Enriching Integrated Statistical Open City Data by Combining Equational Knowledge and Missing Value Imputation

Stefan Bischof[a,*], Andreas Harth[b], Benedikt Kämpgen[c], Axel Polleres[d,e], Patrik Schneider[a]

[a]*Siemens AG Österreich, Siemensstrasse 90, 1210 Vienna, Austria*
[b]*Karlsruhe Institute of Technology, Karlsruhe, Germany*
[c]*FZI Research Center for Information Technology, Karlsruhe, Germany*
[d]*Vienna University of Economics and Business, Vienna, Austria*
[e]*Complexity Science Hub Vienna, Austria*

Abstract

Several institutions collect statistical data about cities, regions, and countries for various purposes. Yet, while access to high quality and recent such data is both crucial for decision makers and a means for achieving transparency to the public, all too often such collections of data remain isolated and not re-usable, let alone comparable or properly integrated. In this paper we present the Open City Data Pipeline, a focused attempt to collect, integrate, and enrich statistical data collected at city level worldwide, and re-publish the resulting dataset in a re-usable manner as Linked Data. The main features of the Open City Data Pipeline are: (i) we integrate and cleanse data from several sources in a modular and extensible, always up-to-date fashion; (ii) we use both Machine Learning techniques and reasoning over equational background knowledge to enrich the data by imputing missing values, (iii) we assess the estimated accuracy of such imputations per indicator. Additionally, (iv) we make the integrated and enriched data, including links to external data sources, such as DBpedia, available both in a web browser interface and as machine-readable Linked Data, using standard vocabularies such as QB and PROV.

Apart from providing a contribution to the growing collection of data available as Linked Data, our enrichment process for missing values also contributes a novel methodology for combining *rule-based inference* about equational knowledge with inferences obtained from *statistical Machine Learning* approaches. While most existing works about inference in Linked Data have focused on ontological reasoning in RDFS and OWL, we believe that these complementary methods and particularly their combination could be fruitfully applied also in many other domains for integrating Statistical Linked Data, independent from our concrete use case of integrating city data.

Keywords: open data, Linked Data, data cleaning, data integration

1. Introduction

The public sector collects large amounts of statistical data. For example, the United Nations Statistics Division[1] provides regularly updated statistics about the economy, demographics and social indicators, environment and energy, and gender on a global level. The statistical office of the European Commission, Eurostat[2], provides statistical data mainly about EU member countries. Some of the data in Eurostat has been aggregated from the statistical offices of the member countries of the EU. Even several larger cities provide data in on their own open data portals, e.g., Amsterdam, Berlin, London, or Vienna[3]. Increasingly, such data can be downloaded free of charge and used under liberal licences.

Such open data can benefit public administrations, citizens and enterprises. The public administration can use the data to support decision-making and back policy decisions in a transparent manner. Citizens can be better informed about government decisions, as publicly available data can help to raise awareness and underpin public discussions. Finally, companies could develop new business models and offer tailored solutions to their customers based on such open data. As an example for making use of such data, consider Siemens' Green City Index (GCI) [1], which assesses and compares the environmental performance of cities. In order to compute the KPIs used to rank cities' sustainability, the GCI used qualitative and also quantitative indicators about city performance, such as for instance CO_2 emissions or energy consumption per capita. Although many of these quantitative indicators had been openly available, the respective datasets had to be collected, integrated, and checked for integrity violations mostly manually because of the following reasons: (i) heterogeneity: ambiguous data published by different Open Data sources in different formats, (ii) missing data, that needed to be added manually through additional research in text documents

*Corresponding author
Email addresses: bischof.stefan@siemens.com (Stefan Bischof), harth@kit.edu (Andreas Harth), kaempgen@fzi.de (Benedikt Kämpgen), axel.polleres@wu.ac.at (Axel Polleres), patrik@kr.tuwien.ac.at (Patrik Schneider)

[1] http://unstats.un.org/unsd/

[2] http://ec.europa.eu/eurostat/

[3] http://data.amsterdam.nl/, http://daten.berlin.de/, http://data.london.gov.uk/, and http://data.wien.gv.at/

or estimated by experts, and, last but not least, (iii) outdated data: soon after the GCI had been published in 2012, its results were likely already obsolete.

Inspired by this concrete use case of the GCI, the goal of the present work is on collecting, integrating, and enriching quantitative indicator data about cities including basic statistical data about demographics, socio-economic factors, or environmental data, in a more automated and integrated fashion to alleviate these problems.

Even though there are many relevant data sources which publish such quantitative indicators as open data, it is still cumbersome to use data from multiple sources in combination and to keep this data up-to-date. The system we present in this paper, the Open City Data Pipeline, thus contributes by addressing all of the three above challenges (i)–(iii) in a holistic manner:

(i) Heterogeneity: All this data is published in different formats such as CSV, JSON, XML, proprietary formats such as XLS, just as plain HTML tables, or even worse within PDF files – and so far to a much lesser degree only as RDF or even as Linked Data [2]. Also, the specifications of the individual data fields – (a) how indicators are defined and (b) how they have been collected – are often implicit in textual descriptions only and have to be processed manually for understanding whether seemingly identical indicators published by different sources are indeed comparable.

Our contribution: we present a systematic approach to integrate statistical data about cities from different sources as *Statistical Linked Data* [3] as a standardised format to publish both the data and the metadata. We build a small ontology of core city indicators, around which we can grow a statistical Linked Data cube: we use standard Linked Data vocabularies such as the RDF Data Cube (QB) [4] vocabulary to represent data of statistical data cubes, as well as the PROV [5] vocabulary to track the original sources of the data, and we create an extensible pipeline of crawlers and Linked Data wrappers collect this data from the sources.

(ii) Missing values: Data sources like Eurostat Urban Audit cover many cities and indicators. However, for reasons such as cities providing values on a voluntary basis, the published datasets show a large ratio of missing values. The impact of missing values is aggravated when combining different data sets, due to either covering different cities or using different, non-overlapping sets of indicators.

Our contribution: our assumption – inspired also by works that suspect the existence of quantitative models behind the working, growth, and scaling of cities [6] – is that most indicators in such a scoped domain as cities have their own structure and dependencies, from which we can build statistical prediction models and ontological background knowledge in the form of equations.[4] We have developed and combined integrated methods to compute missing values on the one hand using statistical inference, such as different standard regression methods, and on the other hand rule-based inference based on background knowledge in the form of equations that express knowledge about how certain numerical indicators can be computed from others.[5] While this new method is inspired by our own prior work on using statistical regression methods [7] and equational knowledge in isolation [8]), as we can demonstrate in our evaluation, the combination of both methods outperforms either method used alone. We re-publish the imputed/estimated values, adding respective PROV records, and including error estimates, as Linked Data.

(iii) Updates and changes: Studies like the GCI are typically outdated soon after publication since reusing or analysing the evolution of their underlying data is difficult. To improve this situation, we need regularly updated, integrated data stores which provide a consolidated, up-to-date view on data from relevant sources.

Our contribution: the extensible single data source wrappers (based on the work around rule-based linked data wrappers by Stadtmüller et al. [9]) in our pipeline architecture, are crawling each integrated source regularly (once a day) for new data, thus keeping the information as up-to-date as possible, while at the same time re-triggering the missing value enrichment methods and thereby continuously improving the quality of our estimations for missing data: indeed we can show in our evaluations that the more data we collect in our pipeline over time, the better our prediction models for missing values get.

In summary, our work's contribution is twofold, both in terms of building a practically deployed, concrete system to integrate and enrich statistical data about cities in a uniform, coherent and re-usable manner, and contributing novel methods to enrich and assess the quality of Statistical Linked Data:

1. as for the former, we present the Open City Data Pipeline which is based on *a generic, extensible architecture* and how we integrate data from multiple data sources that publish numerical data about cities in a modular and extensible way, which we re-publish as Statistical linked data.
2. as for the latter, we describe *the combination of statistical regression methods with equational background knowledge*, which we call *QB equations*, in order to impute and estimate missing values.

[4] We sometimes refer to "predicting" instead of "imputing" values when we mean finding suitable approximation models to estimate indicator values for cities and temporal contexts where they are not (yet) available. These predictions may (not) be confirmed, if additional data becomes available.

[5] Such equational knowledge could be also understood as "mapping" between indicators, which together with manually crafted equality mappings between indicators published by different data sources can be exploited for enrichment, e.g. if one source publishes the population and area of a city, but not the population density, then this missing value, available for other cities directly from other sources, could be computed by an equation.

We also *evaluate* our approach in terms of measuring the errors (by evaluating the estimated root mean square error rate (RMSE) per indicator) of such estimates, demonstrating that firstly, the combination of statistical inference with equations indeed pays off, and secondly, the regular update and collection of additional data through our pipeline contributes to improve our estimations for missing values in terms of accuracy. Note that the method of enrichment by QB equations can not only be used for imputing missing values, but also be used to assess the quality of ambiguous values from different data sources: by "rating" different observed values for the same indicator and city from different sources against their distance to our estimation, we have means to return confidence in different sources in such an integrated system.

The remainder of the paper is organised as follows. Section 2 introduces the necessary preliminaries in terms of Statistical Linked Data and other technical background, such as an overview of the used machine learning methods for missing value imputation. Section 3 gives an overview of the City Data Pipeline architecture, including a description of data sources and a description of how the resulting data set is made available in a re-usable and sustainable manner via a web interface, a Linked Data interface and a public SPARQL endpoint. Section 4 describes the data gathering as well as the main challenges in this context. Section 5 explains the missing data prediction process in more detail. Section 6 refines this process by introducing and applying QB equations. Both the basic value imputation mechanism and the refinement by QB equations are evaluated in Section 7. Section 8 puts our approach in the context of related work. Section 9 gives conclusions, provides lessons learnt and summaries directions for future research.

2. Preliminaries

In the following, we briefly introduce some core terms and notations used throughout the paper. We start with how Statistical Linked Data allows for modelling, wrapping, crawling, and querying of numerical data. We continue with provenance annotations to allow linking and integration as well as tracking the origin of numerical data. Then, we explain how equational background knowledge allows inferencing of numeric information. Also, we explain the basics of missing value prediction using machine learning methods.

Statistical Linked Data. Our focus is on data integration using web technologies. As such, we use technologies such as RDF [10], RDFS, OWL, and SPARQL 1.1 (both query language [11] and update language [12]) to represent, query, and integrate statistical data. We assume the reader is familiar with these standards. Statistical Linked Data refers to statistics published according to the Linked Data principles [13] reusing the RDF Data Cube Vocabulary (QB) [4] as a basis for representing both the individual data points and the metadata. QB is a widely-used vocabulary to describe statistical datasets as so-called data cubes using a multidimensional data model [14].

In the following, we illustrate how the metadata of a population dataset can be modelled using QB. If not stated otherwise, we use (abbreviated) Turtle notation for RDF[6]. The following is an excerpt from Turtle documents containing data from Eurostat:

```
</id/urb_cpop1#ds> a qb:DataSet ;
  rdfs:label "Population on 1 January by age groups and
       sex - cities and greater cities" ;
  qb:structure </dsd/urb_cpop1#dsd> .

</dsd/urb_cpop1#dsd> a qb:DataStructureDefinition;
  qb:component [ qb:dimension dcterms:date ] ;
  qb:component [ qb:dimension estatwrap:cities ] ;
  qb:component [ qb:dimension estatwrap:indic_ur ] ;
  qb:component [ qb:measure sdmx-measure:obsValue ] .
```

In the example, a data structure definition (DSD) defines the independent, categorical properties of the dataset, so-called dimensions: date, city, and indicator. Also, the DSD defines one dependent numeric property, so-called measure. The data structure definition could also include all valid dimension values, such as all city URIs for dimension `estatwrap:cities`.

Now, we give an example of how one data point can be modelled using QB:

```
_:obs1 a qb:Observation ;
       qb:dataSet </id/urb_cpop1#ds> ;
       estatwrap:cities </dic/cities#AT001C1> ;
       estatwrap:indic_ur </dic/indic_ur#DE1001V>
       dcterms:date "2013" ;
       sdmx-measure:obsValue "1741246" .
```

The example describes an observation of 1,741,246 inhabitants of Vienna in 2013 in the population dataset of Eurostat. Since any individual data point within a dataset is uniquely described by its dimension-value combinations, observations are usually modelled using blank nodes.

In the remainder of the paper we use the terms (statistical) dataset, QB dataset and (data) cube synonymously. The QB specification defines the notion of "well-formed cubes"[7] based on constraints that need to hold on a dataset. When generating and publishing QB datasets, we ensure that these constraints are fulfilled. For instance, when we later generate new observations via predictions and computations we also generate new datasets containing these values.

For publishing statistics in arbitrary formats as Statistical Linked Data, wrappers can access data from the original source, either in real-time or in batch mode, from the original format, e.g., CSV, and provide the data as Statistical Linked Data to the consumer. To collect data from different sources in one place, Linked Data crawlers can start with a seed list of dataset URIs and follow links to access the connected RDF documents. Observations from a dataset can be queried (e.g., filtered and aggregated) using SPARQL [3, 15].

Linking and Provenance Annotations. In our scenario, integration means building a unified view that allows querying observations from several datasets as if they would reside in a single dataset (we will refer later to this unified view as the global cube). To allow for querying the unified view the query processor requires mappings and the means to use these mappings during query evaluation.

[6] Use `http://prefix.cc/` to look up prefix declarations.
[7] `https://www.w3.org/TR/vocab-data-cube/#wf`

Consider a query to return all values of the indicator "population" of the area "Vienna", in the year "2010" simultaneously over two datasets. The two datasets may use different identifiers for the same dimensions, e.g., `estatwrap:geo` vs. `sdmx-dimension:refArea` and dimension values, e.g., `http://-estatwrap.ontologycentral.com/dic/cities#AT001C1` vs. `db-pedia:Vienna`. Hence, the query would need to take these different URIs into account, or the query processor requires additional data to be able to resolve the differences.

The following RDF snippet contains example links between a dimensions and a dimension value:

```
estatwrap:cities rdfs:subPropertyOf sdmx-dimension:refArea .
<http://estatwrap.ontologycentral.com/dic/cities#AT001C1> owl:
    sameAs dbpedia:Vienna .
```

When considering the semantics of such links when querying the observations of two or more datasets with identical dimensions, observations can be queried simultaneously as if they would reside in a single dataset [16].

