration is needed as input to the simulation model. In this case, the value is implicitly present in the data file, but not recorded as a separate attribute. Sometimes, the implicit values can be deducted from other attribute values in the data file. Otherwise, expert assistance, estimates based on other patients or empirical data gathering are possible ways to obtain an indication of the data value.

Embedded values
The embedded values problem (2.2.3) refers to the situation in which a data field contains more than one value. For example, age and gender are present in the same field (e.g. M42 refers to a male person of age 42), while only age is needed. If input values to the simulation model are present in the dataset in an embedded form, these data fields have to be split in order to be useful in the analysis. Data manipulation tools enable their users to split the attribute values according to a prespecified rule.

Abbreviations
Abbreviations (2.2.4) are the fourth problem type. They are problematic if their meaning has to be derived to be useful for analysis purposes. Especially in case of domain-specific terminology, abbreviations are difficult to interpret. Detecting an abbreviation is easy, but time-consuming. To detect abbreviations, all unique values of an attribute can be retrieved, e.g. using R, and systematically replaced with full text in case they are problematic.

Standard abbreviations can be simply transformed to complete words, but expert assistance is probably required for domain-specific terminology.

Imprecise data
Finally, the imprecise data category (2.2.5) comprises values that are correct, but that are not necessarily specified at the appropriate level of detail. An example are timestamps that contain only the date of execution, instead of the specific time. Measuring units give an indication of the precision of an attribute and, as a result, of the presence of this problem. Measuring units are straightforward to determine, as they are available in the data file or in the metadata. If the attribute values are not specified at the appropriate level of detail, additional data collection is required. Extra data can be extracted from the EHRs of the ED or another department if the data are recorded in more detail. An example is the extraction of detailed radiological data from the EHRs of the radiology department. Another option is empirical data gathering. If additional data cannot be collected, modelling at a higher abstraction level is necessary.

4. Case study

To demonstrate the applicability and usefulness of the DAQAPO-package in a real-life context, the package is applied to a real-life dataset. The real-life dataset consists of data extracted from the EHR of the ED of a Belgian university hospital. The number of patient visits in the ED under study amounted to 57,650 in 2016, 60,727 in 2017, and this number is expected to increase even further in 2018. As there is no proportionate capacity expansion, the increase in patient visits causes (over)crowding in the ED. The extracted EHR data will be used as input data to a simulation model in order to analyse and optimise ED performance. A key step in the simulation modelling process is to assess the quality of this data.

EHRs are used throughout the entire hospital to standardise data gathering and to facilitate data exchange between departments. They capture the medical information of every patient and his flow throughout the hospital. Each hospital department has its own EHR, which is adapted to their specific needs. The extracted data file from the ED EHR contains anonymised patient records for all patients that visited the ED in October, November and December 2016. The patient records contain basic personal and medical information. Furthermore, patient flow information is registered at every

stage in the ED. The quality of the data should be carefully assessed, as accurate and timely data registration, especially the registration of non-clinical information, might not always be the primary focus of healthcare providers.

In terms of structure, each row in the dataset contains all information for a single patient. The dataset contains data on 14,902 patients and has 64 columns containing both patient and activity attributes. For quality assessment purposes, the data file is transformed into an activity log format similar to Figure 2. After conversion, the activity log contains a total of 154,736 rows, all related to one of the 14,902 patients. Each row refers to an activity that is executed on a patient. In addition to the activity and case_id, each row contains a completion timestamp of the activity and 44 case attributes. These case attributes are a combination of string, categorical, dummy, time and numerical variables. Examples are age, transport mode to the ED, triage code, discharge type, outflow destination, etc. Resource information and the start timestamps of activities are not recorded in the EHR of the ED under study. They can be seen as missing attributes, since they are necessary input values to the simulation model.

