Project no. 826278

SERUMS

Research & Innovation Action (RIA)
SECURING MEDICAL DATA IN SMART PATIENT-CENTRIC HEALTHCARE SYSTEMS

Report on Initial Specification of Smart Patient Health Record Format
D2.1

Due date of deliverable: 30th April 2019

Start date of project: January 1st, 2019

Type: Deliverable
WP number: WP2

Responsible institution: Sopra-Steria Ltd.
Editor and editor's address: Andreas Francois Vermeulen, Euan Blackledge, Sopra-Steria Ltd.

Version 1.0

Project co-funded by the European Commission within the Horizon 2020 Programme		
Dissemination Level		
PU	Public	√
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	

Change Log

Rev.	Date	Who	Site	What
1	25/01/19	V Janjic	USTAN	Original Version
2	26/01/19	E Blackledge	SOPRA	Version 001.000
3	26/01/19	AF Vermeulen	SOPRA	Version 001.000

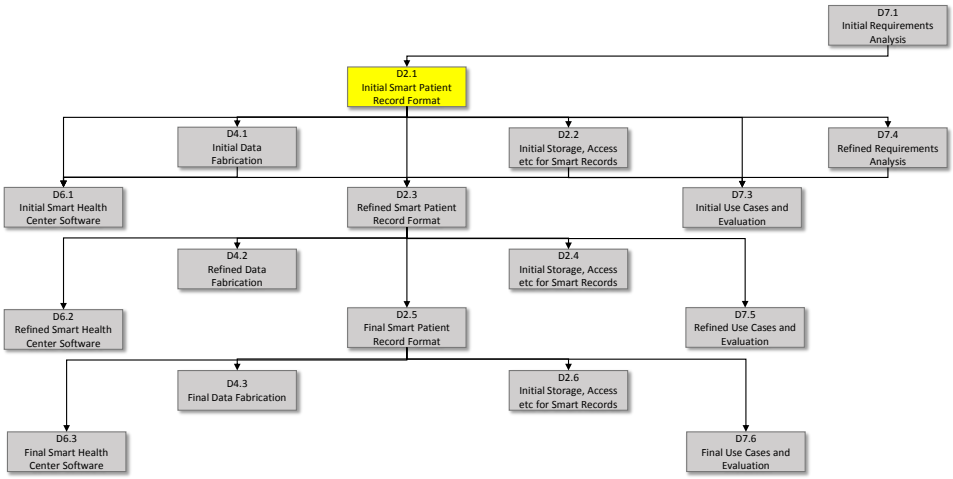


Figure 1: Dependencies between D2.1 and other deliverables

Executive Summary

This deliverable describes the initial version of the smart patient record format that will be used throughout the **Serums** project for encapsulating the patient data. It is a part of WP2, the objectives of which are to:

1. define a format for smart patient records that can contain data distributed over multiple devices
2. develop mechanisms to track the lineage of accesses to the smart patient records
3. to develop mechanisms to control storage and access rights for smart patient records
4. to build on existing and develop new machine learning methods for extracting meta-data from unstructured data

Precise specification of the format of the data that will be used in the project is crucial to many parts of the project, e.g. for the development of the block-chain technology to track the access to this data (WP2), distributed privacy-preserving data analytics (WP3), data fabrication technology to generate synthetic but realistic data to be used in development and testing of the other technologies in the project (WP4) and use cases development (WP7). Therefore, the results of this deliverable will be used extensively in the other parts of the project. Figure 1 shows the graph

of the dependencies between this deliverable and other relevant deliverables in the project.

In the subsequent deliverable related to the smart patient record format, we will extend the format presented in this deliverable to accommodate to the data that can reside on different sources within the same organisation (D2.3), as well as the data that might need to be collected/exchanged outside of the local environment, e.g. in scenarios of trans-national exchange of data (D2.5).

Contents

Chapter 1

Introduction

Smart Patient Record represents a central core for the information about a particular patient in the smart health centre system. It represents the central point where all the information about the patient is held, including both *static* personal data such as age, height, weight, date of birth etc. that is unlikely to change, and *dynamic* data related to e.g. the treatments the patient is undertaking, screenings and prescribed medications. The main challenge for the **Serums** project is to deal with *decentralisation* of the information related to a particular patient. In modern health-care systems, the data about a single patient may reside on different subsystems in the same health-care centre, or even, in some scenarios, scattered across different health-care institutions. It will also be collected from a variety of devices, some of which (e.g. personal monitoring devices) will need to communicate with the health-care systems over open networks. We need to be able to represent all this data in a standardised way, taking into account the possible geographical distribution of both where the data is stored and where is it collected from.

It is essential to derive a standard, precise and machine readable format of the data related to a single patient. This also includes all the meta-data associated with the data (e.g. whether the data is local or remote, how it is accessed etc). The smart patient record format also need to provide possibility of different types of access rights for different entities (patients, general practitioners, specialists) who will be accessing the data. Ideally, we want to derive a format that will cover different use cases used in the **Serums** project. This would allow generic distributed data analytics mechanism (which is the objective of WP3) to be performed on the medical data. Additionally, it would allow successful data fabrication (the objective of WP4), giving us access to a vast amount of *realistic* data that will have the same format as the real data (in terms of the fields used, ranges and distribution of values in the fields and correlation between different fields), but will be synthetic, without the possibility of being associated with any real data, and therefore not being subject of privacy and ownership concerns. This data will be used throughout the project both for developing new technologies and for stress-testing them on large volumes of data. Finally, the precise format of the data is essential for storage and

access mechanisms.

This deliverable represents a first step towards deriving a uniform **Serums** smart patient record format. Here, we are focusing specifically on the *centralised* data, where all of the patient data is available at the same place. In the subsequent deliverable, D2.3 and D2.5, we will extend this format to support distributed data. We propose the *data vault* as an appropriate generic format in which the data will be kept. Data vaults are recognised in data science as a universal format that allows easier and more automatic data analytics. Our ultimate goal is to be able to represent data of all use cases in form of data vaults. Here, we describe the initial version of the proposed format. We first briefly describe the use case on which the initial version of the format was based (Chapter 2). We then discuss the universal data-vault format in Chapter 3, followed by the description of the specific format of the Edinburgh Cancer Data Gateway use case and the ways in which this can be converted into the data-vault format. Finally, we provide more information about the implementation of the universal data format in a form that can be used by the other parts of the **Serums** infrastructure (Chapter 4).

Chapter 2

Use Case Description

The core use of the Smart Patient Electronic Health Record (SPEHR) is to create a singular electronic health record that the patient has autonomous control over as per General Data Protection Regulation (GDPR). [2] One of the objectives of the **Serums** project is to understand what is required to derive a *universal* Smart Patient Health Record format that will be applicable to a wide variety of different use cases that manipulate patient data, as well as what techniques and methodologies are required for a practical implementation of this format. We are especially concerned with providing the patient a possibility to grant specific granular-level access to the record to the approved service providers and be able to revoke the access if required.

In this section, we describe the use case that we have used to derive an initial version of the smart patient record format. Our focus was on the use case provided by the **USTAN** project partner, but we are aiming to show that the proposed format in Section 3 is also applicable to the other use cases considered in the project.

2.1 Edinburgh Cancer Data Gateway with integrated Patient Reported Outcome Measures (PROMS)

We aim to build a predictor within NHS Lothian for toxicity levels from treatment regimens for cancer patients with or without comorbidities. Giving patients the opportunity to give more accurate information on their symptoms throughout the treatment (possibly daily whilst at home), and combining that information with patient characteristics, cancer information and treatment regimen, will allow clinicians to adapt treatments better to individual patients with better patient outcomes and controlled toxicity levels. Further data from hospitalisations and comorbidities will contribute to a more accurate prediction.

The Edinburgh Cancer Centre (ECC), based in the Western General Hospital (WGH), contains National Health Service Lothian (NHS Lothian) cancer patient data from multiple resources, scattered across different systems and platforms. This makes it difficult to use the information collected in a useful way. There

is a lack of proxy between the different (sub)systems or mechanisms to join the information automatically. This information includes some coming directly from oncology (Chemocare, SES Oncology DB, SMR6, etc) but also data on hospitalisations (SMR1), cause of death (NRS COD), Charlson Comorbidity Index (CCI), prescribing and other patient information. When this data is combined, it can be used to give a more accurate picture as to how patients are treated and how their treatment could be improved to consider comorbidities, reduce toxicity, serious side effects, etc. Chemotherapy often leads to increased toxicity levels and it may be desirable to compare the likelihood of increased toxicity for certain patients on given treatment regimes in the presence of comorbidities, especially when these comorbidities introduce further medications that exacerbate toxicity levels. Our Cancer Data Gateway aims to improve the quality and capability of reporting outcomes within South East Scotland Oncology databases in real time using routinely captured and integrated electronic healthcare data.

The development of our use case will be done in three stages:

1. We will use oncology data, SMR1, NRS COD and CCI as the comorbidity indicator for the predictor of toxicity

2. Where instead of CCI we add detailed prescribing data to more accurately predict toxicity of chemotherapy treatments when taking further medications

3. Adding further patient information gained from Patient Reported Outcome Measures (PROMS)

We will develop a simple and user-friendly web application for patients to provide information on symptoms, etc, on a daily basis. Usually PROMS are based on questionnaires, and we will discuss with the clinical team how these questionnaires will be designed and integrated in a web-based approach. This information will be integrated with other information available on the patient, and enable clinical staff to change/add medications and decisions on patient treatments. We want to add PROMS as a mechanism to further enrich the information available on the system to symptoms that patients have in between treatments to have a more accurate picture as to how their toxicity would increase. It is known that PROMS can be used to improve the quality of life and even survival for some forms of cancer.

We will develop a model based on advanced analysis of this data to enable us to hand over the metadata structure to IBM to assist them to synthesise data. It is this synthesised data that will be used to build and test the access rights and the machine learning models in order to protect patient privacy.

Chapter 3

Smart Patient Health Format

In this chapter, we discuss the concept of Universal Smart Patient Health Record, the aim of which is to provide a general way of representing the patient data that will cover a variety of uses and scenarios, and that will also be machine readable to allow its usage in data fabrication and blockchain technology. In Section 3.1, we discuss requirements for format of such universal patient record. We then (Section 3.2) introduce a well-known concept from data science, *data vault*, which is a standard representation of the data in big data systems. In Section 3.2.1, we describe a specific instance of a data vault, T-P-O-L-E data vault, which we propose to use as the format for the patient data in the **Serums** project. We then (Section 3.3) describe the format in which the data of the use case described in Chapter 2 was originally stored, create a database schema out of it and discuss how the data can be reorganised into data vault. Finally, in Section 3.4, we discuss how we can establish correlation between values of different fields of the patient records (e.g. whether particular value in one field necessarily implies a particular value, or a range of values, in other field), which is necessary for deriving precise rules required for data fabrication infrastructure of WP4, for generating synthetic but realistic data.