To make observations more traceable and allow to judge the trustworthiness of data, we go beyond the lightweight approach of using Dublin Core properties such as `dc:publisher` to refer from a dataset to its publisher. We use the PROV ontology [5] to add provenance annotations, such as the agents and activities that were involved in generating observations from other observations (e.g., predicting, inferencing). The following RDF fragment shows a PROV example of two observations, where a QB observation `ex:obs123` was derived from another observation `ex:obs789` via an activity `ex:activity456` on the 15th of January 2017 at 12:37. This derivation was executed according to the rule `ex:rule937` with an agent `ex:fred` being responsible.

```
ex:obs123 prov:generatedAtTime "2017-01-15T12:37:00" ;
    prov:wasDerivedFrom ex:obs789 ;
    prov:wasGeneratedBy ex:activity456 .

ex:activity456 prov:qualifiedAssociation [
            prov:wasAssociatedWith ex:fred ] ;
    prov:hadPlan ex:rule937 .
```

Equational Background Knowledge. In the Semantic Web, information sometimes can be *inferred* deductively by applying ontological reasoning over suitably formalised background knowledge that often can be evaluated based on rules (e.g. reasoning about subproperties and subclasses, or entity consolidation using `owl:sameAs` inferences) [17]. Less common is equational knowledge defining functional dependencies among certain attributes of a resource.

For example, if we know that the city Bolzano has 54,031 female residents and a value of 109.0 for the Eurostat indicator "Women per 100 men", then we can compute a value of 49,570 male residents from the following equation:

$$\text{women per 100 men} = \frac{\text{population female} \cdot 100}{\text{population male}}$$

Two of our previous works [8, 16] have shown that it is possible to define rules to execute such equations, also considering 1) that equations are undirected, 2) that numeric datasets may be modelled with an arbitrary number of dimensions, and 3) that both forward-chaining inference and query rewriting are suitable approaches. In case no computation up to a fixpoint is needed, the executions were realised with simple SPARQL CONSTRUCT or INSERT queries (or, in off-the-shelf SPARQL engines by iteratively applying such queries).

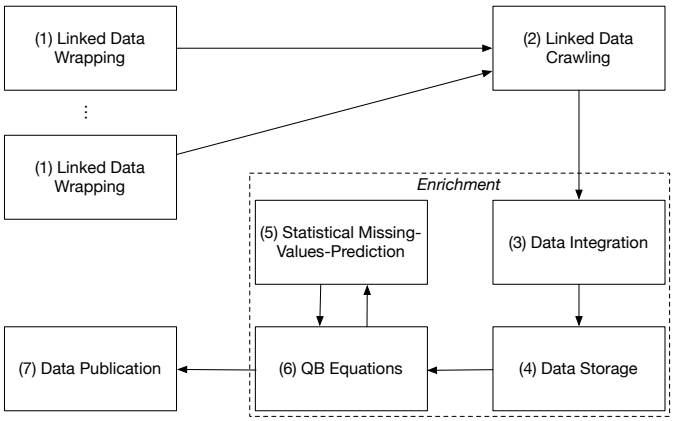

Figure 1: Open City Data Pipeline workflow

Missing Value Prediction. In our attempt to impute (predict) missing values for certain indicators and cities, our assumption is that every such indicator has its own distribution (e.g., normal, Poisson) and relationship to other indicators. Hence, we aim to evaluate different regression methods and choose the best fitting model to predict the missing values. In the field of Data Mining [18, 19] various regression methods for prediction were developed. We focus on well-established methods such as *K-Nearest-Neighbour Regression*, *Multiple Linear Regression* and *Random Forest Decision Trees*, since these methods are straightforward to apply and show a robust behaviour. In case the data is very sparse, these methods are not applicable since they require complete (subsets) of data. To this end, a common and robust method is the *regularised iterative PCA algorithm*: to first perform a Principal Component Analysis (PCA) to reduce the number of dimensions of the data set and use the new compressed dimensions, called *principal components* (PCs) as predictors [19, 20]. For measuring the quality of predictions (possibly used in equations), we use the *root mean squared error* (RMSE) and *normalised root mean squared error in %* (RMSE%) [18].

3. Overview and System Architecture

The workflow of the Open City Data Pipeline (OCDP) is illustrated in Figure 1 and consists of several steps:

1. Data is provided as Statistical Linked Data via wrappers which have to be created once per source in the Wrapping step.
2. A crawler collects data regularly (currently, weekly) from different sources in the Crawling step through the wrappers.
3. In the Data Integration step the data is integrated into the global cube, where data is enriched by links and heterogeneities resolved.

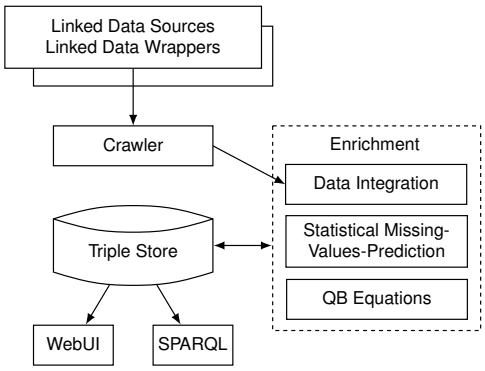

Figure 2: Open City Data Pipeline architecture

4. In the Data Storage step, the data is loaded into a SPARQL endpoint.
5. One further enrichment step exploits equational background knowledge in the form of QB equations.
6. Another further enrichment step applies statistical methods for missing values prediction.
7. Finally, in the Data publication step, the resulting enriched Linked data is made accessible.

In order to realise these steps, the architecture of the OCDP system implements several components. Figure 2 gives a high level overview of the architecture with a triple store being the central part. The data quality improvement workflow uses various methods to improve data quality and enrich the data.

We start with surveying data sources that serve as input to the pipeline in Section 3.1. We introduce the different components, their inputs, outputs, and interfaces in Section 3.2 and explain how we make the resulting data available in Section 3.3.

3.1. Data Sources

Many interesting statistical data sources are nowadays available. Many indicators in these data sources are provided on a country level and only a subset of indicators are available on the city level. We have identified the following potential providers of statistical data concerning cities:

- DBpedia[8];
- Wikidata[9];
- Eurostat with Urban Audit;
- United Nations Statistics Division (UNSD) statistics;
- U.S. Census Bureau statistics;
- Carbon Disclosure Project[10];
- individual city data portals.

In particular, we use statistical data from the United Nations and from Eurostat, which are integrated and enriched by the OCDP. The data sources contain data ranging from the years 1990 to 2016, but most of the data concerns the years after 2000. Further, not every indicator is covered over all years, where the highest coverage of indicators is between 2004 and 2015 (see Tables 1 and 2). Most European cities are contained in the Eurostat datasets. The UNSD contains the capital cities and cities with a population over 100 000, all listed in the United Nations Demographic Yearbook[11].

The previous OCDP of ISWC 2015 [7] contains data from 1990 to 2013 with 638,934 values from the Eurostat data source and 69,772 values from the U.N. data source. Due to some reorganisation in the Eurostat and U.N. datasets, Eurostat contains now 506,854 values and the U.N. provides 40,532 values. Regarding indicators, we now have 209 instead of 215 Eurostat and 64 instead of 154 U.N. indicators. The reason for the drop in indicators is due to the fact that the U.N publishes fewer datasets. The same effect can be seen for the cities, where we have 966 instead of 943 Eurostat and 3,381 instead of 4,319 U.N cities. Due to the smaller size of the datasets (see Tables 1 and 2), we now have an improved missing values ratio of 81.7% (before 86.3%) for Eurostat, resp. 94.4% (before 99.5%) for the U.N. dataset.

We now describe each of the data sources in detail.

Eurostat. Eurostat[12] offers various datasets concerning E.U. statistics. The data collection is conducted by the national statistical institutes and Eurostat itself. Of particular interest is the Urban Audit (UA) collection, which started as an initiative to assess the quality of life in European cities. UA aims to provide an extensive look at the cities under investigation, since it is a policy tool to the European Commission: "The projects' ultimate goal is to contribute towards the improvement of the quality of urban life" [21]. Currently, data collection takes place every three years (last survey in 2015) and is published via Eurostat Urban Audit. All data is provided on a voluntary basis which leads to varying data availability and missing values in the collected datasets. At the city level, Urban Audit contains over 200 indicators divided into the categories Demography, Social Aspects, Economic Aspects, and Civic Involvement. Currently, we extract the datasets that include the following topics:

- Population by structure, age groups, sex, citizenship, and country of birth
- Fertility and mortality
- Living conditions and education
- Culture and tourism
- Labour market, economy, and finance
- Transport, environment, and crime.

[8] http://dbpedia.org/
[9] http://wikidata.org/
[10] https://www.cdp.net
[11] http://unstats.un.org/unsd/demographic/products/dyb/dyb2012.htm
[12] http://ec.europa.eu/eurostat

Table 1: Values of the Eurostat Dataset

Year(s)	Cities	Indicators	Available	Missing	Missing Ratio (%)
1990	131	88	1 799	9 641	84.27
2000	433	163	6 420	63 996	90.88
2005	598	168	20 460	79 836	79.60
2010	869	193	56 528	110 996	66.26
2015	310	69	2 030	19 291	90.48
2004–2016	879	207	437 565	1 331 250	75.26
All (1990–2016)	966	209	506 854	2 257 171	81.66

Table 2: Values of the United Nations Dataset

Year(s)	Cities	Indicators	Available	Missing	Missing Ratio (%)
1990	5	3	8	7	46.67
2000	1 078	61	3 861	61 836	94.12
2005	777	61	2 110	45 226	95.54
2010	1 525	64	5 866	91 670	93.99
2015	216	3	568	77	11.94
2004–2016	2 095	64	28 849	511 759	94.66
All (1990–2016)	3 381	64	40 532	685 548	94.42

United Nations Statistics Division (UNSD). The UNSD offers data on a wide range of topics such as education, environment, health, technology and tourism. The focus of the UNSD is usually on the country level, but there are some datasets on cities available as well. Our main source is the UNSD Demographic and Social Statistics, which is based on the data collected annually (since 1948) by questionnaires to national statistical offices[13]. Currently we use the datasets on the city level that include the following topics:

- Population by age distribution, sex, and housing
- Households by different criteria (e.g., type of housing)
- Occupants of housing units / dwellings by broad types (e.g., size, lighting, etc.)
- Occupied housing units by different criteria (e.g., walls, waste, etc.)

The full UNSD Demographic and Social Statistics data has over 650 indicators, wherein we kept a set of 64 course-grained indicators and dropped the most fine-grained indicator level. For example, we keep *housing units total* but drop *housing units 1 room*. We prefer more coarse-grained indicators to avoid large groups of similar indicators which are highly correlated.

3.2. Pipeline Components

We now give an overview of each of the components of the OCDP system.

Statistical Linked Data Wrappers. Currently, none of the mentioned data sources publishes statistical data as Statistical Linked Data upfront. Thus, we use a set of wrappers which publish the data from these sources according to the principles listed in Section 2. Section 4.1 below explains these wrappers in more detail.

[13] http://unstats.un.org/unsd/demographic/

Linked Data Crawler. A Linked Data crawler starts with a seed list of URIs and crawls relevant connected Linked Data. The resulting RDF data is collected in one big RDF file and eventually loaded into the triple store. Section 4.4 explains the linked data crawler in more detail.

Triple Store. We use a standard Virtuoso 7 triple store as a central component to store data at different processing stages. For data loading we use the Virtuoso SQL console which allows faster data loading. For all other data access we rely on Virtuoso's SPARQL 1.1 interface which allows not only to query for data but with SPARQL Update also to insert new triples.

Enrichment Component: Data quality improvement workflow. In an iterative approach we improve data quality of the crawled raw data. This component covers steps (4)-(6) in the workflow shown in Figure 1: in this configurable workflow we use the several different sub-components corresponding to these steps consecutively. Each workflow component first reads input data (=observations) from the triple store via SPARQL queries, processes the data accordingly and inserts new triples into the triple store either via SPARQL Insert queries or the Virtuoso bulk loader facility (the first option is more flexible – it allows the execution of the workflow on a different machine – the second usually allows faster data loading).

The workflow currently uses three different subcomponents corresponding to steps (4), (5) and (6), respectively:

- The *Data Integration* sub-component, corresponding to step (3), performs some linking and data integration steps and materialises the global cube in a separate named graph. This linking and materialisation effectively resolves different types of heterogeneity found in the raw data: (i) different URIs for members, (ii) different URIs for dimensions, (iii) different DSDs (although the DSDs must be compatible to some extent for the integration to make sense). Eventually the global cube provides a unified view over many datasets from several sources. This component is implemented with SPARQL Update queries and supplied background knowledge for the integration. The exact process of linking statistical data and the materialisation will be described in more detail in Section 4.3 and Section 4.2 below.
- The *Statistical Missing Values Prediction* sub-component for missing value prediction, corresponding to step (5), extracts the whole global cube generated by the materialisation as one big data matrix, which is then used for applying different standard statistical regression methods to train models for missing value prediction. This component is implemented as a set of R scripts which extract the data with SPARQL queries. We then train and evaluate the models for each of the indicators. If the selected model delivers predictions in a satisfactory quality we apply the model and get estimates for the indicators. Finally the component exports the statistical data together with error estimates to one RDF file which is then loaded into the triple store with the Virtuoso bulk load feature and added

to the global cube. Section 5 below explains the details of this components in more detail.

- The *QB Equations* sub-component, corresponding to step (6), uses equations from different sources to infer even more data. To this end, we introduce QB equations. These QB equations provide an RDF representation format for equational knowledge and a semantics as well as a forward chaining implementation to infer new values. QB equations are implemented in a naive rule engine which directly executes SPARQL INSERT queries on the triple store. Section 6 introduces the concept of QB equations with syntax, semantics and implementation.

Lastly, in Section 6.3, we also explain the interplay between the Data Enrichment sub-components in more detail, that is – roughly – after cleansing and linking in component (4) we first run the QB equations component (6) once, to compute any values by equations that can be derived from the raw factual data alone, then approximate the remaining missing values by the statistical missing values prediction component (5), after which finally we run the QB equations component (6) again to improve predictions from (5) iteratively. As we will see in a detailed evaluation in Section 7, this iterative combination indeed performs better than using either (5) or (6) alone.

3.3. Data Publication

Eventually after the data is crawled and loaded into the triple store, improved and enriched by our workflow, the resulting global cube is available for consumption.

We provide a SPARQL endpoint[14] based on Virtuoso, where the global cube is stored in a named graph[15]. The prefix names used in the examples above are already set in Virtuoso, thus no prefix declarations are necessary for SPARQL queries.

We also provide a simple user interface[16] to query values for a selected indicator and city in the global cube. Queries are directly executed on the triple store during loading of the website using a JavaScript library called Spark; thus one can have a look at the SPARQL queries in the source code. We show all predicted values for transparency reasons. We simply order by the error value, i.e., the most trustworthy value per year is always shown first.