The next paragraphs outline the key findings of applying the developed functions to the dataset. An investigation of the missing values in the dataset reveals that the number of missing values is less than 1% for all obligatory case attributes (e.g. case_id, age, discharge type, etc.), except for triage code. At first sight, this attribute seems to be missing in 12.88% of the rows in the activity log. However, if the mutual dependency between the execution of the triage activity and the assignment of a triage code is investigated, all patients that have undergone triage are assigned a triage code. This indicates that all missing values are related to patients for which no triage is executed, so these values are not missing. Moreover, the findings reveal that triage is never executed at night. When looking at optional case attributes (e.g. ambulance_id, inpatient unit, types of radiological examinations, etc.), a large number of missing values is detected for the inpatient unit a patient is admitted to. This attribute is missing for 56.42% of the rows in the activity log. Since this attribute is only recorded for patients who are admitted to the hospital, not all the values are actually missing. By looking at the mutual dependency with the discharge type attribute, it is found that only 1% of these missing values are related to patients who are actually admitted to the hospital. This corresponds to 67 distinct patients. These results show that combining the evaluation of missing values and mutually dependent attributes can lead to valuable insights.

In addition to missing values, a lot of incomplete records are identified in the dataset. Every patient flow should at least consist of registration, triage, clinical examination and medical completion by a physician, and departure. For none of the patients in the dataset, all activities are recorded. Registration and departure are the only activities that are present for every patient. Triage is not registered for 12.88% of the patients (for reasons outlined above), while for 88.53% of the patients the clinical examination is missing. For 8.31% of the patients, the medical completion by a physician is not recorded. Optional activities, such as radiological or laboratory examinations, or the different treatments that are possible, may also be missing. Since these activities are optional, their missingness can only be detected based on dependencies with other attribute values. As a result of the incomplete records, it will be difficult to reconstruct the complete patient flow in the simulation model solely using the EHR input data.

Missing entities are identified by assessing the presence of time intervals between two arrivals that exceed 60 minutes. In the three months under investigation, this threshold is exceeded 199 times. The detailed information indicates that 20 times a time period of more than 2 hours between two consecutive arrivals is detected, while 3 times the time interval exceeded 3 hours with a maximum of 3 hours and 27 minutes. All inactive periods take place between 23:00 and 09:00. The actual observed interarrival times are larger during the night shift, and in reality an unusually long time period between consecutive arrivals may occur sporadically. This implies that the detected time periods without arrivals do not necessarily relate to missing entities. These time intervals need further investigation to exclude the presence of missing entities or the wrong registration of arrival times. As missing entities may lead to incorrect arrival patterns, and correspondingly an underestimation of workload, a correct identification of this quality problem is important. Given the fact that the observed inactive periods take place during the night, and the maximum duration of an inactive period does not seem exceptional in practice, they probably represent actual periods without arrivals.

Besides missing data, four types of wrong data issues are detected in the dataset. Firstly, the logical activity order is violated several times. According to the data, 7.43% of the patients left the ED before they were medically finished. Other activity order violations occur for radiological and laboratory examinations, which are sometimes executed before they are ordered, or the results are available before execution. The rule that a departure to

an inpatient unit should be preceded by a formal bed assignment (i.e. mutation plan), which should in turn be preceded by a mutation request (i.e. a request to the inpatient units for patient admission and, consequently, bed assignment), is also violated in the dataset, but for less than 1% of the cases.

Secondly, mutual dependency is not always complied with. Some examples are already given in the paragraph on missing values. Another example is the link between age and specialisation. Of the patients aged under 16 that are admitted to the hospital, 10.87% are not immediately assigned to a pediatric unit. Furthermore, no mutation request is demanded and no mutation plan is assigned to 30.92% of all admitted patients. A positive finding is that all relevant related activities are always present together in the dataset (apart from possible problems with the order in which they are executed). For example, a report of a radiological examination is only prepared if the examination is also executed.

Thirdly, violations of attribute ranges are detected in the dataset. The dataset contains 131 rows with an activity timestamp outside the extraction period from 01/10/2016 to 31/12/2016. These rows refer to activities that are executed on patients that arrived within the extraction period. This information may be correct, but the rows are best removed from the dataset to avoid bias when analysing and using these data.