3.1 Universal Smart Patient Health Record (USPHR)

In any medical organisation, it is essential to keep a good organisation of the data about individual patients. Besides the usual requirements for this data, in terms of being easily accessible and well structured, new legislations such as GDPR put additional requirements in terms of the organisation, ownership, access and communication of this data. The data is owned by the patient themselves and they need to be able to have full access to it, while the other parties (general practicioners, specialists, insurers) must have access to only the parts of the data that are releveant to diagnostics, treatment and insurance. Currently, there is no unified way to represent the patient data across different organisations. Each individual organisation (e.g. National Health Service in the UK) might have their own way of organising the data. Quite frequently, this data is not even in the electronic format. This ef-

fectively prevents developing any generic mechanisms for storing, communicating and analysing the medical data, as each solution is tied up to a specific format of the data that is used by a particular organisation.

Universal Smart Patient Health Record (USPHR) represents a first attempt of a unified mechanism to create and manage electronic patient records for individual patients in a secure and transparent way, centralising a golden source of health information. It will overcome the current challenges arising from managing different formats of the diverse electronic patient records present in different institutions/scenarios by creating a universal format that will be applicable in a variety of situations. We have the following requirements for the format of USPHR:

- it needs to be *general* enough so that it can encapsulate different use cases in the **Serums** project and wider;
- it needs to be *specific* enough that it allows generation of machine-readable metadata that can be used for data fabrication and other parts of the **Serums** infrastructure;
- it needs to be *flexible* enough to allow different levels of access by different interested parties, as well as the usage of blockchain to record lineage and provenance of the data;

The USPHR will provide a fundamental core structure for the mechanisms from WP3, WP4, and WP5.

3.2 Data Vault

Data vault is a set of guidelines on how to how to organise the tables and relationships within a database system. As such can work on top of any database system such as PostgreSQL, a distributed file system such as Hadoop, or even a NoSQL database such as MongoDB. The data vault model itself puts priority on providing long-term historical storage of data from multiple operational systems. The base design principal is single version of the truth (SVOT) that supports data warehousing for either a single centralised database or a distributed synchronised database. This method is accommodating of change. Its adaptability provides resilience over time to changes in the data and data structures. The data vault methodology is based on SEI/CMMI Level 5 best practices.

In practice, the data vault has been designed to take in data from disparate sources and systems and centralise the data in a non-destructive manner. End users do not interact directly with the raw data in the database, they instead interact with a layer of cleansed data which is produced on top in the form of a data warehouse. This means that data is never dropped from the source, even if it is thought to be erroneous, ensuring that full auditablility is built into the system.

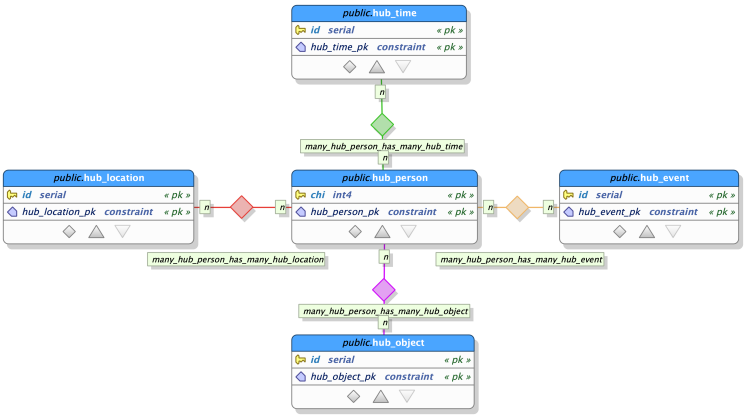


Figure 3.1: Data Vault - T-P-O-L-E Hubs

In our use cases, the hospitals have their data spread across numerous systems within the hospitals' computer infrastructures. In the production of the Smart Patient Health Record (SPHR) we are looking to centralise this data in such a way that it maintains its history. As the project develops, new data sources will be added to the SPHR. In a traditional database model this would require the structure of the database to be altered in order to accommodate this change. With the data vault model however, new sources are easily added into the model with no need to alter the existing foundation.

The data vault model is comprised of three different types of tables:

- *hubs*, that contain a list of unique business keys with low propensity to change (Figure 3.1);
- *links* that represent associations or transactions between business keys (Hubs) (Figure 3.2);
- *satellites*, where temporal and descriptive attributes of the hubs and links are stored (Figure 3.3);

Reference tables are a standard part of a healthy data vault model. They hold look-ups and standards for common codes within the complete data vault. This is the area where the standards are stored ready for the data vault to use during the upload.

3.2.1 T-P-O-L-E Data Vault

The Time-Person-Object-Location-Event (T-P-O-L-E) Data Vault is a highly specific data vault that is suitable to a variety of business use cases. This is the instance of the data vault that we propose to use as a universal format for smart health patient records. In T-P-O-L-E data vault, there are five types of hubs:

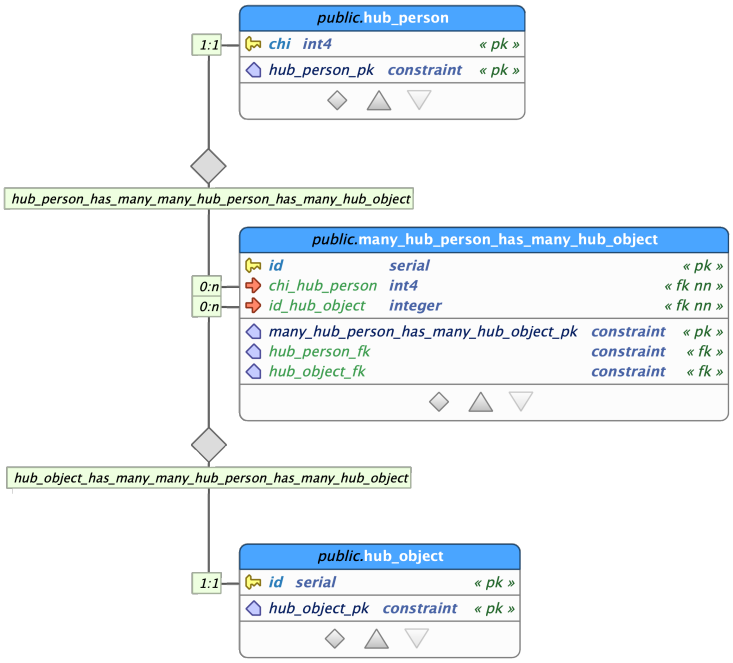


Figure 3.2: Data Vault - T-P-O-L-E Links

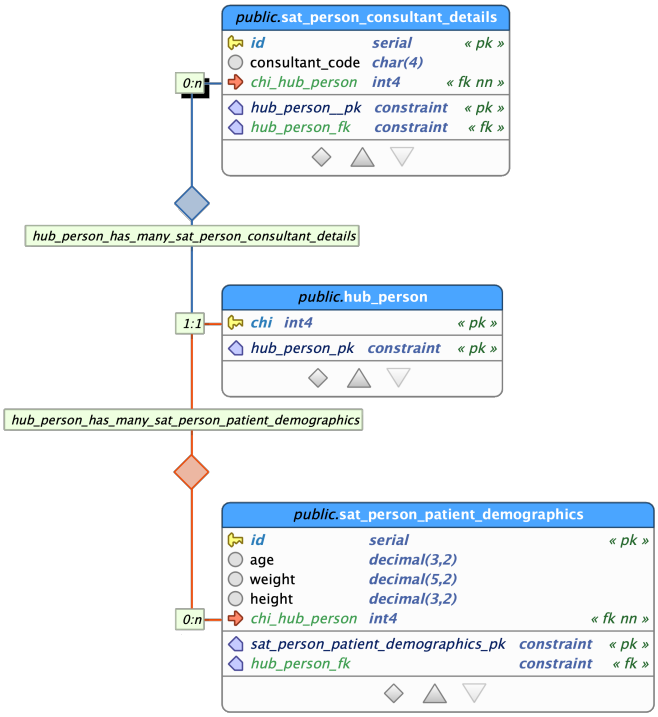


Figure 3.3: Data Vault - T-P-O-L-E Satellites

- *Time* hub, where the dates and times of events are stored. It prescribes usage of ISO 8601-1:2019 and ISO 8601-2:2019 as standards for time/date representation.

- *Person* hub, where information about persons is stored using the concept of "Golden Nominal". This type of record is a single person record with a unique reference to that person.

- *Object* hub, where the objects who represent any other referable entitites involved in the use case are stored. Examples of the objects are:

  - Organisations (Legal entity of Bank, Hospital or School)
  - Physical objects (Bank Card, Vehicle or hospital bed)
  - Buildings (Physical Bank, Hospital or School)

- *Location* hub, where coordinates of the locations are stored. The precision of the coordinates is within one thousandth of a degree of latitude and longitude (i.e. to less than one square meter) using ISO 6709:2008 Standard[1]. Note that the location can store different descriptions via location satellites for the same location, e.g. a location could be a building's location, physical address and a bus stop on a bus route at the same time.

- *Event* hub, where the individual actions or events are stored.

As is usual in data vaults, links connect elements in different hubs. These links are different in each specific implementation of the T-P-O-L-E data vault model, but we also require them to have a highly-descriptive meta-data to denote what is stored in the links and how they are stored. The satellites are connected to T-P-O-L-E hubs to add the temporal and descriptive attributes and descriptive attributes. These satellites holds all the attributes for a specific type of hub, storing the values that were valid for data in the hub between two specific points in time.

Note that data from data vault is never removed or modified as it is immutable. To reflect the new state of the data, we insert a transaction into the data vault. This allows tracking of a complete history of the states through which the data has moves, ensuring we can have full provenance in lineage of the data.