4. Data Conversion, Linking, and Integration

We now explain our approach for data conversion, linking, and integration. The approach is modular and extensible in the sense that every new data source can be prepared for consideration separately and independently from other sources. The data integration pipeline can be re-run at any time and thus allow for up-to-date data.

The approach consists of the following components:

- Linked Data wrappers that publish numerical data from various data sources as Statistical Linked Data (Section 4.1);
- the definition of a unified view over all relevant Statistical Linked Data (Section 4.2);
- semi-automatically generated links between Statistical Linked Data from different sources (Section 4.3);
- a rule-based Linked Data crawler to collect the relevant data and creates the unified view (Section 4.4).

4.1. Wrappers

We use Linked Data as interface to access and represent relevant data sources (e.g., Eurostat or UNSD), which are originally published in tabular form. The uniform Linked Data interface hides the specialities and structure of the original data source. When the wrapper receives an HTTP request for a particular dataset, it retrieves the data on-the-fly from the original source, transforms the tabular representation to RDF, using the RDF Data Cube vocabulary, and returns the RDF representation of the original tabular data.

The wrappers provide a table of contents with links to all available datasets (as a collection of `qb:DataSet` triples), including the data structure definition of the datasets (as `qb:DataStructureDefinition`). The individual data points are modelled as observations (as `qb:Observation`). The data structure definition includes the available dimensions (as `qb:dimension`) and concept schemes (as `skos:ConceptScheme`). We require a list of dataset and data structure definitions to be able to crawl the data.

Each wrapper coins URIs for identifying the relevant resources, for example, indicators or locations. We use URIs as unique identifiers for datasets, dimensions, and dimension values from different data sources.

The data sources identify indicators differently. For example, UNSD provides population numbers in dataset “240”, while Eurostat provides population numbers in dataset ”urb_cpop1”. We use the City Data Ontology to unify the various indicator identifiers. Similarly, locations have varying identifiers and sometimes varying names in the different data sources. For a relatively clear-cut example consider the city of Vienna: UNSD uses city code “001170” and label “WIEN”, whereas Eurostat uses code “AT001C1” and label “Wien”. The wrappers generate a Uniform Resource Identifier (URI) for every city out of the unique identifiers in the original tabular data.

We use the following wrappers that provide access to the underlying data source via a Linked Data interface:

Eurostat Wrapper. The Eurostat wrapper[17] makes the Eurostat datasets, originally available in tabular form at the Eurostat website, available as Linked Data. Eurostat provides several dictionary files in SDMX format; these files are used to construct a list of dimension values in the data structure definition and to generate URIs for relevant entities (such as cities). All files are accessed from the original Eurostat server once the wrapper

[14] http://citydata.wu.ac.at/ocdp/sparql
[15] http://citydata.wu.ac.at/qb-materialised-global-cube
[16] http://kalmar32.fzi.de/indicator-city-query.php
[17] http://estatwrap.ontologycentral.com/

receives a HTTP request on the particular URI, ensuring that the provided RDF data is up-to-date. Population data in the Eurostat wrapper[18] uses http://estatwrap.ontologycentral.com/dic/indic_ur#DE1001V to identify "Population on the 1st of January, total". The indicator URI is mapped to indicator URIs from the City Data Ontology in a subsequent step.

UNSD Wrapper. The UNSD wrapper[19] makes the UNSD datasets, originally available in tabular form at the UNSD website, available as Linked Data. The UNSD wrapper provides a simple data structure definition describing the available dimensions and measure. In total, we cover 14 datasets ranging from population to housing data. Most indicators, e.g., population of the "240" dataset,[20] are directly mapped to an indicator URI from the City Data Ontology, namely http://citydata.wu.ac.at/ns#population.

4.2. Unified View over Statistical Linked Data

As the different data sources use different identifiers (and the wrappers use different URIs), we need to link the varying URIs before we can do an integrated querying of the data. As the foundation for efficiently querying Statistical Linked Data – and in turn enriching the data as described in Section 5 and Section 6 – we define a unified view of all crawled datasets about cities in a simplified version of the global cube [16]. In the following, we describe the structure of the global cube.

We define the unified view as the basis for querying as follows. The qb:Observations (consisting of dimensions and measures) have the following structure, starting with the dimensions:

- For the time dimension we use dcterms:date.
- For the time dimension values we use single years represented as String values such as "2015".
- For the geospatial dimension we use sdmx-dimension:refArea, which is recommended by the QB standard.
- For the geospatial dimension values we use instances of dbpedia:City, such as dbpedia:Vienna.
- For the indicator dimension we use cd:hasIndicator.
- For the indicator dimension values we use instances of cd:Indicator, such as cd:population_female. For the indicator dimension values, we defined the CDP ontology as the main hub of indicator URIs to link to since there was no list with common indicator values.

Most data source follow the practice of using an unspecific measure sdmx-measure:obsValue and a dimension indicating the measured variable, e.g., estatwrap:indic_na. For the unified view, we thus also assume data cubes to have only one general measure, sdmx-measure:obsValue. Please note that there are different equivalent alternative representations of the same information. Specifically for measure properties, in QB there is a choice for the structuring of the observations. Either use a single observation value property and a dedicated indicator dimension, or encode the indicator in the measure property. To sum up: in-line with established usage, we use a single measure property, but that structure contains all the information that would also be present in the alternative representation.

If we want to pose queries over the two datasets, we have two options. Either specifically write the query to consider possibly different identifiers (i.e., need to know all identifiers) or 2) assume existing links and reasoning. Then, if we query for values for the canonical identifiers (as for any other identifier in the equivalence class), we also get the values for the respective other identifiers. In the paper, we assume reasoning to allow for flexible addition of new sources without the need to change the queries for each new data source.

Take as an example we want a query all values of the indicator "population" of the area "Vienna", in the year "2010" over data from both datasets. The indicator would be expressed as a dimension, with a URI representing "population" as dimension value. The area would be expressed with a dimension, with a URI representing "Vienna" as dimension value. The query looks like the following:

```
SELECT ?city ?year ?value
WHERE {
 ?obs cd:hasIndicator cd:population ;
      sdmx-dimension:refArea dbpedia:Vienna ;
      dcterms:date ?year ;
      sdmx-measure:obsValue ?value .
}
```

Our unified view uses the basic modelling features of the QB vocabulary. In particular, we model indicators in a way that include what otherwise might be encoded as separate dimensions. In the more complex modelling, we would need to use the union of all dimensions of the source datasets, which would lead to introducing an"ALL" dimension value for those dimensions that are not distinguished by the particular dataset (see [16] for details on this more normalised representation). However, all the newly introduced dimensions per dataset would need to be considered in querying which complicates the queries. Rather than adding a dimension "sex" to encode gender, we create separate indicator URIs, for example for population, population male and population female. A benefit of the relatively simple structure is that queries and rules operating on the unified view are also simple.

We have published the data structure definition of the global cube using the QB vocabulary. Besides the general measure (sdmx-measure:obsValue), the qb:DataStructureDefinition of the global cube uses the mentioned dimensions dcterms:date, sdmx-dimension:refArea, and cd:hasIndicator. Also, we have defined instances of qb:AttributeProperty for cd:estimatedRMSE (for describing the error), cd:preferredObservation (for linking to more reliable values), prov:wasGeneratedBy (for describing provenance information) and prov:generatedAtTime (for the time of generation) that help to interpret and evaluate the trustworthiness of values.

Please note that data sources use different identifiers for

[18] http://estatwrap.ontologycentral.com/id/urb_cpop1
[19] http://citydata.wu.ac.at/Linked-UNData/
[20] http://citydata.wu.ac.at/Linked-UNData/data/240

dimensions and dimension URIs. In the global cube, we use canonical URIs to represent resources from different data sources.

4.3. Linking and Mapping Data

We start by explaining the required mappings for dimension URIs, followed by explaining the required mappings for dimension value URIs. In general, the data from the UNSD wrapper, due the simpler representation in the original data source, requires less mappings than the data from the Eurostat wrapper.

The two data sources exhibit the three dimensions for `dcterms:date` (year), `sdmx-dimension: refArea` (city) and `cd:hasIndicator` (indicator). We map the following dimension URIs of the global cube using `rdfs:subPropertyOf`:

- For the time dimension our wrappers directly use `dcterms:date`. The time dimension hence does not require any further mapping.
- For the geospatial dimension the UNSD wrapper uses `sdmx-dimension:refArea`. The Eurostat wrapper uses different representations for the geospatial dimension, such as `eurostat:geo`, `eurostat:cities` and `eurostat:metroreg`, which we link to `sdmx-dimension:refArea`.
- For the indicator dimension we use `cd:hasIndicator`. Again, the UNSD wrapper directly uses that URI, while the data from the Eurostat wrapper requires links from `eurostat:indic_na` and `eurostat:indic_ur` to `cd:hasIndicator`.

The Eurostat site provides a quite elaborate modelling of dimensions, code lists and so on in SDMX files. The datasets from the Eurostat wrapper use various units such as `:THS` denoting "Thousand" and `:COUNT` denoting that the number was computed from a count operation. However, all other dimensions of datasets from Eurostat we consider in the pipeline besides the three canonical dimensions of the global cube exhibit only one single possible dimension value (e.g., `:THS`). Hence, we can assume that all other dimensions and their values are part of the indicator.

The UNSD site has a simpler structure than Eurostat. The modelling of different dimensions and code lists is less elaborate. Thus, for the UNSD wrapper, we have ensured on the level of the published RDF that each dataset only provides the canonical dimensions.

The two wrappers use different URIs for the same dimensions, e.g., `eurostat:geo` and `sdmx-dimension:refArea`. The wrappers also use different URIs for the same dimension values, e.g.,

- For the time dimension values we use single years represented as String values such as `"2015"`.
- For the geospatial dimension values we link to DBpedia URIs from other representations such as `http://estatwrap.ontologycentral.com/dic/cities#AT001C1` and `http://citydata.wu.ac.at/resource/40/001170#000001`.
- For the indicator dimension values we link to instances of `cd:Indicator`, such as `cd:population` and `cd:population_male`. The UNSD wrapper directly uses these values. For the URIs used in the data from the Eurostat wrapper, we link to instances of `cd:Indicator`.

We now describe how we generated these links to map data from different sources to the canonical representation, starting with the dimension and dimension value URIs. We manually created the `rdfs:subPropertyOf` triples connecting the Eurostat dimension URIs with our canonical URIs, and semi-automatically generated the indicator URIs from an Excel sheet provided by Eurostat. We then created an RDF document with links from the newly generated URIs to the URIs of the Eurostat wrapper. We manually adapted the UNSD wrapper to use the newly generate URIs as indicator URIs.

We choose to have a one-to-one (functional) mapping of every city from our namespace to the English DBpedia URI, which in our re-published data is encoded by `owl:sameAs` relations. We identify the matching DBpedia URIs for multilingual city names and apply basic entity recognition, similar to Paulheim et al. [22], with three steps using the city names from UNSD data:

- Accessing the DBpedia resource directly and following possible redirects.
- Using the Geonames API [21] to identify the resource.
- For the remaining cities, we manually looked up the URI on DBpedia.

The mappings of geospatial URIs from the Eurostat wrapper were done in a similar fashion. All the mappings are published online as RDF documents that are accessed during the crawling step.

4.4. Data Crawling and Integration

The overall RDF graph can be published and partitioned in different documents. Thus, to access the relevant RDF documents, the system has to resolve the URIs of entities related to the dataset. Related entities are all instances of QB-defined concepts that can be reached from the dataset URI via QB-defined properties. For example, from the URI of a `qb:DataSet` instance, the instance of `qb:DataStructureDefinition` can be reached via `qb:structure`. Similarly, instances of `qb:ComponentProperty` (dimensions/measures) and `skos:Concept` (members) can be reached via links.

Once all numeric data is available as Linked Data, we need to make sure to collect all relevant data and metadata starting from a list of initial URIs. First, the input to the crawling is a seed list of URIs of instances of `qb:DataSets`. One example of a "registry" or "seed list" of dataset URIs is provided by the PlanetData wiki[22]. A seed list of such datasets is published as

[21] http://api.geonames.org/

[22] http://wiki.planet-data.eu/web/Datasets

RDF and considered as input to the crawling. We use two such seed lists: one with links to the relevant instances of qb:DataSet from the UNSD wrapper, and another one with links to the relevant instances of qb:DataSet from the Eurostat wrapper.

Then, Linked Data crawlers deploy crawling strategies for RDF data where they resolve the URIs in the seed list to collect further RDF and in turn resolve a specific (sub-)set of contained URIs. An example Linked Data crawler is LDSpider [23], which uses a depth-first or breadth-first crawling strategy for RDF data. Linked Data crawlers typically follow links without considering the type.

A more directed approach would apply a crawling strategy that starts with resolving and loading the URIs of qb:DataSets relevant for the task, and then in turn resolves and loads instances of QB concepts that can be reached from the dataset URIs.

To specify how to collect Linked Data, we use the Linked Data-Fu language [9] in which rule-based link traversal can be specified. For instance, to retrieve data from all qb:DataSets, we define the following rule:

```
{
  ?ds rdf:type qb:DataSet.
} =>
{
  [] http:mthd httpm:GET .
     http:requestURI ?ds .
} .
```

The head of a rule corresponds to an update function of an internal graph representation in that it describes an HTTP method that is to be applied to a resource. In our example, the head of a rule applies a HTTP GET method to the resource ?ds. The body of a rule corresponds to the condition in terms of triple patterns that have to hold in the internal graph representation. In our example, ?ds is defined as an instance of qb:DataSet.

Similarly, we retrieve instances of qb:DataStructureDefinition, qb:ComponentSpecification, qb:DimensionProperty, qb:AttributeProperty, qb:MeasureProperty, qb:Slice, qb:SliceKey, and qb:ObservationGroup. Also, we access the list of possible dimension values (based on qb:codeList in data structure definitions) as well as each single dimension value. The only instances we do not resolve are observations since these are usually either modelled as blank nodes or provided together with other relevant information with the RDF document containing qb:DataSet or qb:Slice.

Crawling may include further information, e.g., rdfs:seeAlso links from relevant entities or owl:sameAs links to equivalent URIs. Assuming that the number of related instances of QB concepts starting from a QB dataset is limited and that links such as rdfs:seeAlso for further information are not crawled without restriction (e.g., only from instances of QB concepts), the directed crawling strategy terminates after a finite amount of steps.