Finally, the problem of multi-registration is identified, and this is probably the largest quality problem related to wrong data in our case study. There is no resource information in the dataset, but multi-registration can be detected at the case level. For 91.68% of the cases, two or more activities are recorded within the same minute. Consequently, the accuracy of some of the recorded timestamps can be questioned.

As is clear from the discussion above, the DAQAPO-package can be used to detect and evaluate data quality problems in a real-life dataset. It provides an easy and fast way to identify and quantify potential data quality problems. Once possible data quality problems for the dataset under study are determined, their detection and quantification only takes a few minutes by use of the DAQAPO-package, while manual detection can easily take several hours. All types of datasets can be assessed after conversion to the appropriate format. Furthermore, the functions are generic and can be adjusted to the dataset under study by specifying its parameters. The output leads to useful insights which should be taken into account before use of the data as input to a simulation model. The output also makes it possible to consult with the hospital in a structured way on the data quality problems present

in their EHRs.

5. Discussion

The developed data quality framework and assessment techniques facilitate the identification and quantification of data quality problems in EHR data. A final step to obtain high quality input data is data quality improvement. This involves a decision on how to deal with the detected issues. On the one hand, the problem can be addressed at the source in order to prevent data quality problems in the future. On the other hand, researchers can try to resolve the data quality problems in order to make the available EHR data suitable for the application.

Accurate data registration is important, and improvements are still possible as to prevent data quality problems. The identification and assessment of data quality problems may enhance the hospital's awareness of the problem. One way to avoid quality problems in EHRs is to improve the data collection system itself [40], for example by expanding the number of data entries, using error messages when a required data field is missing, enforcing the registration of both start and end timestamps of an activity, etc. More general approaches can be found in, for example, Rahm and Do [30]. Besides intervening on the system side, staff training on correct data registration may prevent data quality problems from happening [40]. Introducing and enforcing standard procedures for data registration which nurses and physicians should follow can also be effective to avoid several data quality problems. Awareness creation among staff members is important as EHR data are not registered with a primary focus on OR applications, but for billing or clinical purposes. Finally, given the growing interest of hospital managers in operational analyses of hospital processes, and the role of high quality input data in this regard, adapting EHR systems to collect more and accurate operational data can be valuable [41, 11, 42].

Assessing the quality of the current data using DAQAPO is a valuable starting point to prioritise particular changes to data recording procedures. Based on historical data, data quality problems present in the EHRs under study can be identified. For these quality problems, relevant assessment methods and suitable function parameters based on the specific ED context can be determined. Once a set of relevant quality tests from the DAQAPO-package is selected, they can be applied in real-time by integrating them in the data recording system. To this end, a trigger needs to be determined

which defines the moment at which a particular quality test needs to be executed. A straightforward example of a trigger involves performing the set of quality tests on a regular basis (e.g. hourly, daily, weekly, etc.). This enables the ED to quickly gain insight into data quality problems that are present in the EHRs and, for example, to identify problems that accrue systematically. This makes it possible for hospital managers to anticipate any quality problems, and to intervene promptly when necessary (e.g. adjust registration procedures for ED personnel). In this way, the developed quality assessment techniques may contribute to the evaluation of data quality on a regular basis, which can in turn contribute to a higher EHR data quality. The DAQAPO-package can also serve as a starting point for the implementation of relevant quality checks in the data recording system. This way, data quality assessment can be completely automated. This allows to evaluate the quality of the EHRs in real-time and to prevent data quality problems from reoccurring in the future.

Preventive measures are worthwhile to reduce the number of data quality problems, but completely error-free data is difficult, if not impossible, to attain. In addition, preventive action takes time while data quality errors are mostly detected at the moment the data are needed for analysis purposes. Consequently, corrective actions are unavoidable to resolve data quality problems present in EHRs. Multiple methods to deal with data quality problems are discussed in literature, most of them focusing on missing data. Methods to deal with missing data are, among others, listwise- and pairwise deletion, mean substitution, regression-based single imputation, multiple imputation and maximum likelihood estimation [43, 44, 37, 45]. Guo et al. [46] and Kuo et al. [47] developed a simulation-optimisation approach to estimate service-time distributions in case only start- or end timestamps of activities are present in the dataset. Simulation-optimisation is also used by Liu et al. [10] to estimate simulation model parameters in case of data scarcity. A prerequisite for this method is the availability of actual KPI data to compare model output against. Other problem types for which solution methods are provided in literature are outliers, invalid data [48], inaccurate arrival timestamps [49], and patient pathway detection in order to determine the logical order of activities or dependency between patient types and activity order [41, 50]. Solving all problems in a dataset might not be possible, but these methods may provide a helpful starting point to deal with certain (solvable) data quality problems.