[1] https://www.iso.org/standard/39242.html

3.3 Data Vault for Edinburgh Cancer Data Gateway

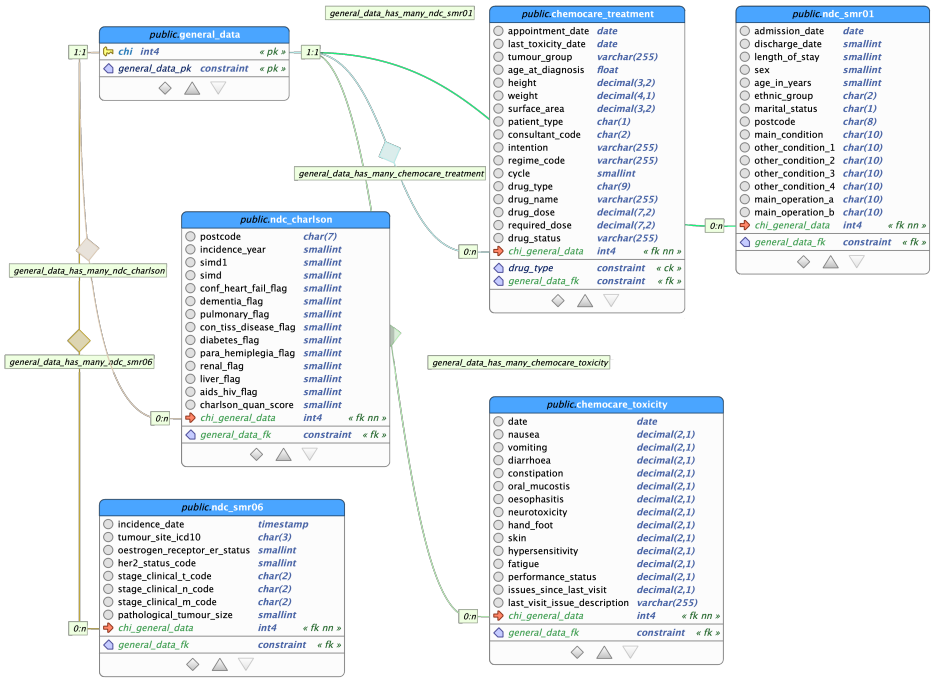


Figure 3.4: ERD for Edinburgh Cancer Data Gateway

The Entity Relationship Diagram (ERD) on Figure 3.4 shows the details about the structure of the data from the Edinburgh Cancer Data Gateway use case. The use case contains some static data about each patient, which is the data that will remain mostly unchaged for the duration of chemotherapy. It is stored in several tables (`general_data`, `ndc_charison` and `ndc_smr01`) and contains, for example, the identifier number, age, age at a diagnosis of a tumor, ethnic, postcode group and so on. There is also dynamic data about chemocare treatments and toxicity (`chemocare_treatment` and `chemocare_toxicity`), where new records are added after each chemotherapy session.

To build the data vault from this data, we have used the *Hierarchical Data Format* [1] (HDF, version 5). The basic design principle for HDF5 is to store and organise large volumes of data (See Figure 3.5). We have used the H5PY[2] library to build the Edinburgh Cancer Data Gateway data vault. The hubs of this data vault are:

- *Time*. We suport ISO 8601-1:2019[3] and ISO 8601-2:2019[4] notation for date

[2] https://pypi.org/project/h5py/
[3] https://www.iso.org/obp/ui#iso:std:iso:8601:-1:ed-1:v1:en
[4] https://www.iso.org/obp/ui#iso:std:iso:8601:-2:ed-1:v1:en

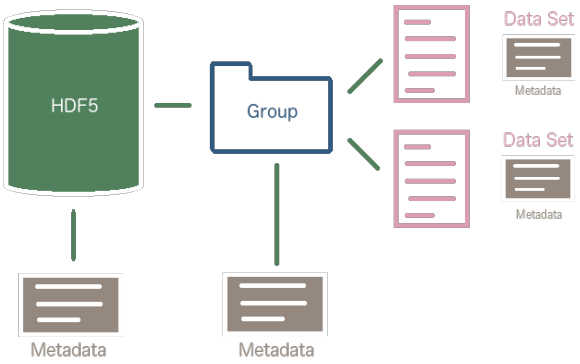


Figure 3.5: Hierarchical Data Format (HDF) version 5

format. This represents date and time, together with the timezone (yyyy-mm-ddThh:mm:ss.nnnnnn+|-hh:mm).

- *Person*. This hub contains core information about the persons involved in the health-care process. These can be patient, medical staff or other authorised persons. The role of the person is determined by the person role type in the hub.

- *Object*. This hub contains information about objects involved in the health-care process. These can be blood samples, hospital beds etc. The role of the object is determined by the object role type in the hub.

- *Location*. This hub contains information about the locations involved in the health-care process. We support the ISO 6709:2008 standard[5]. The role of the location is determine by a location role type in the hub. We are currently assuming a single type, latitude-longitude.

- *Event*. This hub contains information about the events that happen in the use case. Examples of the events are chemotherapy treatment, visit to a doctor, medical test, answers to a questionaire etc.

This is an instance of the T-P-O-L-E data vault that we described in Section 3.2.1. The HDF5 structure of it is shown in Figure 3.7, while the ERD diagram is shown in Figure 3.6.

[5] https://www.iso.org/standard/39242.html

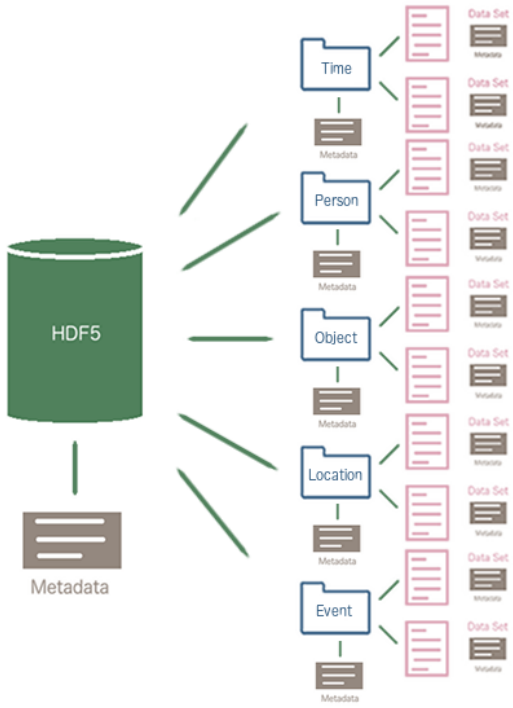


Figure 3.7: T-P-O-L-E HDF version 1.00

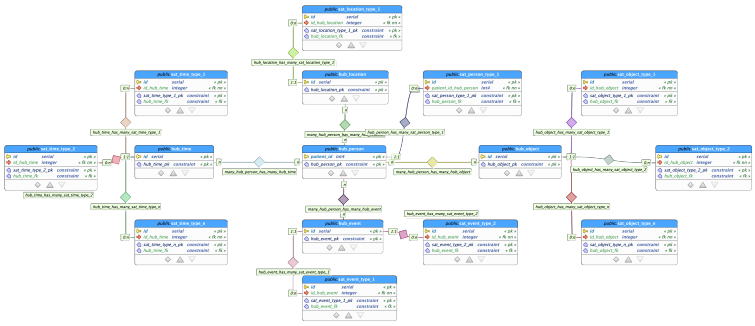


Figure 3.6: T-P-O-L-E for Edinburgh Cancer Data Gateway

3.4 Correlation Between Attributes of Smart Patient Medical Data Records

In order to develop the data fabrication infrastructure in WP4, which will be able to generate large volumes of synthetic but realistic data that will be used throughout the project, we need to know the precise structure of the smart patient records, in

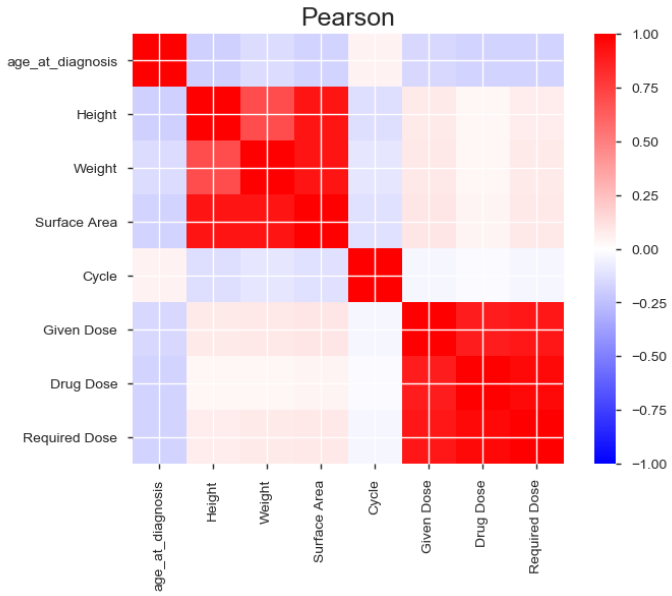


Figure 3.8: Correlations between different fields in the Edinburgh Cancer Data Gateway use case

terms of the format of the data, types of attributes, possible values that they can take and the distribution of values. For example, the Edinburgh Cancer Data Gateway focuses mostly on the patients with breast cancer, so we expect that the age of the patients in the `age` field of the smart patient record will take values between 18 and 90, with a normal distribution and a mean value of 62 (the mean age at which the breast cancer is diagnosed). In addition to this, we will also need to know correlation between any attributes that are linked in some way. For example, looking at the Figure 3.4, we can see that the `chemocare_toxicity` has `nausea` and `vomiting` fields. It is not possible for a patient to have positive value for `vomitting` without also having a positive value for `nausea`. Also, `diarrhoea` and `constipation` fields are mutually exclusive.

For the Edinburgh Cancer Data Gateway, we have used Pandas-Profiling[6] tool to analyse the raw data that we have received. Pandas-Profiling has the capability of establishing strong and weak correlations between different fields of the records, and representing them in a variety of formats. Figure 3.8 shows an example of the visualisation of correlation between different fields of the Edinburgh Cancer Data Gateway records. We will use this to precisely formulate the relationship between fields and use this as an input to the data fabrication technology.

[6] https://anaconda.org/conda-forge/pandas-profiling

Chapter 4

Technical Implementation

This chapters give more technical details about the implementation of the concepts we have illustrated in Chapter 3.

4.1 Process overview

- Member of use case team with access to the data runs pandas-profiling and produces output HTML files (see Chapter A.4)
- This is high level overview of the structure of the data, covering common and outlier values, correlations between columns, and data distribution
- With this we can then model the data using pgModeler. This allows us to abstract out the current table definitions into the Data Vault framework (See Section 3.2)
- pgModeler also generates the PostgreSQL tables, SQL files, XML schemas, and image files
- From the PostgreSQL tables we can then generate the structure of the HDF5 dataset using the Python program
- The only thing that needs to be supplied at this point are the table names in the database and the desired name of the output dataset, the rest is automatic
- The resulting output is then ready to receive the data
- This Hierarchical Data Format 5 (HDF5) dataset is the Smart Patient Record

4.2 The thinking

By generating the Smart Patient Record in this format we are able to create a separate record per patient that can be independently stored, encrypted, and transmitted.

Furthermore, the design is entirely modular so can be expanded upon as and when new data sources become available, or no longer permitted.

The HDF5 file itself is naturally compressible during its creation. It can then be compressed even further before transmission which will allow for tiny data packets to be sent over whichever distribution method is chosen. Further advantages are the scalability of the technology and the reliability built into the distributed storage.

4.3 Open source

Everything that has been chosen to generate the Smart Patient Record so far has been open source and covered by either the MIT or GPL license. This brings the major advantage of the cost being limited to man-hours. Another advantage is the flexibility that this allows for, with us being able to custom design the processes, implementing any level of security or enforcing any level of standard that the project sees fit.