Besides all the relevant data and metadata of qb:DataSets, we collect the following further information:

- The City Data Ontology[23] (CDP ontology) that contains lists of common statistical indicators about cities.
- The QB Equations Ontology[24] that contains the vocabulary to describe QB equations and is further detailed in Section 6.
- The Eurostat QB equations[25] that contains a set of QB equations generated from formulas published by Eurostat as further detailed in Section 6.
- Background information[26] that links indicators of Estatwrap to the CDP ontology as further described in Section 4.3.
- Background information providing additional owl:equivalentProperty links[27] between common dimensions not already provided by the wrappers such as between the different indicator dimension URIs estatwrap:indic_ur, cd:hasIndicator and eurostat:indic_na.

Besides explicit information available in the RDF sources, we also materialise implicit information to 1) make querying over the triple store easier and 2) automatically evaluate relevant QB and OWL semantics. We execute the QB normalisation algorithm[28] in case the datasets are abbreviated. Also, we execute entailment rules[29] for OWL and RDFS. However, we only enable those normalisation and entailment rules that we expect to be evaluated quickly and to provide sufficient benefit for querying.

For instance, we evaluate rules about the semantics of equality, e.g., symmetry and transitivity of owl:sameAs. We again describe the semantics of such axioms using Linked Data-Fu. However, because we do not need the full materialisation of the equality, but only the canonical URIs, we define custom rules that only generate the triples involving the canonical URIs. Thus, the resulting dataset contains all triples required to integrate and query the canonical representation, but not more.

The crawling and integration is specified in several Linked Data-Fu programs. The programs are executed periodically using the Linked Data-Fu interpreter[30] in version 0.9.12. The interpreter issues HTTP requests to access the seed list, follows references to linked URIs, and applies the derivation rules to materialise the inferences. The crawled and integrated data is then made available for loading into a triple store. Before loading the observations into the triple store we ensure for each observation that the correct dimension URIs and member URIs are used, filter out non-numeric observation values and mint a new observation URI if a blank node is used. Finally, the filtered and skolemised observations are loaded into an OpenLink Virtuoso triple store (v07) using the standard RDF bulk loading feature[31].

Thus, the global cube can be queried in subsequent imputation and calculation steps.

[23] http://citydata.wu.ac.at/ns

[24] http://citydata.wu.ac.at/ocdp/qb-equations

[25] http://citydata.wu.ac.at/ocdp/eurostat-equations

[26] http://kalmar32.fzi.de/triples/indicator-eurostat-links.nt

[27] http://kalmar32.fzi.de/triples/dimension-property-links.nt

[28] https://www.w3.org/TR/vocab-data-cube/#normalize-algorithm

[29] http://semanticweb.org/OWLLD/

[30] https://linked-data-fu.github.io/

[31] See http://citydata.wu.ac.at/ocdp/import for a collection of information about the loading process.

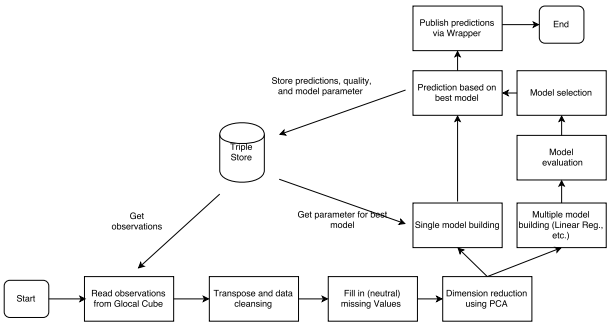

Figure 3: Prediction Workflow

5. Imputation: Predicting Missing Values

As discussed in Section 1 and Section 3.1, the filling in of missing values by reasonable predictions is a central requirement for the OCDP, since we discovered a large number of missing values in our datasets (see Table 1 and 2).

The prediction workflow is given in Figure 3. The initial step regards the loading, transposing, and cleansing of the observations taken from the global cube. Then, for each indicator, we impute all the missing values with neutral values for the principal components analysis (PCA), and perform the PCA on the new matrix, which creates the principal components (PC) that are used as predictors. Next, the predictors are used for the model building step using a basket of statistical Machine learning methods such as multiple linear regression. Finally, we use the best model from the basket to fill in all missing value in the original matrix and publish them using the Missing Values wrapper.

In our earlier work [7], we have evaluated two approaches to choose the predictors, one based on applying the base methods to complete subsets in the data and the other based on PCA. In the present paper, we only use the PCA-based approach, since, although it delivers slightly lower prediction accuracy, it allows us to cope more robustly with the partially very sparse data, such that we can also predict values for indicators that do not provide sufficiently large subsets of complete, reliable predictors.

Base Methods. Our assumption is that every indicator has its own statistical distribution (e.g., normal, exponential, or Poisson distribution), sparsity, and relationship to other indicators. Hence, we aim to evaluate different regression methods and choose the best fitting method/model to predict the missing values per indicator. In order to find this best fitting method, we measure the prediction accuracy by comparing the *normalised root mean squared error* in % (RMSE%) [18] of every tested regression method. While in the field of Data Mining [18, 19] (DM) numerous regression methods for missing value prediction were developed, we chose the following three "standard" methods for our evaluation due to their robustness and general performance:

K-Nearest-Neighbour Regression (KNN), models denoted as M_{KNN}, is a wide-spread DM technique based on using a distance function to a vector of predictors to determine the target values from the training instance space. As stated in [19], the algorithm is simple, easily understandable and reasonably scalable. KNN can be used in variants for clustering as well as regression.

Multiple Linear Regression (MLR), models denoted as M_{MLR}, has the goal to find a linear relationship between a target and several predictor variables. The linear relationship can be expressed as a regression line through the data points. The most common approach is *ordinary least squares* to measure and minimise the cumulated distances [19].

Random Forest Decision Trees (RFD), models denoted as M_{RFD}, involve the top-down segmentation of the data into multiple smaller regions represented by a tree with decision and leaf nodes. Each segmentation is based on splitting rules, which are tested on a predictor. Decision nodes have branches for each value of the tested attribute and leaf nodes represent decision on the numerical target. A random forest is generated by a large number of trees, which are built according to a random selection of attributes at each node. We use the algorithm introduced by Breiman [24].

Principal Component Analysis. All three of the above-described base methods need a complete data matrix as a basis for calculating predictions for the respective target indicator column. Hence, we need for each target indicator (to be predicted) a complete training data subset of predictor indicators. However, as discussed in [7], when dealing with very sparse data, such complete subsets are very small and would allow us to predict missing values only for a few indicators and cities. Instead, we omit the direct use of indicators as predictors. Instead, we first perform a PCA to reduce the number of dimensions of the data set and use the new compressed dimensions, called *principal components* (PCs) as predictors for the above described three base methods: as stated in [19], the PCA is a common technique for finding patterns in data of high dimensions (in our case, many different indicators for many different cities and years). We use PCA to compress the large number of indicators to a smaller set of principal components which can later be used as predictors. The second main advantage of PCA is in terms of dealing with sparse data: as described in [20], all the missing values in the raw data matrix can be replaced by *neutral* values for the PCA created according to the so-called *regularised iterative PCA algorithm*. This step allows to perform PCA on the entire data matrix, even if only a few complete subsets exist.

5.1. Preprocessing

Before we can apply the PCA and subsequently the base regression methods we need to pre-process and prepare the data from the global cube to bring it into the form of a two-dimensional data matrix. This preprocessing starts with the extraction of the observations from the global cube. Since the described standard DM methods can not deal with the hierarchical, multi-dimensional data of the global cube, we need to "temporary flatten" the data back to tuples. For this, we pose the following SPARQL query, with an increasing year range that is currently *2004–2017*.

```
SELECT DISTINCT ?city ?indicator ?year  ?value
FROM <http://citydata.wu.ac.at/qb-materialised-global-cube>
WHERE {
```

```
?obs dcterms:date ?year.
?obs sdmx-dimension:refArea ?city.
?obs cd:hasIndicator ?indicator.
?obs sdmx-measure:obsValue ?value.
{ ?obs a cd:CrawledObservation } UNION { ?obs a cd:
     factualQBeObservation }.

FILTER(xsd:integer(?year) >= 2004)

} ORDER BY ?indicator ?city ?year
```

The SPARQL query flattens the multidimensional data to an input data table with tuples of the form:

$$\langle City, Indicator, Year, Value \rangle.$$

Based on the initial table, we perform a simple preprocessing as follows:

- Removing nominal columns and encode boolean values;
- Merging the dimensions year and city to one, resulting in: $\langle City\ Year, Indicator, Value \rangle$;
  that is, we further flatten the consideration of city per year to city/year "pairs"
- Finally, we transpose the initial table to a two-dimensional data matrix with one row per city/year-pair one column per indicator, resulting in tuples of the form: $\langle City\ Year, Indicator_1 Value_1, \ldots, Indicator_n Value_n \rangle$;
- From this large matrix, we delete columns and rows which have a missing values ratio larger than 99%, that is, we remove city/year pairs or indicators that have too many missing values to make reasonable predictions, even when using PCA.

Our initial data set from merging Eurostat and UNSD contains 1 961 cities with 875 indicators. By merging city and year and transposing the matrix we create 12 008 city/year rows. After deleting the cities/year-pairs and indicators with a missing values ratio larger than 99%, we have the final matrix of 6 298 rows (city/year) with 212 columns (indicators).

Note that the flattening approach and deletion of too sparse rows/columns are generic and could obviously still be applied if we added more data sources, but our experiments herein focus on the Eurostat and UNSD data.

5.2. Prediction using PCA and the Base Regression Methods

Next, we are ready to perform PCA on the data matrix created in the previous subsection. That is, we *impute* all the missing values with *neutral* values for the PCA, according to the above-mentioned *regularised iterative PCA algorithm* described in [20]. In more detail, the following steps are evaluated having an initial data set A_1 as a matrix and a predefined number of predictors n (we test this approach also on different $n's$):

1. Select the target indicator I_T;
2. Impute the missing values in A_1 using the regularised iterative PCA algorithm resulting in matrix A_2 and remove the column with I_T;

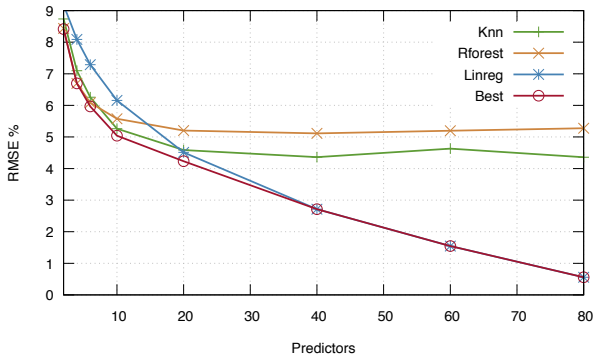

Figure 4: Prediction results using PCA

3. Perform the PCA on the A_2 resulting in a matrix A_3 of a maximum of 80 PCs;
4. Append the column of I_T to A_3 creating A_4 and calculate the correlation matrix A_C of A_4 between I_T and the PCs;
5. Create the submatrix A_5 of A_4 on the selection of the PCs with the highest absolute correlation coefficients and limit them by n;
6. Create submatrix A_6 of A_5 for validation by deleting rows with miss. values for I_T;
7. Apply stratified tenfold cross-validation on A_6. which results in the best performing model M_{Best};
8. Use the method for M_{Best} to build a new model on A_5 (not A_6) for predicting the missing values of I_T.

5.3. Evaluation and Publishing

Figure 4 shows the results for the median RMSE% with an increasing number of predictors (selected from the 80 PCs) and compares the performance of KNN, RFD, MLR, and the selection of best method. Clearly, for 80 predictors MLR performs best with a median RMSE% of 0.56%, where KNN (resp. RFD) has a median RMSE% of 4.36% (resp. 5.27%). MLR is the only method that improves steady up to 80 predictors. KNN provides good results for a lower number of predictors, but starts flattening with 20 predictors. Contrary to MLR, the parameter of KNN and MLR have to be adjusted according to number of predictors, hence optimising the number of clusters for KNN could improve the result. The red line in Figure 4 shows the median RMSE% with the best regression method chosen. Up to 60 predictors, the overall results improves by selecting the best performing method (for each indicator). The best median RMSE% of 0.55% is reached with 80 predictors, where MLR is predominant and only 5 out of 232 indicators are predicted by KNN. We emphasise that, compared to the result of our earlier experiments in [7], the median RMSE% improved from 1.36% to 0.55%, which is mainly related to the lower sparsity of the datasets.

Finally, we note again why we added PCA, as opposed to attempting predictions based on complete subsets: in our preliminary evaluations, based on the comparison of the two approaches in [7], by picking the best performing regression

method per indicator with ten predictors from the raw data based on complete subsets the median RMSE% was 0.25%. However, due to the low occurrence of complete subsets of reasonable size for ten predictors, only one third of the missing values could be imputed compared to using PCA. We acknowledge that this comes at a cost, as the a median RMSE% when using PCA goes up to 0.55% with 80 predictors (see above). However, due to the sparsity in the data we decided to trade better completeness for accuracy of the prediction.

We publish the predicted values created by the combination PCA and selecting the best regression method per indicator where we apply a threshold of RMSE% of 20% as a cut off. This leads with the current evaluation to no removal of any indicator. Following our strategy of using statistical linked data wrappers, we publish the predicted values using the *Missing Values wrapper*,[32] which provides a table of content, a structure definition, and datasets that are created for each prediction execution.

5.4. Workflow and Provenance

The full prediction workflow of our statistical prediction for missing values is shown in Figure 3 and is based on all observations but the old predicted values in the global cube. The *data preprocessing and transposing* for the input data matrix is written in Python, but all other steps such as *PCA*, *model building*, and *model evaluation* are developed in R [25] using its readily available "standard" packages (another advantage of relying on standard regression methods). All the scripts and their description are available on the website of the *Missing Values wrapper*. We conducted an evaluation of the execution time on our Ubuntu Linux server with 2 cores, 2.6 GHz, and 16 GB of RAM. A single prediction run requires approx. 10min for each indicator (approx. 3 min for each method) resulting in a total time of about 35 hours for all indicators, which still is reasonably doable for re-running wrappers, recomputing models and predictions in a weekly batch job.

Looking back to Figure 3, one can see that the workflow branches after four steps, where we distinguish two cases. In the case of no previous executions, we perform the full prediction steps as described in the previous section. In the case of previous executions, we already have provenance information available in our triple store, which describes the last execution and the related model provenance information (for each indicator). The model provenance includes for each indicator the number of PCs, the number of predictors used from these PCs, the chosen prediction base method, method parameters (i.e., the number of clusters in the KNN), and the RMSE%.