Data quality problem prevention and correction both contribute to a

higher input data quality, which in turn improves simulation model reliability. Nonetheless, EHR data may not be the only source of input data to a simulation model. As not every detail of ED operations can be captured in the EHRs, and not all data quality problems in EHRs are resolvable, this data should be combined with information from other sources such as observations, interviews, etc. All input data sources have their limitations, but integrating data obtained by different data collection methods is advisable as this can improve simulation model reliability.

6. Conclusion and future research

Data quality assessment can be a time-consuming task. This paper presented a data quality framework with its associated assessment techniques. The developed R-package DAQAPO standardises and facilitates the process of identifying and quantifying potential data quality problems. The functions are defined in a generic way, and can hence be tailored to a specific context through the use of the function's parameters. The usefulness of the DAQAPO-package for assessing input data quality is demonstrated with a real-life case study. The framework and assessment techniques are described and illustrated from the viewpoint of ED simulation model development, but this does not preclude its use in other research contexts such as other OR studies in healthcare.

The efforts presented in this paper can be extended in future research. First of all, future research may focus on analysing to what extent the framework is directly applicable to other OR techniques and to OR in other (healthcare) domains, and on identifying which extensions are required to even further generalise the framework to other contexts. Secondly, new assessment techniques can be developed. Both advancements of existing techniques, and the development of new techniques for data quality problems not included in the current set of functions, can be worthwhile. Finally, investigating solutions for the different data quality problems is a valuable direction for future work.

Appendix

Missing_values function (DQP[2] 1.1)
The function *missing_values* is developed in order to detect missing values in a dataset. The function has 5 parameters:

- The ***activity_log*** parameter is used to refer to the name of the activity log (i.e. dataset) under investigation.[3]
- The ***level_of_aggregation*** parameter defines whether the function should be applied to the complete dataset (i.e. define all missing values in the dataset), a single column (i.e. define all missing values in a specific column) or the activity level (i.e. define missing values for each activity separately).
- In case the function is applied at the column level, the column name should be indicated in the ***colname*** parameter.
- The ***details*** parameter defines if detailed information on the presence of a quality problem is desired (default), or if summary statistics suffice.[3]
- The ***filter_condition*** parameter provides the possibility to insert a condition to obtain a subset of the activity log. In case a condition is defined, the function is only applied to the retrieved subset. This enables the user to interactively use the functions by focusing on more specific subsets of the activity log.[3]

Incomplete_cases function (DQP 1.1)
The *incomplete_cases* function quantifies the number of incomplete patient flows in a dataset. The function has one parameter in addition to the three parameters all functions have in common:

- The ***activity_vector*** parameter lists the activities which need to be present for a case in order to be complete.

[2] i.e. Data Quality Problem
[3] The *activity_log*, *details* and *filter_condition* parameters are common to all functions. Their explanation is not repeated in the description of the other functions.

Inactive_periods function (DQP 1.3)
The *inactive_periods* function is developed in order to reveal extended time periods without arrivals. This function has three parameters in addition to the three parameters all functions have in common:

- The ***threshold_in_minutes*** parameter defines the time period that may elapse between consecutive arrivals before it should be reported in the output.
- The ***timestamp*** parameter is present to indicate if start, complete or both timestamps should be used when identifying inactive periods. In case both timestamps are used, the difference between the completion of an activity and the start of the next activity is evaluated.
- The ***only_consider_start_activity*** parameter specifies the first activity that is executed on a case. The timestamp of this activity is used as approximation for the arrival time.