We will however have to be aware that end users will not always feel comfortable about allowing this type of software onto their systems. Once we have the processes in place we believe that we will be able to recreate them under other system restraints that may be more representative of the medical space as a whole.

4.4 The technology

The creation of the Smart Patient Record relies on three technologies for the creation of the record: PostgreSQL, Python, and HDF5. Additionally we will use Hadoop for the storage of the records.

PostgreSQL has been chosen primarily due to the modelling software that is available for it. pgModeler is being developed by Raphael A. Silva and is available under the GPL v3 license. By modelling with this software we are able to produce a validated SQL file that we will pass over to IBM as a start point for their data fabrication.

Python is the lynchpin in this trio, with libraries written specifically to profile data, handle connections to PostgreSQL, and output data to HDF5. This enables us to focus our attention on the process and functionality, automating as much as possible.

HDF5 offers us a key advantage going forward. As well as being designed to work with large, complex datasets, we will also be able to add metadata to each of the datasets. This will allow us to add in extra information about the data, aiding in the upcoming machine learning parts of the work package.

HDF5 on Hadoop (HDFS) is ultimately how we store the Smart Patient Record. This offers us many advantages throughout the life of the project. As well as being a flexible and scalable storage solution, Hadoop can be integrated into platforms such as Microsoft Azure, Amazon Web Services (AWS), IBM Cloud, and Google

Cloud. With Accenture's preference for AWS, we will aim for that as our primary target platform.

4.5 Next stage

With the format now defined, this PostgreSQL database schema will be handed over to IBM to begin the process of data fabrication. This will involve a process of rule refining as we work alongside our use case supplier to fine tune the constraints of the data, working towards as realistic facsimile of the source data as possible.

Once this data has been fabricated, we will use this to begin building and training our machine learning models as part of WP2.2.

4.6 SQL Scripts

The following SQL script have been prepared to assist the members of the consortium to create the required SQL databases in data engine of their choose.

4.6.1 SQL for PostgreSQL

```
CREATE TABLE `hub_person` (
  `chi` integer,
  PRIMARY KEY (`chi`)
);

CREATE TABLE `hub_event` (
  `id` integer,
  PRIMARY KEY (`id`)
);

CREATE TABLE `many_hub_person_has_many_hub_event` (
  `chi_hub_person` integer,
  `id_hub_event` integer,
  `id` integer,
  PRIMARY KEY (`id`),
  KEY `FK` (`chi_hub_person`, `id_hub_event`)
)
```

4.6.2 SQL for MySQL

```
CREATE TABLE `hub_person` (
  `chi` integer,
  PRIMARY KEY (`chi`)
);

CREATE TABLE `hub_event` (
  `id` integer,
  PRIMARY KEY (`id`)
);

```

```
CREATE TABLE `many_hub_person_has_many_hub_event` (
  `chi_hub_person` integer,
  `id_hub_event` integer,
  `id` integer,
  PRIMARY KEY (`id`),
  KEY `FK` (`chi_hub_person`, `id_hub_event`)
)
```

4.7 XML

```

<table name="hub_person" layer="0" collapse-mode="2" max-obj-count="1">
   <schema name="public"/>
   <role name="postgres"/>
   <position x="440" y="180"/>
   <column name="chi" alias="PATIENT IDENTIFICATION NUMBER" not-null="true
      (cont.)"
    identity-type="ALWAYS">
      <type name="int4" length="0"/>
   </column>
   <constraint name="hub_person_pk" type="pk-constr" table="public.
      (cont.)hub_person">
      <columns names="chi" ref-type="src-columns"/>
   </constraint>
</table>
```

5. Conclusion

In this deliverable, we described the initial version of the Smart Patiend Data Record format that aims to provide a generic way of describing the patient data in medical data science use cases. Our aim was to provide a format that will be generic enough to encompass different scenarios and the use cases we will develop over the course of the **Serums** project, but also precise enough that it can be described in machine-readable language. This is essential for the blockchain and data fabrication and blockchain technology that will be deveoped in WP2 and WP4, which will use this format as an input. We have proposed using data vault concept as a universal format of the medical data, and we have demonstrated how this format can be used with Edinburgh Cancer Data Gateway use case, one of the use cases we will use in the **Serums** project.

The current *Smart Patient Record* (Section 3) is version 001.000 and will evolve over the period of the research of the **Serums** project. The team for *Smart Patient Record* version 001.001 plan to include other research use cases identified by the consortium. The *Smart Patient Record* will evolve as the researchers add extra use cases but WP2 do not foresee any major change to the Universal Smart Patient Record's core structure.