To sum up, we keep provenance for our predictions on three levels:

- For each execution, we publish the median RMSE% over all indicators, number of predictors, creation date, and the creation agent;
- For each indicator, we publish the above mentioned model provenance data;
- For each predicted value published as a `qb:Observation`, we provide the overall absolute estimated RMSE (using the predicate `cd:estimatedRMSE`) and the estimated RMSE% (using the predicate `cd:estimatedNormalizedRMSE`). Further, we point to better observations (published with an lower RMSE%) using the predicate `cd:preferredObservation` which might occur if another approach such as a different base method or QB Equations (discussed in Section 6 below) improve the predicted values.

For describing the model provenance, we use the *MEX vocabulary*, which is compared to other vocabularies (i.e., DMOP [26]) lightweight and designed for exchanging machine learning metadata [27]. We use the *MEX Algorithm* layer to describe our prediction method and its parameter and the *MEX Performance* layer to describe the RMSE%. Further, we describe each execution using attributes of *MEX Execution*.

Example 5.1. The following example gives an intuition into reading the data about missing value predictions.

```
@prefix prov: <http://www.w3.org/ns/prov#> .
@prefix cdmv: <http://citydata.wu.ac.at/MV-Predictions/> .
@prefix mexp: <http://mex.aksw.org/mex-perf/> .
@prefix mexc: <http://mex.aksw.org/mex-core#> .
@prefix mexa: <http://mex.aksw.org/mex-algo#> .

cvmv:predDS1 rdf:type qb:DataSet .
cvmv:predDS1 prov:wasGeneratedBy cvmv:runP1 .
cvmv:predDS1 dc:title "A3_2004-2016_ncp80_seed_100_pred_80" .

cvmv:runP1 rdf:type  mexc:Execution ; prov:Activity .
cvmv:runP1 cdmv:predictionPCs 80 .
cvmv:runP1 mexp:rootMeanSquaredError 1.0705 .
cvmv:runP1 mexc:endsAt "2017-07-31T10:52:02Z"^^xsd:dateTime .

cvmv:runP1 cvmv:hasPredicted  cvmv:runP1_1 .
cvmv:runP1_1 mexc:datasetColumn cd:no_bed-
     places_in_tourist__accommodation_establishments .
cvmv:runP1_1 mexc:hasAlgorithmConfig mexa:Regression .
cvmv:runP1_1 cd:estimatedAbsoluteRMSE 3228.8726 .
cvmv:runP1_1 cd:estimatedNormalizedRMSE 1.78259 .
cvmv:runP1_1 cdmv:size 2737 .


cdmv:obs1 rdf:type cd:PredictedObservation.
cdmv:obs1 cd:hasIndicator no_bed-
     places_in_tourist__accommodation_establishments .
cdmv:obs1 sdmx-dimension:refArea dbpedia:Bolzano .
cdmv:obs1 dcterms:date "2010" .
cdmv:obs1 sdmx-measure:obsValue 1490.4485 .
cdmv:obs1 cd:estimatedAbsoluteRMSE 3228.8726 .
cdmv:obs1 cd:estimatedNormalizedRMSE 1.78259 .
cdmv:obs1 cd:preferredObservation cdmv:obs1 .
cdmv:obs1 qb:dataSet cvmv:predDS1 .
```

The example shows a `qb:DataSet` of predicted values generated by a run on the 2017-07-31 using our PCA-based approach. We show one predicted value and its RMESs for the indicator `no_bed-places_in_tourist__accommodation_establishments` of the city of Bolzano in the year 2010. The best method for this indicator was MLR which is indicated by the triple: `cvmv:runP1_1 mexc:hasAlgorithmConfig mexa:Regression`.

The triple `cdmv:obs1 cd:preferredObservation cdmv:obs1` states that currently there is no better prediction available, i.e., that this observation is itself the most preferred (i.e., best) for the respective indicator for this city/year.

In summary, while through the availability of more and new raw data we could improve the prediction quality compared to [7], this is – essentially, apart from the more structured workflow

[32] http://citydata.ai.wu.ac.at/MV-Predictions/

and publication using provenance information for all predictions – where we stopped missing value prediction in our earlier work in [7]. What we will show next in Section 6 is that prediction quality can be further improved by combining the statistical regression methods from this section with ontological background knowledge in the form of equations.

6. Calculation: QB Equations

OWL 2 gives only little support for reasoning with literal values. Although knowledge about the relations of different numeric literals (equational knowledge) exists even in ontologies, for example the QUDT ontology [28], it can not be used to infer new literal values by an OWL reasoner since OWL 2 in general can not compute new (numeric) literals. For statistical data, especially statistical linked data, equational knowledge can be interesting to compute derived indicators or fill in the gaps of missing values. Examples for useful equational knowledge include unit conversion, indicator definitions, or linear regression models. With QB equations (QBe) we introduce a framework to exploit equational knowledge to infer numerical data using Semantic Web technologies, with fine-grained provenance tracking and error propagation for inaccurate values. After applying the ML prediction methods in the OCDP workflow, the QBes generate observations from the whole combined dataset. The resulting observations are used for evaluation, consistency checking, and are published if they are new or better than any existing observation with the same dimension members.

Example 6.1. The Eurostat indicator "Women per 100 men" is defined as an equation as follows:

$$\texttt{women\_per\_100\_men} = \frac{\texttt{population\_female} \cdot 100}{\texttt{population\_male}}$$

The approach of QBes presented in the following is a combination and extension of two earlier approaches "RDF attribute equations" and "complex correspondences". RDF attribute equations [8] use equational knowledge and give an RDF syntax and a Description Logic semantics to derive numerical OWL data properties from other such properties. The approach was implemented in a backward-chaining manner for SPARQL queries. QBes however, operate on QB observations instead of OWL data properties, are implemented in a forward-chaining manner and provide error propagation and fine-grained provenance tracking. Complex correspondences [16] define rules, with numerical functions, to compute QB observations from other QB observations. They transfer the concept of correspondences over relational data to the Semantic Web data model using the QB vocabulary. In contrast to complex correspondences, QBes are given in an RDF syntax and is more generic since it uses (more general) equations instead of functions resulting in more computed values without the need to (manually) create one rule for each variable in the equation.

6.1. QB Equation Syntax

We express QBes in an RDF syntax. Since – to the best of our knowledge – no vocabulary exists for this purpose so far we have to introduce a new vocabulary expressing QBes and QB rules.

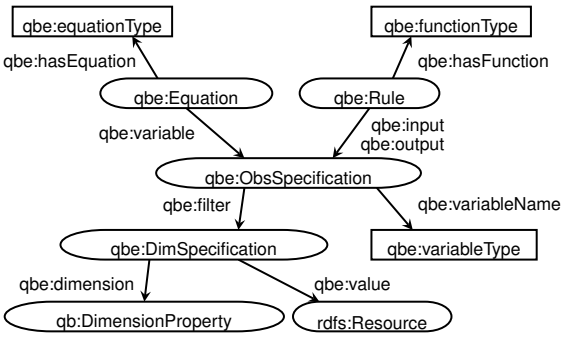

Figure 5: QB equations ontology

Each QBe is identified by a URI and consists of two parts: (i) a representation of a mathematical equation using (arithmetic) functions and variables, and (ii) a mapping of observations to variables using observation specifications.

Figure 5 gives an overview of the QB equations ontology showing all the introduced classes, properties, and datatypes as well as reuse of the QB ontology. When encoded in RDF we call these relationships *QB equations* or *QB rules*. QBes specify relationships between observations which can be reformulated into different "directions" while QB rules are valid only in one direction.

Representation of the Equation or Function. One possibility to represent equations in RDF would be building an operator tree in RDF like the MathML, an XML representation of mathematical concepts, including equations. The SWRL RDF syntax also uses such a verbose syntax for example for predefined mathematical functions.

To keep the representation simple and still human-readable without a custom UI we define the core of the QBes, which is the equation itself, as a literal that directly reuses SPARQL's arithmetic expression syntax,[33] i.e., we use a datatype literal with the datatype `qbe:equationType`, the lexical space of which is defined by the following grammar rule (in the syntax and referring to non-terminal symbols of the SPARQL grammar):

equationType ::= Var '=' NumericExpression

This choice enables standard SPARQL parsers or other standard libraries for mathematical expressions for processing these equations, and – as we will see – straightforward implementation of the application of equations by SPARQL engines. As an example we give again the equation for the Eurostat indicator definition of "Women per 100 men":

```
"?women_per_100_men = ?population_female * 100 / ?
    population_male"^^qbe:equationType
```

The property `qbe:hasEquation` relates an instance of the class `qbe:Equation` to such an equation literal.

[33] cf. http://www.w3.org/TR/sparql11-query/#rNumerivExpression

The lexical space of datatype qbe:variableType is – analogous to qbe:equationType – defined by the SPARQL grammar non-terminal 'Var'.

Observation Specification. The second part maps observations to the variables used in the equation. Usually observations are specified by giving values for all of the dimensions. This approach would be too constraining and might lead to multiple representations of essentially the same equation. Instead an observation specification only needs values for some of the dimensions. In an example of unit conversions, one would only specify the value of the unit dimension because the equation should be applicable to any kind of observation given in that unit, regardless of the values of the other dimensions. Intuitively the values of all other unspecified dimensions must be the same among all specified observations.

Example 6.2. The following example shows the complete definition of the equation for the Eurostat indicator "Women per 100 men". The QBe defines a variable ?women_per_100_men which binds to all observations for which the member of the dimension estatwrap:unit is set to cd:women_per_100_men. The other two variables ?mile are defined analogously. Eventually the QBe gives the equation relating the variables as a qbe:equationType-typed literal.

```
ex:women-per-100-men a qbe:Equation ;
  qbe:variable [ a qbe:ObsSpecification ;
    qbe:filter [ a qbe:DimSpecification ;
      qb:dimension cd:hasIndicator ;
      qbe:value cd:women_per_100_men ] ;
    qbe:variablename "?women_per_100_men"^^qbe:variableType ] ;
  qbe:variable [ a qbe:ObsSpecification ;
    qbe:filter [ a qbe:DimSpecification ;
      qb:dimension cd:hasIndicator ;
      qbe:value cd:population_male ] ;
    qbe:variablename "?population_male"^^qbe:variableType ] ;
  qbe:variable [ a qbe:ObsSpecification ;
    qbe:filter [ a qbe:DimSpecification ;
      qb:dimension cd:hasIndicator ;
      qbe:value cd:population_female ] ;
    qbe:variablename "?population_female"^^qbe:variableType ] ;
  qbe:hasEquation "?women_per_100_men = ?population_female *
       100 / ?population_male"^^qbe:equationType.
```

The type declarations are only given for completeness and are not necessary in practise.

QBes can be evaluated in multiple directions, effectively creating a function to compute a value for each of the variables from all the other variables. In the example above we can infer observations for each of the three indicators cd:women_per_100_men, cd:population_female, and cd:population_male from the other two. Obviously this works only for invertible functions including the usual arithmetic operators: addition, subtraction, multiplication, and division[34]. In fact, we can reuse the definition of simple equations from [8], which guarantee this property:

Definition 1 (from [8]). Let $\{x_1,\ldots,x_n\}$ be a set of variables. A *simple equation* E is an algebraic equation of the form $x_1 = f(x_2,\ldots,x_n)$ such that $f(x_2,\ldots,x_n)$ is an arithmetic expression over numerical constants and variables $x_2,\ldots,x_n$ where f uses the elementary algebraic operators '+', '-', '*','/' and contains each x_i exactly once.

[34] while we have to take care of division by zero, for details cf. [8]

Equations of this form can be easily transformed into an equivalent form $x_i = f'(x_1,\ldots,x_{i-1},x_{i+1},\ldots,x_n)$ for each appearing variable x_i, $2 \leq i \leq n$. For instance, in our example the equation can likewise be used to compute cd:population_female:

```
"?women_per_100_men * ?population_male / 100 = ?
     population_female"^^qbe:equationType
```

These equivalent transformations can be easily computed by standard mathematical libraries (which we will use in our implementation, cf. Section 6.3 below). A central piece of this transformation is a function *solve* with two parameters: the equation as string and the name of the target variable to solve for. The *solve* function algebraically solves an equation for a variable and returns a function. For example solve("a=b/c",c) would return the function "b/a" whereas solve("a=b/c",b) would return "a*c". The function *solve* is implemented in every computer algebra system – for example in Maxima with roots going back to the 1960s. That is, we could write

$$f'(x_1,\ldots,x_{i-1},x_{i+1},\ldots,x_n) = solve(x_1 = f(x_2,\ldots,x_n),x_i)$$

Analogously to qbe:equationType we define a datatype qbe:functionType describing an arithmetic function or expression whose lexical space is again defined via a SPARQL grammar non-terminal rule:

functionType ::= NumericExpression

Following the example for equationType a *function* for computing "Women per 100 men" is the following:

```
"population_female * 100 / ?population_male"^^qbe:functionType
```

QB rules (or functions) are similar to equations but can be evaluated only in one direction. Thus QB rules specify not (generic) variables but one or more input variables and exactly one output variable. These variables are specified in the same way as the variables in QBes, while the output variable does not need a qbe:variableName.

Example 6.3. A QB rule to convert the values for "Women per 100 men" to integer, using the function round to demonstrate a non-invertible function.

```
ex:qbrule1 a qbe:Rule ;
  qbe:input [
    qbe:filter [
      qbe:dimension cd:hasIndicator ;
      qbe:value cd:women_per_100_men ] ;
    qbe:variablename "?women_per_100_men"^^qbe:variableType ] ;
  qbe:output [
    qbe:filter [
      qbe:dimension cd:hasIndicator ;
      qbe:value cd:women_per_100_men_approx ] ] ;
  qbe:function "round(?women_per_100_men)"^^qbe:functionType.
```

6.2. QB Equation Semantics

We define the semantics of QBes by rewriting to a rule language. In fact SPARQL INSERT queries can be seen as rules over RDF triple stores where the pattern of the INSERT clause is the rule head and the graph pattern in the WHERE clause is the rule body. We note that this "idea" is not new and straightforwardly implementing the same concept as interpreting CONSTRUCT statements as rules, introduced e.g. in [29],

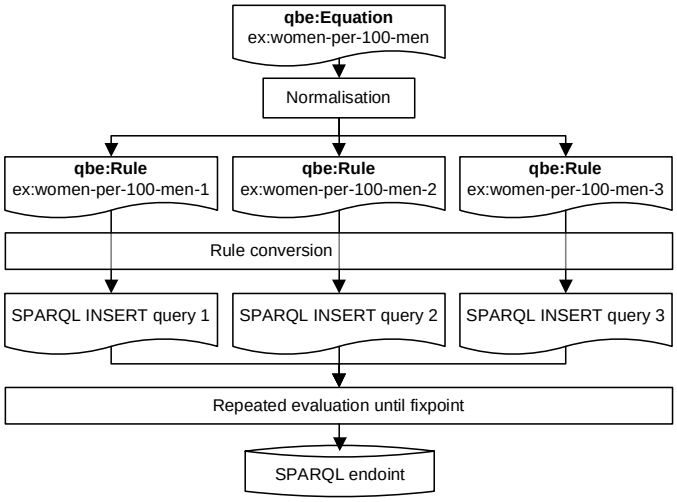

Figure 6: QB equation workflow with the example of a Eurostat indicator definition/equation

Listing 1: Algorithm to convert QB equations to QB rules

```
R := {}
for each (?e, ?eq) where { ?e rdf:type qbe:Equation.
                           ?e qbe:hasEquation ?eq }:
   for each (?v, ?vname) where { ?e qbe:variable ?v .
                                 ?v qbe:variableName ?vname }
       ?f := solve(?eq, ?vname)
       add r(?e, ?vname, ?f) to R
return R
```

Listing 2: Algorithm to create a QB rule

```
def r(?e, ?outvar, ?f):
rule := empty graph
?rulename := sk(?e, ?vname)
rule := { ?rulename a qbe:Rule .
          ?rulename qbe:hasFunction ?f .
          ?rulename prov:wasDerivedFrom ?e .
          ?rulename qbe:output ?outvar . }
for each (?v, ?vname) where  { ?e qbe:variable ?v .
                               ?v qbe:variableName ?vname }
    if ?vname != ?outvar:
        add { ?rulename qbe:input ?v } to rule
return rule
```

where we defined a formal semantics for such rules based on the Answer Set Semantics for non-monotonic Datalog programs (ASP) with external (built-in) predicates and aggregates [30]; builtin-predicates are introduced in SPARQL1.1 through expressions and assignment (BIND ... AS), and non-monotonicity in SPARQL is introduced by features such as OPTIONAL and NOT EXISTS.