Activity_order function (DQP 2.1.1.1)
The function *activity_order* automates the detection of activity order violations. The function has two function-specific parameters:

- The ***timestamp*** parameter determines which timestamps are compared when checking the activity order: start, complete or both. In case both timestamps are considered, overlap of sequential activities is also detected.
- In the ***activity_order*** parameter, a vector should be defined that indicates the regular order of activities. The function checks for all cases in the activity log whether the recorded timestamps follow this order. If the regular order depends on case characteristics, a filter condition can be passed to the function. This way the dataset can be divided in subsets and a separate order for each subgroup of cases can be specified.

Attribute_dependency function (DQP 2.1.1.2)
The identification of violated mutual dependency between attribute values is implemented with the function *attribute_dependency*. The function has two parameters in addition to the three parameters all functions have in common:

- The ***condition_vector1*** parameter contains one or more conditions that are checked in the dataset.
- The ***Condition_vector2*** parameter contains one or more conditions that should be fulfilled if *condition_vector1* holds. Otherwise, the mutual dependency is violated.

Related_activities function (DQP 2.1.1.2)
The function *related_activities* can be used to determine whether a mutual dependency between activities in an activity log is violated. More specifically, the function detects whether an activity that should be registered given the presence of another activity, is recorded in the data file. The function has two function-specific parameters:

- The ***activity1*** parameter contains the name of the first activity in the set of mutual dependent activities.
- The ***activity2*** parameter is used to pass the name of the second activity, which should be present if the first activity is executed, to the function.

Conditional_activity_presence function (DQP 2.1.1.2)
The function *conditional_activity_presence* evaluates the mutual dependency between an activity and an attribute in an activity log. The function has two specific parameters above the general parameters:

- The ***condition_vector*** parameter determines the condition that is tested for an attribute in the data file. This results in a subset of cases from the activity log.
- The ***activity_vector*** parameter specifies the activity that should be present for each case that meets the condition. Since both parameters are vectors, multiple conditions or activities can be specified simultaneously.

Duration_outliers function (DQP 2.1.2.1)
The function *duration_outliers* can be used to determine whether activity duration outliers are present. Outliers can be determined based on

two conditions: a self-defined lower and upper bound, or a maximal deviation from the mean in terms of standard deviations. The default settings of the function define an outlier as a value that falls outside the range $[mean - 3 * SD; mean + 3 * SD]$. The function has four function-specific parameters:

- The activity under consideration is specified in the ***activity_considered*** parameter.
- In case an upper and lower bound are defined, a value is assigned to the parameters ***lower_bound*** and ***upper_bound***.
- In case outliers are defined based on standard deviations, the maximum acceptable number of standard deviations (SD) that a value may deviate from the mean is specified in the ***bound_sd*** parameter.

Time_anomalies function (DQP 2.1.2.1)
The *time_anomalies* function detects all negative or zero durations in the activity log. The function has only one parameter in addition to the general parameters:

- The ***anomaly_type*** parameter indicates if negative, zero or both negative and zero activity durations have to be detected in the activity log.

Multi_registration function (DQP 2.1.2.1)
The *multi_registration* function identifies the registration of multiple activities for the same case (case level) or by the same resource (resource level) in a too short timespan. This function has three function-specific parameters:

- The ***level_of_aggregation*** parameter specifies if the registration of multiple activities for the same case (case level) or by the same resource (resource level) in a short timespan has to be identified.
- The ***timestamp*** parameter determines if start, complete or both timestamps should be compared to detect inexact timestamps. In case both timestamps are used, the time between the end of an activity and the start of the successive activity is considered. With start or complete timestamps, the difference between the selected timestamp of successive activities is studied.

- The ***threshold_in_seconds*** parameter determines the minimum time interval (in seconds) between two activities executed by the same resource or on the same case in order to be realistic.