Appendix A

Scripts

A.1 Full Cancer Informatics SQL for PostgreSQL

The following SQL scripts for PostgreSQL will deploy the empty Cancer.Informatics schema that is required to store the source system data for the data crawlers to process into the Smart Health Data Vault (SHDV)

```
1
2  -- Database generated with pgModeler (PostgreSQL Database Modeler).
3  -- pgModeler version: 0.9.2-alpha1
4  -- PostgreSQL version: 11.0
5  -- Project Site: pgmodeler.io
6  -- Model Author: ---
7
8
9  -- Database creation must be done outside a multicommand file.
10 -- These commands were put in this file only as a convenience.
11 -- -- object: dv_chemocare_treatment_db | type: DATABASE --
12 -- -- DROP DATABASE IF EXISTS dv_chemocare_treatment_db;
13 -- CREATE DATABASE dv_chemocare_treatment_db;
14 -- -- ddl-end --
15 --
16
17 -- object: public.hub_person | type: TABLE --
18 -- DROP TABLE IF EXISTS public.hub_person CASCADE;
19 CREATE TABLE public.hub_person (
20    chi int4 NOT NULL GENERATED ALWAYS AS IDENTITY ,
21    CONSTRAINT hub_person_pk PRIMARY KEY (chi)
22
23 );
24 -- ddl-end --
25 ALTER TABLE public.hub_person OWNER TO postgres;
26 -- ddl-end --
27
28 -- object: public.hub_time | type: TABLE --
29 -- DROP TABLE IF EXISTS public.hub_time CASCADE;
30 CREATE TABLE public.hub_time (
31    id serial NOT NULL,
32    CONSTRAINT hub_time_pk PRIMARY KEY (id)
33
34 );
35 -- ddl-end --
```

```
ALTER TABLE public.hub_time OWNER TO postgres;
-- ddl-end --

-- object: public.sat_person_patient_measurements | type: TABLE --
-- DROP TABLE IF EXISTS public.sat_person_patient_measurements CASCADE;
CREATE TABLE public.sat_person_patient_measurements (
    id serial NOT NULL,
    age_at_diagnosis float,
    height decimal(3,2),
    weight decimal(4,1),
    surface_area decimal(3,2),
    chi_hub_person int4 NOT NULL,
    CONSTRAINT sat_person_patient_measurements_pk PRIMARY KEY (id)

);
-- ddl-end --
ALTER TABLE public.sat_person_patient_measurements OWNER TO postgres;
-- ddl-end --

-- object: public.hub_object | type: TABLE --
-- DROP TABLE IF EXISTS public.hub_object CASCADE;
CREATE TABLE public.hub_object (
    id serial NOT NULL,
    CONSTRAINT hub_object_pk PRIMARY KEY (id)

);
-- ddl-end --
ALTER TABLE public.hub_object OWNER TO postgres;
-- ddl-end --

-- object: public.sat_time_appointment_date | type: TABLE --
-- DROP TABLE IF EXISTS public.sat_time_appointment_date CASCADE;
CREATE TABLE public.sat_time_appointment_date (
    id serial NOT NULL,
    appointment_date date,
    id_hub_time integer NOT NULL,
    CONSTRAINT sat_time_appointment_date_pk PRIMARY KEY (id)

);
-- ddl-end --
ALTER TABLE public.sat_time_appointment_date OWNER TO postgres;
-- ddl-end --

-- object: public."sat_time_last-toxicity_date" | type: TABLE --
-- DROP TABLE IF EXISTS public."sat_time_last-toxicity_date" CASCADE;
CREATE TABLE public."sat_time_last-toxicity_date" (
    id serial NOT NULL,
    last_toxicity_date date,
    id_hub_time integer NOT NULL,
    CONSTRAINT "sat_time_last-toxicity_date_pk" PRIMARY KEY (id)

);
-- ddl-end --
ALTER TABLE public."sat_time_last-toxicity_date" OWNER TO postgres;
-- ddl-end --

-- object: public.sat_object_tumour_group | type: TABLE --
-- DROP TABLE IF EXISTS public.sat_object_tumour_group CASCADE;
CREATE TABLE public.sat_object_tumour_group (
    id serial NOT NULL,
    tumour_group varchar(255),
    id_hub_object integer NOT NULL,
```

```
    CONSTRAINT sat_object_tumour_group_pk PRIMARY KEY (id)

);
-- ddl-end --
ALTER TABLE public.sat_object_tumour_group OWNER TO postgres;
-- ddl-end --

-- object: public.sat_person_consultant_chemo | type: TABLE --
-- DROP TABLE IF EXISTS public.sat_person_consultant_chemo CASCADE;
CREATE TABLE public.sat_person_consultant_chemo (
    id serial NOT NULL,
    consultant_code char(2),
    chi_hub_person int4 NOT NULL,
    CONSTRAINT sat_person_consultant_chemo_pk PRIMARY KEY (id)

);
-- ddl-end --
ALTER TABLE public.sat_person_consultant_chemo OWNER TO postgres;
-- ddl-end --

-- object: public.hub_event | type: TABLE --
-- DROP TABLE IF EXISTS public.hub_event CASCADE;
CREATE TABLE public.hub_event (
    id serial NOT NULL,
    CONSTRAINT hub_event_pk PRIMARY KEY (id)

);
-- ddl-end --
ALTER TABLE public.hub_event OWNER TO postgres;
-- ddl-end --

-- object: public.sat_object_drug_details | type: TABLE --
-- DROP TABLE IF EXISTS public.sat_object_drug_details CASCADE;
CREATE TABLE public.sat_object_drug_details (
    id serial NOT NULL,
    drug_type char(9),
    drug_name varchar(255),
    given_drug decimal(72),
    drug_dose decimal(7,2),
    required_dose decimal(7,2),
    id_hub_object integer NOT NULL,
    CONSTRAINT sat_object_drug_details_pk PRIMARY KEY (id)

);
-- ddl-end --
ALTER TABLE public.sat_object_drug_details OWNER TO postgres;
-- ddl-end --

-- object: public.sat_object_treatment_details | type: TABLE --
-- DROP TABLE IF EXISTS public.sat_object_treatment_details CASCADE;
CREATE TABLE public.sat_object_treatment_details (
    id serial NOT NULL,
    intention varchar(255),
    regime varchar(255),
    cycle int4,
    id_hub_object integer NOT NULL,
    CONSTRAINT sat_object_treatment_details_pk PRIMARY KEY (id)

);
-- ddl-end --
ALTER TABLE public.sat_object_treatment_details OWNER TO postgres;
-- ddl-end --
```

```
160
161 -- object: public.sat_event_drug_status | type: TABLE --
162 -- DROP TABLE IF EXISTS public.sat_event_drug_status CASCADE;
163 CREATE TABLE public.sat_event_drug_status (
164     id serial NOT NULL,
165     drug_status varchar(255),
166     id_hub_event integer NOT NULL,
167     CONSTRAINT sat_event_drug_status_pk PRIMARY KEY (id)
168
169 );
170 -- ddl-end --
171 ALTER TABLE public.sat_event_drug_status OWNER TO postgres;
172 -- ddl-end --
173
174 -- object: public.many_hub_person_has_many_hub_time | type: TABLE --
175 -- DROP TABLE IF EXISTS public.many_hub_person_has_many_hub_time CASCADE;
176 CREATE TABLE public.many_hub_person_has_many_hub_time (
177     chi_hub_person int4 NOT NULL,
178     id_hub_time integer NOT NULL,
179     id serial NOT NULL,
180     CONSTRAINT many_hub_person_has_many_hub_time_pk PRIMARY KEY (id)
181
182 );
183 -- ddl-end --
184
185 -- object: hub_person_fk | type: CONSTRAINT --
186 -- ALTER TABLE public.many_hub_person_has_many_hub_time DROP CONSTRAINT IF
        (cont.) EXISTS hub_person_fk CASCADE;
187 ALTER TABLE public.many_hub_person_has_many_hub_time ADD CONSTRAINT
        (cont.)hub_person_fk FOREIGN KEY (chi_hub_person)
188 REFERENCES public.hub_person (chi) MATCH FULL
189 ON DELETE RESTRICT ON UPDATE CASCADE;
190 -- ddl-end --
191
192 -- object: hub_time_fk | type: CONSTRAINT --
193 -- ALTER TABLE public.many_hub_person_has_many_hub_time DROP CONSTRAINT IF
        (cont.) EXISTS hub_time_fk CASCADE;
194 ALTER TABLE public.many_hub_person_has_many_hub_time ADD CONSTRAINT
        (cont.)hub_time_fk FOREIGN KEY (id_hub_time)
195 REFERENCES public.hub_time (id) MATCH FULL
196 ON DELETE RESTRICT ON UPDATE CASCADE;
197 -- ddl-end --
198
199 -- object: public.many_hub_person_has_many_hub_object | type: TABLE --
200 -- DROP TABLE IF EXISTS public.many_hub_person_has_many_hub_object CASCADE
        (cont.);
201 CREATE TABLE public.many_hub_person_has_many_hub_object (
202     chi_hub_person int4 NOT NULL,
203     id_hub_object integer NOT NULL,
204     id serial NOT NULL,
205     CONSTRAINT many_hub_person_has_many_hub_object_pk PRIMARY KEY (id)
206
207 );
208 -- ddl-end --
209
210 -- object: hub_person_fk | type: CONSTRAINT --
211 -- ALTER TABLE public.many_hub_person_has_many_hub_object DROP CONSTRAINT
        (cont.)IF EXISTS hub_person_fk CASCADE;
212 ALTER TABLE public.many_hub_person_has_many_hub_object ADD CONSTRAINT
        (cont.)hub_person_fk FOREIGN KEY (chi_hub_person)
213 REFERENCES public.hub_person (chi) MATCH FULL
214 ON DELETE RESTRICT ON UPDATE CASCADE;
```

```
215 -- ddl-end --
216
217 -- object: hub_object_fk | type: CONSTRAINT --
218 -- ALTER TABLE public.many_hub_person_has_many_hub_object DROP CONSTRAINT
       (cont.)IF EXISTS hub_object_fk CASCADE;
219 ALTER TABLE public.many_hub_person_has_many_hub_object ADD CONSTRAINT
       (cont.)hub_object_fk FOREIGN KEY (id_hub_object)
220 REFERENCES public.hub_object (id) MATCH FULL
221 ON DELETE RESTRICT ON UPDATE CASCADE;
222 -- ddl-end --
223
224 -- object: public.many_hub_person_has_many_hub_event | type: TABLE --
225 -- DROP TABLE IF EXISTS public.many_hub_person_has_many_hub_event CASCADE;
226 CREATE TABLE public.many_hub_person_has_many_hub_event (
227     chi_hub_person int4 NOT NULL,
228     id_hub_event integer NOT NULL,
229     id serial NOT NULL,
230     CONSTRAINT many_hub_person_has_many_hub_event_pk PRIMARY KEY (id)
231
232 );
233 -- ddl-end --
234
235 -- object: hub_person_fk | type: CONSTRAINT --
236 -- ALTER TABLE public.many_hub_person_has_many_hub_event DROP CONSTRAINT
       (cont.)IF EXISTS hub_person_fk CASCADE;
237 ALTER TABLE public.many_hub_person_has_many_hub_event ADD CONSTRAINT
       (cont.)hub_person_fk FOREIGN KEY (chi_hub_person)
238 REFERENCES public.hub_person (chi) MATCH FULL
239 ON DELETE RESTRICT ON UPDATE CASCADE;
240 -- ddl-end --
241
242 -- object: hub_event_fk | type: CONSTRAINT --
243 -- ALTER TABLE public.many_hub_person_has_many_hub_event DROP CONSTRAINT
       (cont.)IF EXISTS hub_event_fk CASCADE;
244 ALTER TABLE public.many_hub_person_has_many_hub_event ADD CONSTRAINT
       (cont.)hub_event_fk FOREIGN KEY (id_hub_event)
245 REFERENCES public.hub_event (id) MATCH FULL
246 ON DELETE RESTRICT ON UPDATE CASCADE;
247 -- ddl-end --
248
249 -- object: hub_person_fk | type: CONSTRAINT --
250 -- ALTER TABLE public.sat_person_consultant_chemo DROP CONSTRAINT IF
       (cont.)EXISTS hub_person_fk CASCADE;
251 ALTER TABLE public.sat_person_consultant_chemo ADD CONSTRAINT
       (cont.)hub_person_fk FOREIGN KEY (chi_hub_person)
252 REFERENCES public.hub_person (chi) MATCH FULL
253 ON DELETE RESTRICT ON UPDATE CASCADE;
254 -- ddl-end --
255
256 -- object: hub_person_fk | type: CONSTRAINT --
257 -- ALTER TABLE public.sat_person_patient_measurements DROP CONSTRAINT IF
       (cont.)EXISTS hub_person_fk CASCADE;
258 ALTER TABLE public.sat_person_patient_measurements ADD CONSTRAINT
       (cont.)hub_person_fk FOREIGN KEY (chi_hub_person)
259 REFERENCES public.hub_person (chi) MATCH FULL
260 ON DELETE RESTRICT ON UPDATE CASCADE;
261 -- ddl-end --
262
263 -- object: hub_time_fk | type: CONSTRAINT --
264 -- ALTER TABLE public."sat_time_last-toxicity_date" DROP CONSTRAINT IF
       (cont.)EXISTS hub_time_fk CASCADE;
265 ALTER TABLE public."sat_time_last-toxicity_date" ADD CONSTRAINT
```

```
   (cont.)hub_time_fk FOREIGN KEY (id_hub_time)
REFERENCES public.hub_time (id) MATCH FULL
ON DELETE RESTRICT ON UPDATE CASCADE;
-- ddl-end --

-- object: hub_time_fk | type: CONSTRAINT --
-- ALTER TABLE public.sat_time_appointment_date DROP CONSTRAINT IF EXISTS
   (cont.)hub_time_fk CASCADE;
ALTER TABLE public.sat_time_appointment_date ADD CONSTRAINT hub_time_fk
   (cont.)FOREIGN KEY (id_hub_time)
REFERENCES public.hub_time (id) MATCH FULL
ON DELETE RESTRICT ON UPDATE CASCADE;
-- ddl-end --

-- object: hub_object_fk | type: CONSTRAINT --
-- ALTER TABLE public.sat_object_tumour_group DROP CONSTRAINT IF EXISTS
   (cont.)hub_object_fk CASCADE;
ALTER TABLE public.sat_object_tumour_group ADD CONSTRAINT hub_object_fk
   (cont.)FOREIGN KEY (id_hub_object)
REFERENCES public.hub_object (id) MATCH FULL
ON DELETE RESTRICT ON UPDATE CASCADE;
-- ddl-end --

-- object: hub_object_fk | type: CONSTRAINT --
-- ALTER TABLE public.sat_object_drug_details DROP CONSTRAINT IF EXISTS
   (cont.)hub_object_fk CASCADE;
ALTER TABLE public.sat_object_drug_details ADD CONSTRAINT hub_object_fk
   (cont.)FOREIGN KEY (id_hub_object)
REFERENCES public.hub_object (id) MATCH FULL
ON DELETE RESTRICT ON UPDATE CASCADE;
-- ddl-end --

-- object: hub_object_fk | type: CONSTRAINT --
-- ALTER TABLE public.sat_object_treatment_details DROP CONSTRAINT IF
   (cont.)EXISTS hub_object_fk CASCADE;
ALTER TABLE public.sat_object_treatment_details ADD CONSTRAINT
   (cont.)hub_object_fk FOREIGN KEY (id_hub_object)
REFERENCES public.hub_object (id) MATCH FULL
ON DELETE RESTRICT ON UPDATE CASCADE;
-- ddl-end --

-- object: hub_event_fk | type: CONSTRAINT --
-- ALTER TABLE public.sat_event_drug_status DROP CONSTRAINT IF EXISTS
   (cont.)hub_event_fk CASCADE;
ALTER TABLE public.sat_event_drug_status ADD CONSTRAINT hub_event_fk
   (cont.)FOREIGN KEY (id_hub_event)
REFERENCES public.hub_event (id) MATCH FULL
ON DELETE RESTRICT ON UPDATE CASCADE;
-- ddl-end --
```