We explain the semantics of QBes in three steps: (i) normalisation of QBes to QB rules (N QB rules generated from a QBe in N variables), (ii) conversion of QB rules (generated from QBes or as input) to SPARQL INSERT queries, (iii) a procedure to evaluate a program (a set of SPARQL INSERT queries) until a fixpoint is reached. See Figure 6 for an overview of the semantics with the example of the Eurostat indicator "Women per 100 men" in three variables used below. Note here, that in the general case rules with the expressive power of ASP with external predicates do not have a unique, finite fixpoint, but we will define/discuss how we can guarantee termination in our case.

6.2.1. Normalisation

A QBe in n variables can be viewed as a meta rule representing n rules: as discussed before, for each variable x_i a rule to compute x_i from all the other variables in the equation e can be generated by resolving the equation to $x_i = solve(e, x_i)$. To simplify the semantics specification we thus first normalise each QBe to n QB rules and then in the next step give the semantics for QB rules.

That is, the QB rules generated in the normalisation, have x_i as the output variable, and the other $(n-1)$ variables as input variables with $f'(x_1,\ldots,x_{i-1},x_{i+1},\ldots,x_n)$ being the function to compute the output.

Listing 1 shows the main algorithm of the conversion. The function `r` in Listing 2 takes three parameters: the original QBe URI, the name of the output variable, and the function. The function `sk(...)`, with a variable number of parameters is a Skolem function deterministically returning a unique URI for each unique parameter combination.

Eventually, after applying Listing 1 we could replace all QBes in an RDF graph with the QB rules in R, i.e. we see these representations equivalent. The remainder of our semantics only deals with rules.

Example 6.4. After normalisation the QBe of Example 6.2 results in three QB rules, one for each of the two variables. The QB rule to compute the Eurostat "Women per 100 men" indicator. Instead of variables we now have `input` and `output` and the equation was replaced by a function in the input variables.

```
ex:women-per-100-men-w a qbe:Rule ;
  prov:wasDerivedFrom ex:women-per-100-men ;
  qbe:input [
    qbe:filter [
      qb:dimension cd:hasIndicator ;
      qbe:value cd:population_male ] ;
    qbe:variablename "?population_male"^^qbe:variableType ] ;
  qbe:input [
    qbe:filter [
      qb:dimension cd:hasIndicator ;
      qbe:value cd:population_female ] ;
    qbe:variablename "?population_female"^^qbe:variableType ] ;
  qbe:output [
    qbe:filter [
      qb:dimension cd:hasIndicator ;
      qbe:value cd:women_per_100_men ] ] ;
  qbe:hasFunction "?population_female * 100 / ?population_male
       "^^qbe:functionType.
```

For each of the other two indicators `cd:population_female` and `cd:population_male` one QB rule is created analogously.

6.2.2. Rule Conversion

In this step QB rules are converted to SPARQL INSERT queries. The query has to implement several tasks: retrieve the input observations, compute the output observations, generate several URIs, perform error propagation and provenance tracking and ensure termination when evaluated repeatedly.

Compared to other rule languages SPARQL queries provide very complex features, but, as shown earlier [31, 32], can be compiled to –essentially– non-recursive Datalog with negation,

wherefore INSERT queries, read as rules, have the same expressivity.

Without loss of generality, we make the following assumptions (which could be easily checked in a pre-processing step, e.g., with a SPARQL ASK query assuring that there is a single measure value per observation):

- there is always only a single measure per observation
- the measure predicate that holds the measure value is fixed to `sdmx-measure:obsValue`

On a high level, the INSERT queries corresponding to QB rules have the following structure:

```
INSERT {
  output observation template
    - with PROV annotations to describe the generation by QBe
        rules
    - error estimation  }
WHERE {
  one pattern for each input observation
    - with all dimensions as specified in the DSD and error
        estimate

  BIND patterns for IRI creation for
    - ID for the newly generated observation
    - prov:Activity

  further BIND patterns to
    - assign current time to variable for PROV annotation
    - compute measure value of target observation
    - estimate error of target observation

  Termination condition  }
```

Output Observation. The output observation is set in the head with the fixed dimensions from the DSD and fixed dimension values if specified in the observation specification of the QB rule. The other dimension values are taken from the input variables. The rule head for Example 6.4 would look like the following query fragment, incorporating the PROV annotations:

```
?obs qb:dataSet globalcube:global-cube-ds ;
  cd:hasIndicator cd:women_per_100_men ;
  dcterms:date ?year ;
  sdmx-dimension:refArea ?city ;
  sdmx-measure:obsValue ?value ;
  prov:wasDerivedFrom ?population_male_obs, ?
       population_female_obs ;
  prov:wasGeneratedBy ?activity ;
  prov:generatedAtTime ?now ;
  cd:estimatedRMSE ?error .
```

It is important to note that the SPARQL INSERT application is *idempotent*, i.e., repeated applications of a generated SPARQL INSERT query will not add any more triples after the first application. Idempotence would be lost if blank nodes are used in the head, because they would create a fresh blank node for every application of a SPARQL query, even if the SPARQL query returns only a single result. Furthermore, we have to ensure that all values generated by the query are completely determined by the variable bindings of the WHERE clause.

Provenance Propagation. For every new derived observation we record the provenance, i.e., each derived observation has a link to each input observation `?obsin1, . . . , ?obsinN` and to the rule or equation used for the computation `?equation`. Firstly this provenance information provides transparency: We know precisely how a derived observation was computed. Secondly we use the provenance information during the derivation process to ensure termination. Furthermore, we record the time of the rule application and the agent, which could be the script or person responsible for the query creation.

```
?activity a prov:activity ;
  prov:qualifiedAssociation [
    a prov:Association ;
    prov:agent   cd:import.sh ;
    prov:hadPlan <http://citydata.wu.ac.at/ocdp/eurostat-rules#
         e4c56a2955372924bde20c2944b2b28f3> ] .
```

The preliminaries in Section 2 give an example of a part of the derivation tree generated by this rule head fragment.

Input Observations. For each input observation one set of triple patterns which asks for one observation is generated for the SPARQL WHERE clause. For each `qbe:DimSpecification` a dimension value is fixed. For all the other dimensions a fixed variable is used in all input observations. In the example below, again generated from Example 6.4 the query contains for all dimension values variables, except for `cd:hasIndicator` which is fixed to `cd:population_male` and `cd:population_female` as specified by the QB rule input dimension specification. Furthermore the observation value and the error estimated are retrieved.

```
?population_male_obs qb:dataSet globalcube:global-cube-ds;
  cd:hasIndicator cd:population_male ;
  dcterms:date ?year;
  sdmx-dimension:refArea ?city ;
  sdmx-measure:obsValue ?population_male ;
  cd:estimatedRMSE ?population_male_error .

?population_female_obs qb:dataSet globalcube:global-cube-ds;
  cd:hasIndicator cd:population_female ;
  dcterms:date ?year;
  sdmx-dimension:refArea ?city ;
  sdmx-measure:obsValue ?population_female ;
  cd:estimatedRMSE ?population_female_error .
```

Value Creation with `BIND`. Several SPARQL variables used for the output observation need to be computed using variables from the input observations. Most importantly the output measure value has to be created using the function of the QB rule.

```
BIND(100.0*?population_female/?population_male AS ?value)
```

Several URIs have to be generated for the rule head. We use a Skolem function to generate these URIs. The inputs of this Skolem function are the URI of the QB rule `rule`, the input variables `var1`, `...varN` and a string `"_static_"` to differentiate the different variables in the head. We implement this Skolem function with string concatenation and a hash function.

```
BIND(IRI(CONCAT(STR(rule) , MDA(CONCAT(STR(?var1), ..., STR(?
     varN))))) AS ?targetvar)
```

We have to generate two URIs: observation, and PROV activity.

```
BIND(CONCAT("http://citydata.wu.ac.at/ocdp/eurostat-rules#",
     MD5(CONCAT("http://citydata.wu.ac.at/ocdp/eurostat-rules
     #28f3",STR(?population_male_obs), STR(?
     population_female_obs))))) AS ?skolem)
BIND(IRI(CONCAT(?skolem, "_obs")) AS ?obs)
BIND(IRI(CONCAT(?skolem, "_activity")) AS ?activity)
```

Furthermore we bind the current time to a variable to use in the provenance part of the head.

```
BIND(NOW() as ?now)
```

Table 3: Computing the propagated estimated RMSE (*per*) for a given expression (*expr*)

expr	*per*(*expr*)
const	0
x_i	r_i
$a+b$	$per(a)+per(b)$
$a-b$	$per(a)+per(b)$
a/b	$(\|a\|+per(a))/(\|b\|-per(b))-a/b$
$a*b$	$(\|a\|+per(a))*(\|b\|+per(b))-a*b$

Error Propagation. Values computed based on values with an associated error also need an error estimate. The procedure to estimate an error of the new value is called *error propagatione* [33, 34].

In our use case we do not promise precise statistical error quantifications, but just want to propagate an upper bound of the error estimations of the inputs to the computed output value. We chose a error propagation function which is simple to implement in standard SPARQL. To this end, we incorporate a relatively naive error propagation function which however can be adapted to more accurate estimations if necessary in the future [33, 34].

We proceed herein as follows. The error values we have from our predictions are given as RMSE, i.e., the root-mean-square-error, which intuitively characterises how far off in absolute numbers the actual value is on average from our prediction. To compute a conservative estimate of how these errors "add up" when used in computations, we proceed as follows. Depending on the function f used for computing the computed output value, the n variables $x_1,\ldots x_n$ and their associated indicators $ind_1,\ldots ind_n$, we denote by $r_1,\ldots r_n$ the estimated RMSEs for these indicators, i.e. $r_i = RMSE(ind_i)$.

In Table 3 we define the propagated estimated RMSE (*per*) of a computed observation recursively over the operator tree of the function term $expr = f(x_1,\ldots x_n)$. Intuitively, we assume here the following: if the real values x'_i for indicators ind_i lie exactly r_i away –i.e., exactly the estimated RMSE above ($x'_i = x_i + r_i$) or below ($x'_i = x_i - r_i$) – from the predicted value x_i, we intend to estimate how much off would a value computed from these predicted values *maximally* be; here, *const* denotes a constant value, and a, b are sub-expressions. Furthermore we assume that the RMSE r_i is always less than the observed value x_i.

If now, for an equation $x_f = f(x_1,\ldots x_n)$, the propagated estimated RMSE $per(f(x_1,\ldots x_n))$ is smaller than the so far estimated RMSE r_f for indicator ind_f then we assume it potentially pays off to replace the predicted value so far with the newly computed value by the rule corresponding to the equation.

To cater for rounding errors during the computation we add a small ε of `0.0001` to the error estimate. In some sense this ε punishes each rule application and thus enables quicker termination later. Eventually the following BIND expression will be generated to compute the propagated error as defined by *per* and assign it to the corresponding variable used in the head of the rule.

```
BIND((ABS(100.0)+0.0)*(ABS(?population_female)+?
     population_female_error)*1.0/ (ABS(?population_male)-?
     population_male_error)-100.0*?population_female*1.0/?
     population_male + 0.0001 as ?error)
```

Termination. So far we introduced triple patterns and `BIND` expressions into the rule body. As remarked above the `BIND` expressions implement Skolem functions and thus avoid duplicating the same output observations over and over again (our SPARQL INSERT queries are idempotent). We now give two different termination conditions which can be used separately or together to ensure termination of the QB rules program.

To ensure termination of the whole SPARQL INSERT rule program we use a similar termination condition as in earlier work [8], we block the repeated application of the same rule to derive a particular observation. With the PROV annotations in fact we create an equation dependency graph. Given an observation o, a SPARQL path expression o `prov:wasDerivedFrom+` o' returns all the observations o' transitively used in the computation of o. Furthermore the SPARQL path expression o `prov:wasDerivedFrom*/prov:wasGeneratedBy/prov:qualifiedAssociation/prov:hadPlan` r gives all the rules r transitively used during the computation of o. So, in order to ensure termination, we define that a QB rule r is only *applicable* to materialise an observation o if r does not occur in the result of that path expression.

In the SPARQL INSERT query can we implement this condition by adding one of the following patterns for each input observation `?i` where `r` is the URI of the rule (or equation) itself.

```
FILTER NOT EXISTS {
?i prov:wasDerivedFrom*/prov:wasGeneratedBy/prov:
     qualifiedAssociation/prov:hadPlan/prov:wasDerivedFrom? r }
```

Thus as a worst case the evaluation will be terminated by this condition after applying each rule n times, where n is the number of QB rules in the system, because after applying each rule once for the derivation of a single observation no rule can be applicable anymore. An example of such a worst case would be a chain of QB rules where $r_i = r_{i+1}$ and $0 < i < n$ and a single given observation for r_0.