Attribute_range function (DQP 2.1.2.3)
The function *attribute_range* checks whether the values of an attribute or timestamp fall within the range of possible values. The function has three parameters in addition to the three parameters all function have in common:

- The function requires the specification of the attribute which should be investigated in the ***column*** parameter.
- The range of possible values for the attribute is passed to the function by the ***domain_range*** parameter.
- In case the domain range consists of timestamps, the format of the timestamps in the domain range (e.g. "DD-MM-YYY hh:mm:ss") should be defined in the ***timestamp_format*** parameter for formatting purposes.

References

References

[1] D. Sinreich, O. Jabali, Staggered work shifts: a way to downsize and restructure an emergency department workforce yet maintain current operational performance, Health Care Management Science 10 (2007) 293–308. doi:10.1007/s10729-007-9021-z.

[2] L. V. A. Downey, L. S. Zun, Determinates of Throughput Times in the Emergency Department, Journal of Health Management 9 (2007) 51–58. doi:10.1177/097206340700900103.

[3] J. Bergs, D. Vandijck, O. Hoogmartens, P. Heerinckx, D. Van Sassenbroeck, B. Depaire, W. Marneffe, S. Verelst, Emergency department crowding: Time to shift the paradigm from predicting and controlling to analysing and managing, International Emergency Nursing 24 (2016) 74–77. doi:10.1016/j.ienj.2015.05.004.

[4] N. R. Hoot, D. Aronsky, Systematic Review of Emergency Department Crowding: Causes, Effects, and Solutions, Annals of Emergency Medicine 52 (2008) 126–136.e1. doi:10.1016/j.annemergmed.2008.03.014.

[5] J. A. Paul, L. Lin, Models for Improving Patient Throughput and Waiting at Hospital Emergency Departments, The Journal of Emergency Medicine 43 (2012) 1119–1126. doi:10.1016/j.jemermed.2012.01.063.

[6] M. A. Ahmed, T. M. Alkhamis, Simulation optimization for an emergency department healthcare unit in Kuwait, European Journal of Operational Research 198 (2009) 936–942. doi:10.1016/j.ejor.2008.10.025.

[7] R. Carmen, I. Van Nieuwenhuyse, Improving patient flow in emergency departments with OR techniques: a literature overview, 2014.

[8] S. Saghafian, G. Austin, S. J. Traub, Operations research/management contributions to emergency department patient flow optimization: Review and research prospects, IIE Transactions on Healthcare Systems Engineering 5 (2015) 101–123. doi:10.1080/19488300.2015.1017676.

[9] M. M. Günal, M. Pidd, Discrete event simulation for performance modelling in health care: a review of the literature, Journal of Simulation 4 (2010) 42–51. doi:10.1057/jos.2009.25.

[10] Z. Liu, D. Rexachs, F. Epelde, E. Luque, A simulation and optimization based method for calibrating agent-based emergency department models under data scarcity, Computers & Industrial Engineering 103 (2017) 300–309. doi:10.1016/j.cie.2016.11.036.

[11] P. Baumgartel, R. Lenz, Towards data and data quality management for large scale healthcare simulations, in: Proceedings of the International Conference on Health Informatics, 2012, pp. 275–280.

[12] T. Altiok, B. Melamed, Input Analysis, in: Simulation Modeling and Analysis with ARENA, Elsevier, 2007, pp. 123–139. doi:10.1016/B978-012370523-5/50008-0.

[13] X. Hu, S. Barnes, B. Golden, Applying queueing theory to the study of emergency department operations: a survey and a discussion of comparable simulation studies, International Transactions in Operational Research 25 (2018) 7–49. doi:10.1111/itor.12400.

[14] W. D. Kelton, R. P. Sadowski, N. B. Zupick, Simulation with Arena, 6. ed., internat. student ed ed., McGraw-Hill Education, New York, NY, 2015.

[15] A. Skoogh, T. Perera, B. Johansson, Input data management in simulation Industrial practices and future trends, Simulation Modelling Practice and Theory 29 (2012) 181–192. doi:10.1016/j.simpat.2012.07.009.

[16] A. P. Reimer, A. Milinovich, E. A. Madigan, Data quality assessment framework to assess electronic medical record data for use in research, International Journal of Medical Informatics 90 (2016) 40–47. doi:10.1016/j.ijmedinf.2016.03.006.