A.2 Full Cancer Informatics SQL for MySQL

The following SQL scripts for MySQL will deploy the empty Cancer.Informatics schema that is required to store the source system data for the data crawlers to process into the Smart Health Data Vault (SHDV)

```
CREATE TABLE `hub_event` (
  `id` integer,
```

```
  PRIMARY KEY (`id`)
);

CREATE TABLE `hub_object` (
  `id` integer,
  PRIMARY KEY (`id`)
);

CREATE TABLE `hub_person` (
  `chi` integer,
  PRIMARY KEY (`chi`)
);

CREATE TABLE `hub_time` (
  `id` integer,
  PRIMARY KEY (`id`)
);

CREATE TABLE `many_hub_person_has_many_hub_event` (
  `chi_hub_person` integer,
  `id_hub_event` integer,
  `id` integer,
  PRIMARY KEY (`id`),
  KEY `FK` (`chi_hub_person`, `id_hub_event`)
);

CREATE TABLE `many_hub_person_has_many_hub_object` (
  `chi_hub_person` integer,
  `id_hub_object` integer,
  `id` integer,
  PRIMARY KEY (`id`),
  KEY `FK` (`chi_hub_person`, `id_hub_object`)
);

CREATE TABLE `many_hub_person_has_many_hub_time` (
  `chi_hub_person` integer,
  `id_hub_time` integer,
  `id` integer,
  PRIMARY KEY (`id`),
  KEY `FK` (`chi_hub_person`, `id_hub_time`)
);

CREATE TABLE `sat_event_drug_status` (
  `id` integer,
  `drug_status` character varying(255),
  `id_hub_event` integer,
  PRIMARY KEY (`id`),
  KEY `FK` (`id_hub_event`)
);

CREATE TABLE `sat_object_drug_details` (
  `id` integer,
  `drug_type` character(9),
  `drug_name` character varying(255),
  `given_drug` numeric,
  `drug_dose` numeric,
  `required_dose` numeric,
  `id_hub_object` integer,
  PRIMARY KEY (`id`),
  KEY `FK` (`id_hub_object`)
);
```

```
CREATE TABLE `sat_object_treatment_details` (
  `id` integer,
  `intention` character varying(255),
  `regime` character varying(255),
  `cycle` integer,
  `id_hub_object` integer,
  PRIMARY KEY (`id`),
  KEY `FK` (`id_hub_object`)
);

CREATE TABLE `sat_object_tumour_group` (
  `id` integer,
  `tumour_group` character varying(255),
  `id_hub_object` integer,
  PRIMARY KEY (`id`),
  KEY `FK` (`id_hub_object`)
);

CREATE TABLE `sat_person_consultant_chemo` (
  `id` integer,
  `consultant_code` character(2),
  `chi_hub_person` integer,
  PRIMARY KEY (`id`),
  KEY `FK` (`chi_hub_person`)
);

CREATE TABLE `sat_person_patient_measurements` (
  `id` integer,
  `age_at_diagnosis` double precision,
  `height` numeric,
  `weight` numeric,
  `surface_area` numeric,
  `chi_hub_person` integer,
  PRIMARY KEY (`id`),
  KEY `FK` (`chi_hub_person`)
);

CREATE TABLE `sat_time_appointment_date` (
  `id` integer,
  `appointment_date` date,
  `id_hub_time` integer,
  PRIMARY KEY (`id`),
  KEY `FK` (`id_hub_time`)
);

CREATE TABLE `sat_time_last-toxicity_date` (
  `id` integer,
  `last_toxicity_date` date,
  `id_hub_time` integer,
  PRIMARY KEY (`id`),
  KEY `FK` (`id_hub_time`)
);
```

A.3 Full Cancer Informatics XML

The following XML for an empty Cancer.Informatics schema that is required to store the source system data for the data crawlers to process into the Smart Health Data Vault (SHDV)

```
1
2  <?xml version="1.0" encoding="UTF-8"?>
3  <!--
4  CAUTION: Do not modify this file unless you know what you are doing.
5           Unexpected results may occur if the code is changed deliberately.
6  -->
7  <schema name="public" layer="0" fill-color="#e1e1e1" sql-disabled="true">
8  </schema>
9
10 <table name="hub_person" layer="0" collapse-mode="2" max-obj-count="1">
11    <schema name="public"/>
12    <role name="postgres"/>
13    <position x="440" y="180"/>
14    <column name="chi" alias="PATIENT IDENTIFICATION NUMBER" not-null="true
         (cont.)"
15     identity-type="ALWAYS">
16       <type name="int4" length="0"/>
17    </column>
18    <constraint name="hub_person_pk" type="pk-constr" table="public.
         (cont.)hub_person">
19       <columns names="chi" ref-type="src-columns"/>
20    </constraint>
21 </table>
22
23 <table name="hub_time" layer="0" collapse-mode="2" max-obj-count="1">
24    <schema name="public"/>
25    <role name="postgres"/>
26    <position x="100" y="180"/>
27    <column name="id" not-null="true">
28       <type name="serial" length="0"/>
29    </column>
30    <constraint name="hub_time_pk" type="pk-constr" table="public.hub_time"
         (cont.)>
31       <columns names="id" ref-type="src-columns"/>
32    </constraint>
33 </table>
34
35 <table name="sat_person_patient_measurements" layer="0" collapse-mode="2"
      (cont.)max-obj-count="7">
36    <schema name="public"/>
37    <role name="postgres"/>
38    <position x="440" y="360"/>
39    <column name="id" not-null="true">
40       <type name="serial" length="0"/>
41    </column>
42    <column name="age_at_diagnosis">
43       <type name="float" length="0"/>
44    </column>
45    <column name="height">
46       <type name="decimal" length="3" precision="2"/>
47    </column>
48    <column name="weight">
49       <type name="decimal" length="4" precision="1"/>
50    </column>
51    <column name="surface_area">
52       <type name="decimal" length="3" precision="2"/>
53    </column>
54    <constraint name="sat_person_patient_measurements_pk" type="pk-constr"
         (cont.)table="public.sat_person_patient_measurements">
55       <columns names="id" ref-type="src-columns"/>
56    </constraint>
```

```
</table>

<table name="hub_object" layer="0" collapse-mode="2" max-obj-count="1">
   <schema name="public"/>
   <role name="postgres"/>
   <position x="820" y="180"/>
   <column name="id" not-null="true">
      <type name="serial" length="0"/>
   </column>
   <constraint name="hub_object_pk" type="pk-constr" table="public.
      (cont.)hub_object">
      <columns names="id" ref-type="src-columns"/>
   </constraint>
</table>

<table name="sat_time_appointment_date" layer="0" collapse-mode="2" max-
   (cont.)obj-count="3">
   <schema name="public"/>
   <role name="postgres"/>
   <position x="100" y="380"/>
   <column name="id" not-null="true">
      <type name="serial" length="0"/>
   </column>
   <column name="appointment_date">
      <type name="date" length="0"/>
   </column>
   <constraint name="sat_time_appointment_date_pk" type="pk-constr" table=
      (cont.)"public.sat_time_appointment_date">
      <columns names="id" ref-type="src-columns"/>
   </constraint>
</table>

<table name="sat_time_last-toxicity_date" layer="0" collapse-mode="2" max-
   (cont.)obj-count="3">
   <schema name="public"/>
   <role name="postgres"/>
   <position x="100" y="20"/>
   <column name="id" not-null="true">
      <type name="serial" length="0"/>
   </column>
   <column name="last_toxicity_date">
      <type name="date" length="0"/>
   </column>
   <constraint name="sat_time_last-toxicity_date_pk" type="pk-constr"
      (cont.)table="public.&amp;quot;sat_time_last-toxicity_date&amp;quot;">
      <columns names="id" ref-type="src-columns"/>
   </constraint>
</table>

<table name="sat_object_tumour_group" layer="0" collapse-mode="2" max-obj-
   (cont.)count="3">
   <schema name="public"/>
   <role name="postgres"/>
   <position x="820" y="20"/>
   <column name="id" not-null="true">
      <type name="serial" length="0"/>
   </column>
   <column name="tumour_group">
      <type name="varchar" length="255"/>
   </column>
   <constraint name="sat_object_tumour_group_pk" type="pk-constr" table="
      (cont.)public.sat_object_tumour_group">
```

```
112         <columns names="id" ref-type="src-columns"/>
113      </constraint>
114   </table>
115
116   <table name="sat_person_consultant_chemo" layer="0" collapse-mode="2" max-
         (cont.)obj-count="3">
117      <schema name="public"/>
118      <role name="postgres"/>
119      <position x="440" y="40"/>
120      <column name="id" not-null="true">
121         <type name="serial" length="0"/>
122      </column>
123      <column name="consultant_code">
124         <type name="char" length="2"/>
125      </column>
126      <constraint name="sat_person_consultant_chemo_pk" type="pk-constr"
            (cont.)table="public.sat_person_consultant_chemo">
127         <columns names="id" ref-type="src-columns"/>
128      </constraint>
129   </table>
130
131   <table name="hub_event" layer="0" collapse-mode="2" max-obj-count="1">
132      <schema name="public"/>
133      <role name="postgres"/>
134      <position x="820" y="300"/>
135      <column name="id" not-null="true">
136         <type name="serial" length="0"/>
137      </column>
138      <constraint name="hub_event_pk" type="pk-constr" table="public.
            (cont.)hub_event">
139         <columns names="id" ref-type="src-columns"/>
140      </constraint>
141   </table>
142
143   <table name="sat_object_drug_details" layer="0" collapse-mode="2" max-obj-
         (cont.)count="8">
144      <schema name="public"/>
145      <role name="postgres"/>
146      <position x="1160" y="60"/>
147      <column name="id" not-null="true">
148         <type name="serial" length="0"/>
149      </column>
150      <column name="drug_type">
151         <type name="char" length="9"/>
152      </column>
153      <column name="drug_name">
154         <type name="varchar" length="255"/>
155      </column>
156      <column name="given_drug">
157         <type name="decimal" length="72"/>
158      </column>
159      <column name="drug_dose">
160         <type name="decimal" length="7" precision="2"/>
161      </column>
162      <column name="required_dose">
163         <type name="decimal" length="7" precision="2"/>
164      </column>
165      <constraint name="sat_object_drug_details_pk" type="pk-constr" table="
            (cont.)public.sat_object_drug_details">
166         <columns names="id" ref-type="src-columns"/>
167      </constraint>
168   </table>
```

```
169
170 <table name="sat_object_treatment_details" layer="0" collapse-mode="2" max
       (cont.)-obj-count="6">
171     <schema name="public"/>
172     <role name="postgres"/>
173     <position x="1160" y="280"/>
174     <column name="id" not-null="true">
175        <type name="serial" length="0"/>
176     </column>
177     <column name="intention">
178        <type name="varchar" length="255"/>
179     </column>
180     <column name="regime">
181        <type name="varchar" length="255"/>
182     </column>
183     <column name="cycle">
184        <type name="int4" length="0"/>
185     </column>
186     <constraint name="sat_object_treatment_details_pk" type="pk-constr"
          (cont.)table="public.sat_object_treatment_details">
187        <columns names="id" ref-type="src-columns"/>
188     </constraint>
189 </table>
190
191 <table name="sat_event_drug_status" layer="0" collapse-mode="2" max-obj-
       (cont.)count="3">
192     <schema name="public"/>
193     <role name="postgres"/>
194     <position x="820" y="480"/>
195     <column name="id" not-null="true">
196        <type name="serial" length="0"/>
197     </column>
198     <column name="drug_status">
199        <type name="varchar" length="255"/>
200     </column>
201     <constraint name="sat_event_drug_status_pk" type="pk-constr" table="
          (cont.)public.sat_event_drug_status">
202        <columns names="id" ref-type="src-columns"/>
203     </constraint>
204 </table>
205
206 <relationship name="many_hub_person_has_many_hub_time" type="relnn" layer=
       (cont.)"0"
207     src-col-pattern="{sc}_{st}" dst-col-pattern="{sc}_{dt}"
208     pk-pattern="{gt}_pk" uq-pattern="{gt}_uq"
209     src-fk-pattern="{st}_fk" dst-fk-pattern="{dt}_fk"
210     pk-col-pattern="id"
211     custom-color="#a0138f"
212     src-table="public.hub_person"
213     dst-table="public.hub_time"
214     src-required="false" dst-required="false"
215     single-pk-col="true"
216     table-name="many_hub_person_has_many_hub_time"/>
217
218 <relationship name="many_hub_person_has_many_hub_object" type="relnn"
       (cont.)layer="0"
219     src-col-pattern="{sc}_{st}" dst-col-pattern="{sc}_{dt}"
220     pk-pattern="{gt}_pk" uq-pattern="{gt}_uq"
221     src-fk-pattern="{st}_fk" dst-fk-pattern="{dt}_fk"
222     pk-col-pattern="id"
223     custom-color="#32b689"
224     src-table="public.hub_person"
```

```
225      dst-table="public.hub_object"
226      src-required="false" dst-required="false"
227      single-pk-col="true"
228      table-name="many_hub_person_has_many_hub_object"/>
229
230  <relationship name="many_hub_person_has_many_hub_event" type="relnn" layer
         (cont.)="0"
231      src-col-pattern="{sc}_{st}" dst-col-pattern="{sc}_{dt}"
232      pk-pattern="{gt}_pk" uq-pattern="{gt}_uq"
233      src-fk-pattern="{st}_fk" dst-fk-pattern="{dt}_fk"
234      pk-col-pattern="id"
235      custom-color="#0bd416"
236      src-table="public.hub_person"
237      dst-table="public.hub_event"
238      src-required="false" dst-required="false"
239      single-pk-col="true"
240      table-name="many_hub_person_has_many_hub_event"/>
241
242  <relationship name="hub_person_has_many_sat_person_consultant_chemo" type=
         (cont.)"rel1n" layer="0"
243      src-col-pattern="{sc}_{st}"
244      pk-pattern="{dt}_pk" uq-pattern="{dt}_uq"
245      src-fk-pattern="{st}_fk"
246      custom-color="#bc1c77"
247      src-table="public.hub_person"
248      dst-table="public.sat_person_consultant_chemo"
249      src-required="true" dst-required="false"/>
250
251  <relationship name="hub_person_has_many_sat_person_patient_measurements"
         (cont.)type="rel1n" layer="0"
252      src-col-pattern="{sc}_{st}"
253      pk-pattern="{dt}_pk" uq-pattern="{dt}_uq"
254      src-fk-pattern="{st}_fk"
255      custom-color="#f3b73f"
256      src-table="public.hub_person"
257      dst-table="public.sat_person_patient_measurements"
258      src-required="true" dst-required="false"/>
259
260  <relationship name="hub_time_has_many_sat_time_last-toxicity_date" type="
         (cont.)rel1n" layer="0"
261      src-col-pattern="{sc}_{st}"
262      pk-pattern="{dt}_pk" uq-pattern="{dt}_uq"
263      src-fk-pattern="{st}_fk"
264      custom-color="#fbbf8c"
265      src-table="public.hub_time"
266      dst-table="public.&amp;quot;sat_time_last-toxicity_date&amp;quot;"
267      src-required="true" dst-required="false"/>
268
269  <relationship name="hub_time_has_many_sat_time_appointment_date" type="
         (cont.)rel1n" layer="0"
270      src-col-pattern="{sc}_{st}"
271      pk-pattern="{dt}_pk" uq-pattern="{dt}_uq"
272      src-fk-pattern="{st}_fk"
273      custom-color="#1c83ef"
274      src-table="public.hub_time"
275      dst-table="public.sat_time_appointment_date"
276      src-required="true" dst-required="false"/>
277
278  <relationship name="hub_object_has_many_sat_object_tumour_group" type="
         (cont.)rel1n" layer="0"
279      src-col-pattern="{sc}_{st}"
280      pk-pattern="{dt}_pk" uq-pattern="{dt}_uq"
```

```
281      src-fk-pattern="{st}_fk"
282      custom-color="#ae85df"
283      src-table="public.hub_object"
284      dst-table="public.sat_object_tumour_group"
285      src-required="true" dst-required="false"/>
286
287  <relationship name="hub_object_has_many_sat_object_drug_details" type="
         (cont.)rel1n" layer="0"
288      src-col-pattern="{sc}_{st}"
289      pk-pattern="{dt}_pk" uq-pattern="{dt}_uq"
290      src-fk-pattern="{st}_fk"
291      custom-color="#213edb"
292      src-table="public.hub_object"
293      dst-table="public.sat_object_drug_details"
294      src-required="true" dst-required="false">
295     <label ref-type="name-label">
296        <position x="-26.1264" y="-56.8607"/>
297     </label>
298  </relationship>
299
300  <relationship name="hub_object_has_many_sat_object_treatment_details" type
         (cont.)="rel1n" layer="0"
301      src-col-pattern="{sc}_{st}"
302      pk-pattern="{dt}_pk" uq-pattern="{dt}_uq"
303      src-fk-pattern="{st}_fk"
304      custom-color="#e7475c"
305      src-table="public.hub_object"
306      dst-table="public.sat_object_treatment_details"
307      src-required="true" dst-required="false">
308     <label ref-type="name-label">
309        <position x="-16.1264" y="-34.2607"/>
310     </label>
311  </relationship>
312
313  <relationship name="hub_event_has_many_sat_event_drug_status" type="rel1n"
         (cont.) layer="0"
314      src-col-pattern="{sc}_{st}"
315      pk-pattern="{dt}_pk" uq-pattern="{dt}_uq"
316      src-fk-pattern="{st}_fk"
317      custom-color="#2b4c13"
318      src-table="public.hub_event"
319      dst-table="public.sat_event_drug_status"
320      src-required="true" dst-required="false">
321     <label ref-type="name-label">
322        <position x="28.3947" y="-1.66066"/>
323     </label>
324  </relationship>
```