Another termination condition is based on the error propagation described above. Intuitively the condition ensures that an observation o from a computation is only materialised if no observation o' exists that (i) shares the same dimension values and (ii) has a lower or comparably low [35] error estimate.

```
?obsa qb:dataSet globalcube:global-cube-ds ;
     dcterms:date ?year;
     sdmx-dimension:refArea ?city ;
     cd:hasIndicator cd:women_per_100_men ;
     sdmx-dimension:sex ?sex ;
     estatwrap:unit ?unit ;
     sdmx-dimension:age ?age ;
     cd:estimatedRMSE ?errora .

     FILTER(?errora <= ?error * CT) }
```

Here, the constant factor `CT` is a value greater than or equal to 1, that determines a *confidence threshold* of how much improvement with respect to the estimated error is required to confidently

[35] That is, we add a *confidence threshold* that can be adapted, based on the confidence in the respective error propagation function, in order to only materialise new computed observations if we expect a *significant* improvement

"fire" the computation of a new observation. Thus, we materialise only observations that we expect to be significantly (i.e.,by a factor of CT) better with respect to error estimates. Since for any reasonable error propagation function the error estimates tend to increase with each rule application, consequently, together the two termination conditions can lead to faster termination.

For our naive error propagation function, $CT = 30.0$ turned out to be a reasonable choice, cf. the evaluation results in Section 7. Choosing $CT = 1$ would require an error propagation function with very high confidence, i.e., that never estimates a too low error, which we cannot guarantee for our naive estimation function.[36]

We note here that we really need *both* termination conditions, since relying on error estimates alone would need to ensure that the error propagation "converges" in the sense that application of rules does not decrease error rates. Our simple method for error propagation – in connection with *cyclic* rule application – does not guarantee this as demonstrated by the following, simple example:

Example 6.5. For two indicators i and j let two equations be $i = j/2$ and $j = i/2$. Essentially, this boils down to the (cyclic) equation $i = i/4$ where – in each application – we would derive smaller error estimate.

While this example is quite obviously incoherent (in the sense of rules being cyclic in terms of the rule dependency graph defined in [8][Definition 12]), we still see that with cyclic application of rules the convergence of error rates cannot be ensured in general. In practice such incoherent systems of equations are hardly useful, however the first termination condition would still serve its purpose.

Example 6.6. Taking the observations in lines 1 and 2 of Table 4 we can compute the "No. of bed-places in tourist accommodation establishments" for Bolzano 2010 as 2434.5 with an RMSE (computed with the propagated error function *per*) of 56.6 (line 3). The QBe observation of line 3 is classified as "better" than the best predicted observation (line 4) because of the RMSE comparison with respect to the confidence threshold: $56.6 \cdot 30 < 3228.8$.

Similarly, the observations from line 5 and 6 are used by the QB equation of the running example to compute the observation in line 7. Since there exists already an observation from the crawl with a better RMSE (line 8), the computed QBe observation will be ignored in this case.

6.3. Implementation of the Combination of Statistical Predictions and QB Equations

Herein, we describe in more detail how we have combined statistical inferences from Section 5 with the application of QB equations in our implementation, following the general workflow outlined above in Section 3. We first explain in a bit more detail how we implement the application of QB equation in our system, whereafter we explain the overall workflow.

[36] Note that this has also been the reason why we introduced the factor CT, as the earlier simpler condition `FILTER((?errora <= ?error)` produced too many over-optimistic – and in fact worse – observations.

Table 4: Example data for Bolzano in the year 2010

Source	Indicator	Value	Error
Crawl (UN)	Population	103582.0	0.0
Prediction	No. of available beds per 100 residents	23.5	0.55
QBe	No. of bed-places in tourist accomm. est.	2434.5	56.6
Prediction	No. of bed-places in tourist accomm. est.	1490.5	3228.8
Crawl (UN)	Population male	49570.0	0.0
Prediction	Population female	54836.2	7044.0
QBe	Women per 100 men	110.6	14.3
Crawl	Women per 100 men	109.0	0.0

6.3.1. Implementation of QB Equations

As described in Section 6.2 we compile QBes into a semantically equivalent set of rules. Usually there are two strategies for query answering in rule based knowledge bases: forward or backward chaining. For the OCDP we decided to implement a forward chaining approach to enrich the global data cube with the newly inferred observations. Forward chaining approaches materialise as much as possible thus allowing faster query evaluation times. On the other hand forward chaining approaches require more memory or disk space and updates lead to re-materialisation. In our case the space requirements are manageable and updates are not frequent.

Our forward chaining implementation approach relies on the iterative application of SPARQL INSERT queries (which implement the rules). Overall, QBes infer and persist new observations in three steps: (Normalisation) convert all QBes to QB rules, (Rule conversion) for each QB rule we create a SPARQL query, and (Query evaluation) iteratively evaluate the constructed SPARQL INSERT queries until a fixpoint is reached (that is, no better observations can be derived).

Normalisation. As described in the semantics above in this first step we convert QBes to QB rules. The algorithm in Listing 1 already outlines our implementation. We implemented the algorithm using Python 2.7 and the libraries `rdflib` for RDF/SPARQL processing and `sympy` providing the *solve* function to algebraically solve an equation for a variable. The Python script reads the QBes from an RDF file containing the QBes,[37] converts them to QB rules and publishes them again as Linked Data[38] and in the triple store.

Creating SPARQL Queries. We create a SPARQL INSERT query for each QB rule. The listing in AppendixA gives as a complete example of the SPARQL INSERT query resulting from converting one of the QB rules. Due to a serious performance problem of SPARQL INSERT queries applied on the Virtuoso triple store in a preliminary evaluation, we used SPARQL CONSTRUCT queries instead, and load the resulting triples into the triple store afterwards. The conversion from QB rules to SPARQL CONSTRUCT queries is analogous to the algorithm described in Section 6.2.2 above to convert a QB rule to a SPARQL INSERT query.

[37] http://citydata.wu.ac.at/ocdp/eurostat-equations.rdf
[38] http://citydata.wu.ac.at/ocdp/eurostat-rules.rdf

Evaluating Queries in Rule Engines. In principle the rule engine naively evaluates SPARQL INSERT queries, or respectively, CONSTRUCT queries + re-loads, over and over again until a fixpoint is reached.

Apart from the termination conditions described in Section 6.2.2 we ensure that the repeated application of a rule on the same observations does not create any new triples by using Skolem constants instead of blank nodes (see also discussion on idempotency above). Thus, in order to check whether a fixpoint has been reached, it is enough to check in each iteration simply if the overall number of triples has changed or not. So, for a naive implementation we could simply use a SPARQL query to count the number of triples in the named graph.

However, unfortunately, in our experiments, such a simple implementation solely based on "onboard" means of the SPARQL engines turned out to be infeasible due to performance reasons. Thus, for the time being, we resorted to just evaluating one iteration of all generated rules, in order to evaluate our conjecture that rules improve our prediction results.

Eventually, we may need to resort to (offline) using a native rule engine. Indeed, in practical applications such rule/datalog engines have shown to perform better than recursive views implemented directly on top of databases in the past for instance for computing RDFS closure, cf. [35]. For the moment, we leave this to future work and resort, as mentioned, to a fixed number of iterations of rule applications.

6.3.2. Implementation of the Combined Enrichment Workflow

There are various possible combinations of QB equations and statistical inferences conceivable. Based on our experiments (and we will further argue these choices the details evaluation of the performance of our approach in Section 7 below, we have decided for the following implementation. Here, we follow the workflow described in Figure 1 and Section 3.2 above. That is, we proceed in three steps as follows:

1. *Materialisation of observations by application of* QB equations *on the raw data*: in a first step we load the integrated and linked observations from the data integration step (4). Note here that each observation, in order to be considered in our rules needs an `cd:estimatedRMSE`, which per default is set to 0 for factual observations. However, note that due to the linking of different data sources, we could potentially have several ambiguous observations for the same city, year and indicator in this raw data already, e.g. two or more different population values from different data wrappers materialised in the global cube. Let us say, we have for city `C1` in the year 2017 three different `population` values in three different factual observations from UNdata, Eurostat and dbpedia, i.e. obs_{UN}: `1,000,000`, $obs_{EuroStat}$: `980,000`, and $obs_{Dbpedia}$: `1,020,000`. In principle, we take no preference among sources, so we proceed as follows in a preprocessing step: for such case we set the `cd:estimatedRMSE` to the difference from the average of all available ambiguous values (for the same indicator and city/year pair). That is, we set:

```
:obsUN estimatedRMSE  0.
:obsEurostat estimatedRMSE 20000.
:obsDbpedia estimatedRMSE 20000.
```

   Aftrer this preprocessing, we apply a first application of QB equations, in which case – obviously – the value from obs_{UN} would be preferred for any equation having `population` as input indicator.
2. *Materialisation of observations by* statistical missing-value prediction: as a second step, we apply enrichment by *statistical regression methods* and computation of estimated RMSEs per indicator as described in Section 5; these statistical regression methods can potentially benefit already from derived additional factual knowledge from the prior step.
3. *Materialisation of further observations by re-application of* QB equations: finally, using these predictions, we iteratively re-apply *QB equations* wherever we expect (through error propagation) an improvement over the current RMSE by using computed values rather than statistical predictions only.

We remark that one could imagine alternatively to re-iterate steps 3. and 2. as well, i.e., by re-computing statistical models from step 2. based on the application of equations in step 3. again, and so on. However, for instance due to impreciseness of error estimations alone, this would be extremely prone to overfitting and – as expected – rather showed a quality decrease than improvement in some preliminary experiments we ran.

7. Evaluation

In this section, we summarise experiments we conducted to evaluate both the performance of our crawls as well as the quality of enrichment of our pipeline.

The OCDP runs distributed over several components. The UNData wrapper is a server component that runs at WU Vienna and the Eurostat wrapper is also a server component that runs in a cloud environment. The crawler and rule engine is a client component that dereferences the seed URIs, follows links, and performs the reasoning that lead to the unified global cube. The resulting files are then inserted into the SPARQL endpoint. From that point on, all further enrichment is carried out over the combined global cube via the SPARQL endpoint.

In Section 7.1 we first describe in more detail the process that leads to the global cube accessible via the SPARQL endpoint. We then cover in Section 7.2 the enrichment process, consisting of statistical missing value prediction and of calculating the values based on QB equations. The missing value prediction is performed asynchronously regularly on a workstation, following the regular crawls of the crawler.

7.1. Crawling and Global Cube Materialisation

The machine that runs the crawling and rule engine Linked Data-Fu is equipped with two eight-core Intel(R) Xeon(R) E5-2670 CPUs @ 2.60GHz and 256 GB of main memory. Crawling and integration runs separately for Eurostat and UNData. The result is one N-Quads file containing the Eurostat portion of

the global cube and one N-Quads file containing the UNData portion of the global cube. We separately run the "rapper" RDF parser over the files to ensure that subsequent SPARQL loading steps do not fail with syntax errors.

The crawling and rule application requires around 6 minutes for Eurostat and around 24 minutes for UNData. For the 1,666,379 observations of Eurostat, 473 RDF documents containing 10,751,759 triples are dereferenced (567 MB). The rule application derives 1,666,619 triples. For the 128,693 observations of UNData, 4,505 RDF documents containing 1,690,231 triples are dereferenced (152 MB). The rule application derives 1,002,135 triples. While accessing UNData yields less triples than accessing Eurostat, the UNData data is distributed over many more files and requires more HTTP requests[39]. Thus, the UNData access takes much longer than the Eurostat access.

Loading the global cube into the Virtuoso SPARQL endpoint requires around 190 seconds. Filtering and skolemisation takes 20 minutes.

7.2. Combining Statistical Missing-Values-Prediction and QB Equations

In Section 5 we have reported on evaluation of the missing values prediction in detail. Recall that we perform PCA to reduce the number of dimensions, which allows us to impute all the missing values with neutral values for the PCA and then evaluate the quality of the predictions using different (the respectively best one per indicator) base statistical regression methods.

As for runtime performance, our current statistical prediction runs need on a Ubuntu Linux server with 2 cores and 2.6 GHz approximately 35 hours for all indicators and testing all base methods. The run time might slightly grow with the number of indicators, hence we aim to optimise the predictions runs by using our model provenance information, and evaluate only the best base method, which should reduce the runtime by factor three.

As mentioned in Section 5, we have identified two main goals for filling in missing value:

1. It is important to build models which are able to predict many (preferably all) missing values.
2. Second, the prediction accuracy of the models is essential, so that the Open City Data Pipeline can fulfil its purpose of publishing high-quality, accurate data and predictions.

Prediction accuracy in our approach is a median 0.55%RMSE over all indicators for the years 2004–2017, which allows us to predict new 608,848 values on top of the existing 693 684. Recall that despite the use of PCA, this difference occurs, since we drop too sparse rows/columns in the data matrix before PCA, in order to accept at an acceptably low overall median %RMSE, so we cannot predict anything for very sparse areas of the data matrix. Still, while we already discussed in Section 5 that the accuracy has improved considerably since our prior work in [7], however, as mentioned beforehand, our main goal was to improve these predictions further by the combination with QB equations, i.e. to both improve the quality of predictions and enable to predict more missing values overall.

So, we will next focus on evaluation presenting some numbers on the considered QB equations themselves and their evaluation performance, and then report on the correctness of and improvements by the combination of QB equations with statistical regression methods.

Section 6.3 described the implementation of QB equations in the OCDP. In this section we give some results about the behaviour of the QB equations part of the OCDP system[40] and some evaluation of the QB evaluation themselves and how they improve the results of the whole OCDP.

Normalisation. The normalisation to generate QB rules from QB equations took 25 seconds to normalise 61 QB equations from Eurostat into 267 QB rules[41].

AppendixA contains a complete example QB equation from Eurostat and one of the normalised QB rules.

Creating SPARQL Queries. First we filter out 76 QB rules for which at least one input variable matches no existing observation. Such rules can never deliver any results and evaluating them is thus needless. Virtuoso could generally not evaluate 44 QB rules which contain seven or more input variables. Eventually we created 147 SPARQL CONSTRUCT queries in five seconds.

AppendixA shows a complete example of a SPARQL CONSTRUCT query together with the corresponding QB rule.

Evaluating Queries in Rule Engines. This one iteration of evaluating all generated 147 SPARQL CONSTRUCT queries took 28 minutes (time-outs for 12 queries) and inserted 1.8M observations (46M triples) into the global cube.

7.3. Combining QB Equations with Statistical Methods

From the different data sources the OCDP crawler collects 991k observations. The statistical missing-values prediction return 522k observations better than any QB equation observation (if existing). The QB equations return 230k observations better than any other prediction or QB equation observation (if existing); additionally, 232k new observations computed by QB equations were actually not predictable at all (due to bad quality) with the statistical regression methods. Eventually the whole OCDP dataset contains 1975k observations.