[17] N. G. Weiskopf, C. Weng, Methods and dimensions of electronic health record data quality assessment: enabling reuse for clinical research, Journal of the American Medical Informatics Association 20 (2013) 144–151. doi:10.1136/amiajnl-2011-000681.

[18] P. Bhattacharjee, P. K. Ray, Patient flow modelling and performance analysis of healthcare delivery processes in hospitals: A review and reflections, Computers & Industrial Engineering 78 (2014) 299–312. doi:10.1016/j.cie.2014.04.016.

[19] D. Sinreich, Y. Marmor, Emergency department operations: the basis for developing a simulation tool, IIE transactions 37 (2005) 233–245.

[20] M. Gul, A. F. Guneri, A comprehensive review of emergency department simulation applications for normal and disaster conditions, Computers & Industrial Engineering 83 (2015) 327–344. doi:10.1016/j.cie.2015.02.018.

[21] S. A. Paul, M. C. Reddy, C. J. DeFlitch, A Systematic Review of Simulation Studies Investigating Emergency Department Overcrowding, SIMULATION 86 (2010) 559–571. doi:10.1177/0037549710360912.

[22] R. Konrad, K. DeSotto, A. Grocela, P. McAuley, J. Wang, J. Lyons, M. Bruin, Modeling the impact of changing patient flow processes in an emergency department: Insights from a computer simulation study, Operations Research for Health Care 2 (2013) 66–74. doi:10.1016/j.orhc.2013.04.001.

[23] C. Oh, A. M. Novotny, P. L. Carter, R. K. Ready, D. D. Campbell, M. C. Leckie, Use of a simulation-based decision support tool to improve emergency department throughput, Operations Research for Health Care 9 (2016) 29–39. doi:10.1016/j.orhc.2016.03.002.

[24] N. Martin, B. Depaire, A. Caris, The Use of Process Mining in Business Process Simulation Model Construction: Structuring the Field, Business & Information Systems Engineering 58 (2016) 73–87. doi:10.1007/s12599-015-0410-4.

[25] M. Armony, S. Israelit, A. Mandelbaum, Y. N. Marmor, Y. Tseytlin, G. B. Yom-Tov, On patient flow in hospitals: A data-based queueing-science perspective, Stochastic Systems 5 (2015) 146–194. doi:10.1214/14-SSY153.

[26] R. J. C. Bose, R. S. Mans, W. M. van der Aalst, Wanna improve process mining results?, in: 2013 IEEE Symposium on Computational Intelligence and Data Mining (CIDM), IEEE, Singapore, Singapore, 2013, pp. 127–134. doi:10.1109/CIDM.2013.6597227.

[27] M. G. Kahn, M. A. Raebel, J. M. Glanz, K. Riedlinger, J. F. Steiner, A Pragmatic Framework for Single-site and Multisite Data Quality Assessment in Electronic Health Record-based Clinical Research:, Medical Care 50 (2012) S21–S29. doi:10.1097/MLR.0b013e318257dd67.

[28] L. L. Pipino, Y. W. Lee, R. Y. Wang, Data quality assessment, Communications of the ACM 45 (2002) 211–218.

[29] R. Y. Wang, D. M. Strong, Beyond Accuracy: What Data Quality Means to Data Consumers, Journal of Management Information Systems 12 (1996) 5–33. doi:10.1080/07421222.1996.11518099.

[30] E. Rahm, H. H. Do, Data cleaning: Problems and current approaches, IEEE Data Engineering Bulletin 23 (2000) 3–13.

[31] W. Kim, B.-J. Choi, E.-K. Hong, S.-K. Kim, D. Lee, A taxonomy of dirty data, Data mining and knowledge discovery 7 (2003) 81–99.

[32] H. Mueller, J.-C. Freytag, Problems, methods, and challenges in comprehensive data cleansing, Professoren des Inst. Fr Informatik, 2005.

[33] J. Barateiro, H. Galhardas, A survey of data quality tools., Datenbank-Spektrum 14 (2005) 15–21.