A.4 Panda Profiling

This output was generated by Python with Pandas-Profiling (https://github.com/pandas-profiling/pandas-profiling)

The process generates for each column in the data sorce the following statistics - if relevant for the column type - are presented in an interactive HTML report:

- Essentials: type, unique values, missing values

- Quantile statistics like minimum value, Q1, median, Q3, maximum, range, interquartile range
- Descriptive statistics like mean, mode, standard deviation, sum, median absolute deviation, coefficient of variation, kurtosis, skewness
- Most frequent values
- Histogram of data values
- Correlations highlighting of highly correlated variables, Spearman and Pearson matrixes

Overview

Dataset info

Number of variables	19
Number of observations	59757
Total Missing (%)	1.0%
Total size in memory	8.7 MiB
Average record size in memory	152.0 B

Variables types

Numeric	6
Categorical	10
Boolean	0
Date	0
Text (Unique)	0
Rejected	3
Unsupported	0

Warnings

`Appointment Date` has a high cardinality: 1086 distinct values **Warning**
`CHI` has a high cardinality: 4914 distinct values **Warning**
`Drug Name` has a high cardinality: 62 distinct values **Warning**
`Drug Type` has constant value Cytotoxic **Rejected**
`Given Dose` has 11078 / 18.5% zeros **Zeros**
`Height` has 2227 / 3.7% zeros **Zeros**
`Last Toxicity Date` has 10857 / 18.2% missing values **Missing**
`Last Toxicity Date` has a high cardinality: 871 distinct values **Warning**
`Regime` has a high cardinality: 223 distinct values **Warning**
`Required Dose` is highly correlated with `Drug Dose` (ρ = 0.96787) **Rejected**
`Surface Area` is highly correlated with `Weight` (ρ = 0.91716) **Rejected**
`Weight` has 2227 / 3.7% zeros **Zeros**
Dataset has 23 duplicate rows **Warning**

Variables

Appointment Date

Categorical

Distinct count	1086

Unique (%)	1.8%
Missing (%)	0.0%
Missing (n)	0

05-01-2016	113	
02-03-2016	111	
30-12-2015	110	
Other values (1083)		59423

Toggle details

CHI
Categorical

Distinct count	4914
Unique (%)	8.2%
Missing (%)	0.0%
Missing (n)	0

0307541363	148	
2606701259	138	
1010952269	123	
Other values (4911)		59348

Toggle details

Consultant Code
Categorical

Distinct count	43
Unique (%)	0.1%
Missing (%)	0.0%
Missing (n)	0

LH	5420	
AB	4173	
MM	3858	
Other values (40)		46306

Toggle details

Cycle
Numeric

Distinct count	85
Unique (%)	0.1%
Missing (%)	0.0%
Missing (n)	0

Infinite (%)	0.0%
Infinite (n)	0
Mean	4.0003
Minimum	1
Maximum	85
Zeros (%)	0.0%

Toggle details

Drug Dose

Numeric

Distinct count	102
Unique (%)	0.2%
Missing (%)	0.0%
Missing (n)	0
Infinite (%)	0.0%
Infinite (n)	0
Mean	566.32
Minimum	0
Maximum	30000
Zeros (%)	0.5%

Toggle details

Drug Name

Categorical

Distinct count	62
Unique (%)	0.1%
Missing (%)	0.0%
Missing (n)	0

FLUOROURACIL	6994	
PACLITAXEL	6097	
CAPECITABINE	5823	
Other values (59)		40843

Toggle details

Drug Status
Categorical

Distinct count	5
Unique (%)	0.0%
Missing (%)	0.2%
Missing (n)	99

Given		48682
Authorised	6017	
Planned	4597	

Toggle details

~~Drug Type~~
Constant

This variable is constant and should be ignored for analysis

Constant value	Cytotoxic

Given Dose
Numeric

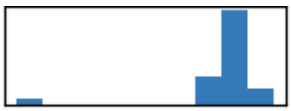
Distinct count	502
Unique (%)	0.8%
Missing (%)	0.0%
Missing (n)	0
Infinite (%)	0.0%
Infinite (n)	0
Mean	728.33
Minimum	0
Maximum	30000
Zeros (%)	18.5%

Toggle details

Height
Numeric

Distinct count	63
Unique (%)	0.1%
Missing (%)	0.0%
Missing (n)	0
Infinite (%)	0.0%