[34] P. Oliveira, F. Rodrigues, P. R. Henriques, A formal definition of data quality problems., in: Proceedings of the 2005 International Conference on Information Quality, ICIQ 2005, 2005.

[35] T. Gschwandtner, J. Grtner, W. Aigner, S. Miksch, A taxonomy of dirty time-oriented data, in: International Conference on Availability, Reliability, and Security, Springer, 2012, pp. 58–72.

[36] R. S. Mans, W. M. P. van der Aalst, R. J. B. Vanwersch, Process Mining in Healthcare, SpringerBriefs in Business Process Management, Springer International Publishing, Cham, 2015. doi:10.1007/978-3-319-16071-9.

[37] K. I. Penny, I. Atkinson, Approaches for dealing with missing data in health care studies: *Dealing with missing data in health care studies*, Journal of Clinical Nursing 21 (2012) 2722–2729. doi:10.1111/j.1365-2702.2011.03854.x.

[38] N. Tsikriktsis, A review of techniques for treating missing data in OM survey research, Journal of Operations Management 24 (2005) 53–62. doi:10.1016/j.jom.2005.03.001.

[39] J. F. Hair (Ed.), Multivariate data analysis: a global perspective, 7. ed., global ed ed., Pearson, Upper Saddle River, NJ, 2010.

[40] K. Kerr, T. Norris, R. Stockdale, Data quality information and decision making: a healthcare case study, ACIS 2007 Proceedings (2007) 98.

[41] W. Abo-Hamad, Patient Pathways Discovery and Analysis Using Process Mining Techniques: An Emergency Department Case Study, in: P. Cappanera, J. Li, A. Matta, E. Sahin, N. J. Vandaele, F. Visintin (Eds.), Health Care Systems Engineering, volume 210, Springer

International Publishing, Cham, 2017, pp. 209–219. doi:10.1007/978-3-319-66146-9-19.

[42] S. Brailsford, J. Vissers, OR in healthcare: A European perspective, European Journal of Operational Research 212 (2011) 223–234. doi:10.1016/j.ejor.2010.10.026.

[43] J. W. Graham, Missing Data Analysis: Making It Work in the Real World, Annual Review of Psychology 60 (2009) 549–576. doi:10.1146/annurev.psych.58.110405.085530.

[44] J. W. Graham, Analysis of Missing Data, in: Missing Data, Springer New York, New York, NY, 2012, pp. 47–69. doi:10.1007/978-1-4614-4018-5_2.

[45] A. Rogge-Solti, R. S. Mans, W. M. van der Aalst, M. Weske, Repairing event logs using timed process models, in: OTM Confederated International Conferences" On the Move to Meaningful Internet Systems", Springer, 2013, pp. 705–708.

[46] H. Guo, D. Goldsman, K.-L. Tsui, Y. Zhou, S.-Y. Wong, Using simulation and optimisation to characterise durations of emergency department service times with incomplete data, International Journal of Production Research 54 (2016) 6494–6511. doi:10.1080/00207543.2016.1205760.

[47] Y.-H. Kuo, O. Rado, B. Lupia, J. M. Y. Leung, C. A. Graham, Improving the efficiency of a hospital emergency department: a simulation study with indirectly imputed service-time distributions, Flexible Services and Manufacturing Journal 28 (2016) 120–147. doi:10.1007/s10696-014-9198-7.

[48] T. A. Runkler, Data Analytics, Vieweg+Teubner Verlag, Wiesbaden, 2012. doi:10.1007/978-3-8348-2589-6.

[49] N. Martin, B. Depaire, A. Caris, Using Event Logs to Model Interarrival Times in Business Process Simulation, in: M. Reichert, H. A. Reijers (Eds.), Business Process Management Workshops, Lecture Notes in Business Information Processing, Springer International Publishing, 2016, pp. 255–267.

[50] L. Perimal-Lewis, D. Teubner, P. Hakendorf, C. Horwood, Application of process mining to assess the data quality of routinely collected time-based performance data sourced from electronic health records by validating process conformance, Health informatics journal 22 (2016) 1017–1029.