Infinite (n)	0
Mean	1.6037
Minimum	0
Maximum	1.98
Zeros (%)	3.7%



Toggle details

Intention
Categorical

Distinct count	8
Unique (%)	0.0%
Missing (%)	0.0%
Missing (n)	0

Palliative		29628
Adjuvant	12651	
Curative	7220	
Other values (5)	10258	

Toggle details

Last Toxicity Date
Categorical

Distinct count	871
Unique (%)	1.5%
Missing (%)	18.2%
Missing (n)	10857

29-12-2015	115	
03-05-2016	109	
08-07-2014	108	
Other values (867)		48568
(Missing)	10857	

Toggle details

Patient Type
Categorical

Distinct count	3
Unique (%)	0.0%
Missing (%)	0.0%

Missing (n)	0

D	48248
I	8045
O	3464

Toggle details

Regime

Categorical

Distinct count	223
Unique (%)	0.4%
Missing (%)	0.0%
Missing (n)	0

CAPOX	3501
FEC 80	2931
PACLITAX WKLY	2880
Other values (220)	50445

Toggle details

~~Required Dose~~

Highly correlated

This variable is highly correlated with `Drug Dose` *and should be ignored for analysis*

Correlation	0.96787

~~Surface Area~~

Highly correlated

This variable is highly correlated with `Weight` *and should be ignored for analysis*

Correlation	0.91716

Tumour Group

Categorical

Distinct count	15
Unique (%)	0.0%
Missing (%)	0.0%
Missing (n)	0

Breast	14417
GI Lower	11131
Lung and Chest	7744
Other values (12)	26465

Toggle details

Weight
Numeric

Distinct count	858
Unique (%)	1.4%
Missing (%)	0.0%
Missing (n)	0
Infinite (%)	0.0%
Infinite (n)	0
Mean	72.172
Minimum	0
Maximum	167.5
Zeros (%)	3.7%

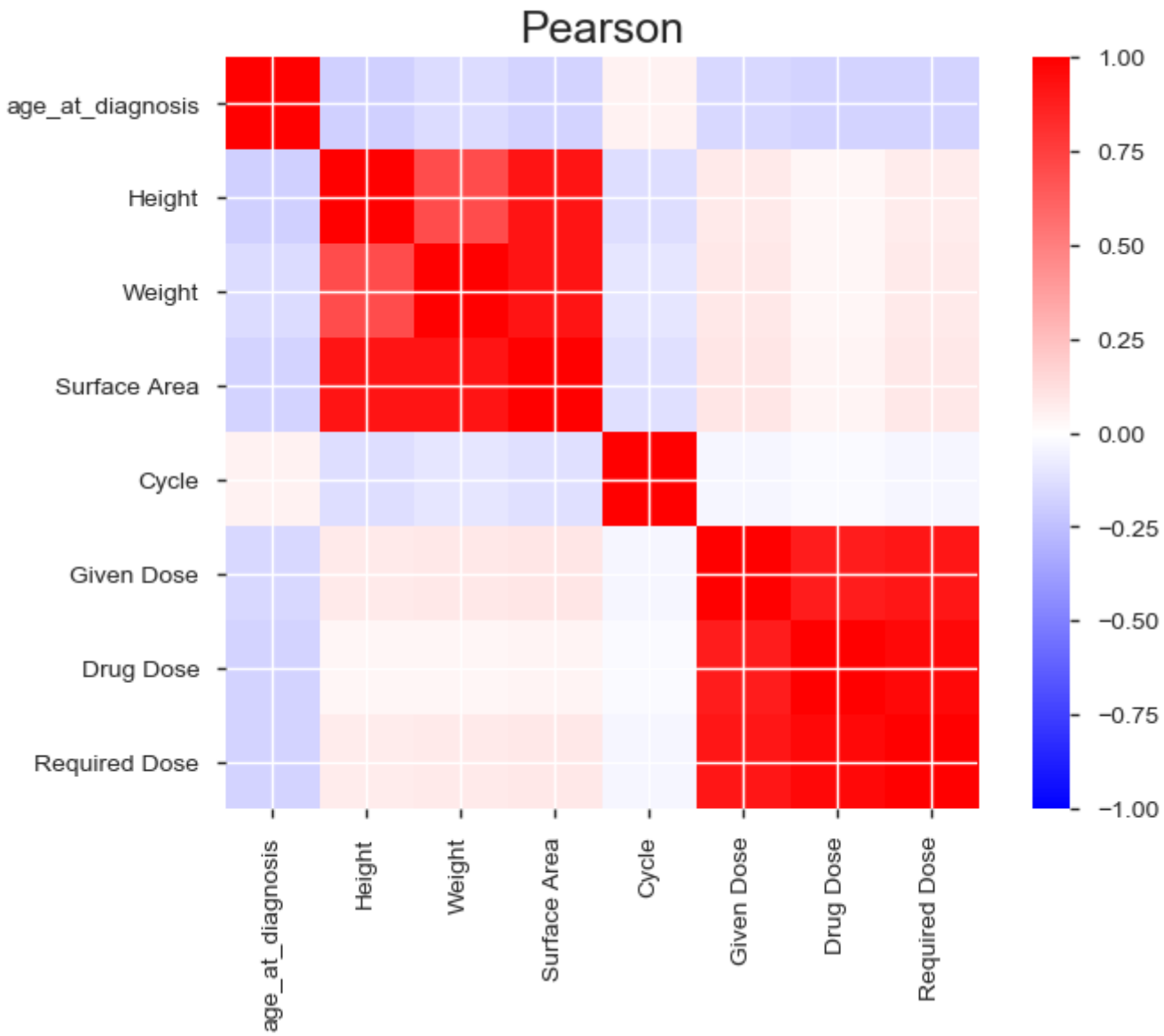
Toggle details

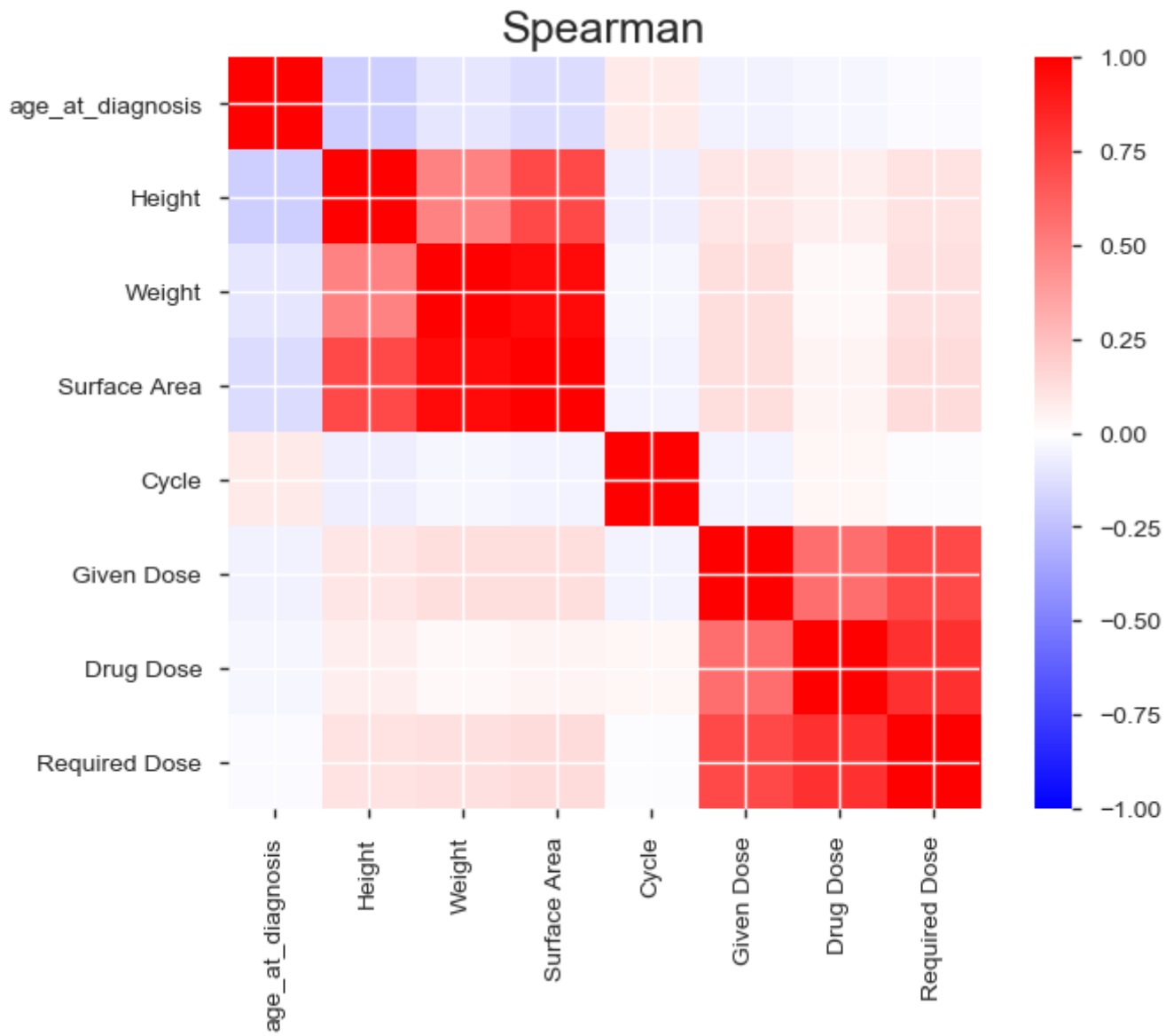
age_at_diagnosis
Numeric

Distinct count	16140
Unique (%)	27.0%
Missing (%)	0.1%
Missing (n)	34
Infinite (%)	0.0%
Infinite (n)	0
Mean	59.307
Minimum	16.321
Maximum	94.827
Zeros (%)	0.0%

Toggle details

Correlations


Pearson
age_at_diagnosis
Height
Weight
Surface Area
Cycle
Given Dose
Drug Dose
Required Dose
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00



Sample

	CHI	Appointment Date	Last Toxicity Date	Tumour Group	age_at_diagnosis	He
0	2608891225	02-01-2014	31-12-2013	Bone Sarcoma	24.369863	1.(
1	2608891225	02-01-2014	31-12-2013	Bone Sarcoma	24.369863	1.(
2	2608891225	02-01-2014	31-12-2013	Bone Sarcoma	24.369863	1.(
3	2608891225	02-01-2014	31-12-2013	Bone Sarcoma	24.369863	1.(
4	2608891225	03-01-2014	31-12-2013	Bone Sarcoma	24.372603	1.(

Bibliography

[1] Mike Folk, Gerd Heber, Quincey Koziol, Elena Pourmal, and Dana Robinson. An overview of the HDF5 technology suite and its applications. 2011.

[2] Information Commissioner's Office. Guide to the General Data Protection Regulation (GDPR). Technical report, 2018.