Apart from these overall numbers, we provide in the following subsections more details on particular aspects on the correctness of and improvements by the combination of statistical regression methods and QB equations for predicting missing values.

[39] We wait 500 ms between requests to not overload the server providing data.

[40] for more details see http://citydata.wu.ac.at/ocdp/import

[41] the Eurostat indicator definition for the population change over 1 year is the only indicator not expressible in QB equations

7.4. Correctness of Generated QB Observations

Firstly, to show the correctness of the QB equation approach on raw data (first step of the workflow described in Section 6.3.2, we compared the observations derivable by QB equations only from crawled observations with the corresponding crawled observations. We made the following observations: in the sources we currently consider, equations had already been applied in the raw data before import, and thus applying them beforehand in Step 1 of the workflow in Section 6.3.2 did not have a notable effect. This can mainly be explained by the fact that our considered equations stem from Eurostat's indicator definitions, and therefore from within *one* dataset, where they are already pre-computed. That is, in the cases of consistent input observations (no more than one observation per city-year-indicator combination) the QBes computed consistent results with respect to the crawl.

Notably, however, for the cases of inconsistent/ambiguous input observations in the raw data, the QB equations also possibly compute ambiguous resulting observations. In fact, we discovered 48643 such cases of inconsistent/ambiguous observations, that is, multiple observations per city-year-indicator combination. While again, as described in Section 6.3.2, Step 1, we do not resolve these inconsistencies in the raw data, we "punish" them in the computation by assigning inconsistent input observations with an estimated RMSE corresponding to the deviation from the average above all the inconsistent observations for a single city-year-indicator combination.

We note that using QB equations could also be used to aid consistency checking, for instance our experiments unveiled also a missing factor in one of the Eurostat indicator definitions[42] as well as wrongly mapped cities and indicators during the development process.

In general, it makes sense to evaluate the QB equations based on the crawled data and thus enrich the crawled dataset to achieve better results in the following application of statistical regression methods. However, in our experiments which focused in the UN and Eurostat datasets, the prior application had only marginal effects: in our case almost half of the new observations (10178 of 26452) that could be generated in this way were for the indicator "Women per 100 men", because this is the only Eurostat indicator for which the UN dataset contained both necessary base indicators (population male and population female); the other cases could again be traced back to inconsistencies in the source data.

7.5. Quality Increase

To test the quality increase of the combined method we tested which ones were the best observations, comparing statistically predicted observations, with QB-equation-generated observation depending on the estimated RMSE associated with each observation, with real factual observations. As described in Section 6 a QB equation observation is only computed if the estimated RMSE multiplied by the confidence threshold (CT) is smaller than the estimated RMSE of any other corresponding observation. Through experimentation during the development of the OCDP we found a confidence threshold of 30 being a good compromise between data quality without sacrificing too many good observations. We got the same or a better RMSEs for 80 of the 82 tested indicators: these 82 indicators are those for which overlapping indicators from the statistical predictions and QB equations were available together with actual observed values from the crawl.

We have summarised the results of this analysis in Table 5 for detailed RMSEs for the predicted and the QB equation observations of all 82 tested indicators. For each indicator, the tables lists in the first three columns the numbers of crawled, predicted and QBe computed predictions. The next three columns list the accuracy in terms of actual RMSE (i.e. not estimated, but comparing real existing observations with values generated through predictions or through QBequations): we see here that, in combination the two methods performed better or equal than statistical predictions alone in most cases (indicated in bold face values in the "Combined" column), i.e., we got as mentioned above the same or better RMSE for 80 out of the tested 82 indicators.

Finally, an important property of the combined method is how precise/accurate the propagated error computation is, since this propagated estimated error (RMSE in our case) is used to decide which observation is classified as the best observation for a given city-year-indicator combination. We thus model this as a binary classification task: for a fixed city-year-indicator combination, given the corresponding prediction observation and the best observation generated by a QB equation, both with error estimates, is the QB equation observation value nearer to the actual value than the predication observation value? In this case we are more interested in a minimal number of false positives (QB equation observation wrongly selected as better) even at the cost of a higher number of false negatives (better QB equation observation wrongly not selected as better). Of the usual measures to quantify the quality of classification tasks we thus are mainly interested in precision. We get an average precision of 90.8% for a confidence threshold of 30, while we miss quite some true positives (significantly lower accuracy). See Table 5 for detailed results ("precision" and "accuracy") of all 82 indicators.

As we can demonstrate in even such an incomplete materialisation of equations allows us to predict a significant amount of new or improved observations, that could not be predicted with equal accuracy solely with the methods described in Section 5.

8. Related Work

In the following, we explain how our work distinguishes itself from the related work in the areas of modelling, integrating and querying of numerical data using web technologies, of predicting/imputing of missing values, and of using declarative knowledge for inferencing of numeric information.

The work of [36] describes a methodology to describe city data for automatic visualisation of indicators in a dashboard.

[42] the indicator "Women per 100 men – aged 75 years and over" is missing a factor 100

Table 5: Evaluation results 82 indicators (for which crawled, predicted, and QBe observations existed). The "Observation source" lists how many chosen best observations which of the three sources contributed. The "RMSE" columns give the RMSEs of Predictions and QBes as well as the combined system. The quality measures, especially the precision, give an indication how well the error propagation classified better observations. In the cases marked with a * no improvements were observed by the QBes, i.e., the statistical predictions were better than any possible QBe.

	Observation source			RMSE			Quality measures	
Indicator	Crawled	Prediction	QBe	Prediction	QBe	Combined	Precision	Accuracy
average size of households	3463	3194	0	0.04	0.08	**0.04**	*	0.63
crude birth rate per 1000 inhabitants	6739	1079	194	10.19	0.38	**10.19**	*	0.36
crude death rate per 1000 inhabitants	6417	1273	59	9.30	0.28	**9.3**	*	0.31
economically active population female	4517	3025	104	3083.12	9636.50	**3083.12**	*	0.72
economically active population male	4520	3025	104	2887.80	12171.77	**2887.8**	*	0.78
economically active population total	4750	2765	108	4581.38	6596.52	**4581.38**	*	0.49
employment jobs in agriculture fishery nace rev 2 a	3244	3003	729	231.26	1044.50	**228.71**	1.00	0.45
employment jobs in construction nace rev 2 f	3447	2294	1317	775.68	1378.20	**775.62**	1.00	0.24
employment jobs in mining manufacturing energy nace rev 2 b-e	3442	2511	1105	2042.92	6692.59	**2042.89**	1.00	0.40
eu foreigners	4268	774	2341	2777.38	597.50	**2746.81**	0.91	0.29
eu foreigners as a proportion of population	4209	2840	306	0.65	0.42	**0.65**	*	0.25
foreign-born	2149	246	3961	18408.69	3095.48	**18305.7**	0.98	0.19
foreign-born as a proportion of population	2136	4074	134	2.94	1.52	**2.94**	*	0.43
foreigners	3290	389	2748	13187.75	1394.98	**13176.48**	0.98	0.22
foreigners as a proportion of population	3261	2914	238	1.65	0.89	**1.65**	*	0.38
households owning their own dwelling	2624	1449	3036	3998.24	50431.29	**3998.24**	*	0.26
households with children aged 0 to under 18	4023	967	2341	1952.82	6703.39	**1952.77**	1.00	0.14
infant mortality per year	5214	1383	1062	1.61	7.98	**1.61**	*	0.28
infant mortality rate per 1000 live births	5083	2046	418	0.58	1.01	**0.58**	*	0.42
lone parent households per 100 househ. with children aged 0-17	3037	4042	107	0.73	3.33	**0.73**	*	0.78
lone parent private households with children aged 0 to under 18	3180	282	3774	674.46	341.52	**674.46**	1.00	0.13
lone pensioner above retirement age households	3609	2849	766	88.58	4590.84	**88.58**	*	0.26
nationals	5569	1122	1056	73639.26	12052.11	**73355.72**	1.00	0.22
nationals as a proportion of population	5531	1733	457	116.24	2.92	**116.24**	0.94	0.33
native-born	2159	350	3845	99075.25	8725.38	**99075.25**	*	0.18
native-born as a proportion of population	2146	3999	197	23.54	2.84	**23.54**	*	0.58
no available beds per 1000 residents	3836	3437	30	57.56	397.42	**57.56**	*	0.63
no bed-places in tourist accomm. establishments	4226	3	3443	166966.67	3349.37	**166948.07**	0.98	0.67
no children 0-4 in day care or school	4051	3113	503	579.63	522.03	**579.62**	0.91	0.47
no children 0-4 in day care publ and priv per 1000 children 0-4	3323	3556	93	20.92	610.03	**20.91**	1.00	0.74
no cinema seats per 1000 residents	2283	3027	1336	33.37	1.82	**33.37**	1.00	0.15
no cinema seats total capacity	2660	2035	2278	822.75	633.30	**798.6**	0.99	0.18
no deaths in road accidents	5574	668	1044	2.42	1.86	**2.42**	*	0.20
no households living in apartments	2115	1261	3369	6103.92	32588.96	**6103.92**	*	0.27
no households living in houses	2153	2027	2542	10502.07	24758.61	**10498.78**	0.88	0.26
no live births per year	6974	231	987	476.76	156.75	489.98	0.08	0.23
no private cars registered	4693	856	1814	43201.98	4490.18	**43047.54**	0.83	0.25
no registered cars per 1000 population	4549	2264	464	562.66	18.06	**562.59**	0.90	0.31
no tourist overnight stays in reg accomm. per year per resident	4821	2674	54	13.10	0.79	**13.1**	*	0.14
non-eu foreigners	4250	377	2730	7884.07	1085.92	**7858**	0.97	0.23
non-eu foreigners as a proportion of population	4191	2902	236	1.40	0.35	**1.4**	*	0.31
one person households	4234	231	2982	3901.77	14048.54	**3900.09**	1.00	0.14
people killed in road accidents per 10000 pop	5172	1765	37	0.33	0.02	**0.33**	*	0.19
persons unemployed female	5741	2105	7	734.71	387.00	**734.71**	*	0.28
persons unemployed male	5798	2054	7	808.36	342.44	**807.88**	1.00	0.22
persons unemployed total	5262	2402	14	450.51	523.96	**450.51**	*	0.51
population	17058	50	99081	746087.75	746119.20	746256.3	0.19	0.34
population female	14181	580	4380	76682.77	75971.54	**76682.77**	*	0.50
population living in private househ. excl. institutional househ.	3602	2498	614	55048.78	25328.24	**55048.78**	*	0.25
population male	14183	4838	122	71411.92	71769.93	**71411.92**	*	0.48
population on the 1st of january 10-14 years total	4914	272	1546	3831.38	656.61	**3831.38**	*	0.23
population on the 1st of january 25-34 years total	6416	276	666	13231.20	1064.67	**13214.86**	1.00	0.21
population on the 1st of january 35-44 years total	6461	194	724	7044.37	1020.55	**7018.79**	1.00	0.21
population on the 1st of january 45-54 years total	6435	366	551	6395.69	838.57	**6352.97**	1.00	0.20
population on the 1st of january 5-9 years total	4866	211	1629	4084.84	717.81	**4084.84**	*	0.22
population on the 1st of january 55-64 years total	8379	134	140	5172.41	808.54	**5091.16**	1.00	0.27
population on the 1st of january 65-74 years total	8401	124	123	4870.38	590.66	**4837.58**	1.00	0.25
population on the 1st of january 75 years and over female	6860	1140	*	471071.80	22798.19	**114272.33**	0.72	0.71
population on the 1st of january 75 years and over male	6889	1129	*	12645.78	279859.81	**12645.78**	*	0.72
population on the 1st of january 75 years and over total	8413	55	195	5594.77	539.03	**5577.07**	1.00	0.20
private households excl. institutional households	5130	3	2468	6759.09	51025.85	**6672.43**	0.96	0.37
proportion households that are lone-pensioner households	3508	3700	*	0.08	0.12	**0.08**	*	0.48
proportion of employment in agriculture fishery	3172	3525	253	0.26	25.71	**0.26**	*	0.55
proportion of employment in construction nace rev11 f	3386	3550	98	0.35	7.71	**0.35**	*	0.75
proportion of employment in industries nace rev11 c-e	3384	3325	325	1.66	9.79	**1.66**	*	0.60
proportion of households living in apartments	1867	4494	265	1.97	4.67	**1.97**	*	0.67
proportion of households living in houses	1929	4214	482	1.80	3.08	**1.8**	*	0.55
proportion of households living in owned dwellings	2299	4308	333	1.75	21.97	**1.75**	*	0.67
proportion of households that are 1-person households	4128	3254	51	0.87	3.06	**0.87**	*	0.77
proportion of households that are lone-parent households	3065	4088	66	0.18	0.57	**0.18**	*	0.68
proportion of households with children aged 0-17	3917	3345	54	0.52	2.20	**0.52**	*	0.79
proportion of population aged 0-4 years	8328	313	0	0.08	1.00	**0.08**	*	0.91
proportion of population aged 10-14 years	4893	1817	8	1.16	0.13	**1.16**	*	0.62
proportion of population aged 15-19 years	8341	272	0	0.10	6.72	**0.1**	*	0.94
proportion of population aged 20-24 years	8326	259	26	0.15	9.05	**0.15**	*	0.95
proportion of population aged 25-34 years	6337	771	190	10.25	0.47	**10.25**	*	0.35
proportion of population aged 35-44 years	6382	781	156	9.27	0.50	**9.27**	*	0.35
proportion of population aged 45-54 years	6356	672	264	10.43	0.48	**10.43**	*	0.24
proportion of population aged 5-9 years	4845	1847	0	1.15	0.14	**1.15**	*	0.65
proportion of population aged 65-74 years	8384	183	72	8.86	0.29	**8.86**	*	0.32
proportion of population aged 75 years and over	8396	198	60	6.71	0.25	**6.71**	*	0.36
proportion of total population aged 55-64	8362	208	74	13.61	0.40	**13.61**	*	0.28

The work is different from our work in several regards. Our work directly reuses W3C standardised vocabularies PROV and QB as well as common Linked Data wrappers and crawlers and as such is more widely applicable. Our work makes use of a pre-existing carefully handcrafted list of common statistical indicators (city data ontology) that was mainly inspired from the Eurostat urban audit but also takes into account other city data sources; the ISO 37120:2014 used by [36] only lists 100 roughly defined indicators whereas urban audit uses over 200 indicators with available numbers and for some of these indicators also provides computation formulas. The work of [36] focusses on automatic selection of suitable data for indicators using Prolog inferences and automatically selecting the right visualisation; this was demonstrated with only one indicator bicycle "trips per station". Our work instead focusses on the combination of declarative knowledge and machine learning for deriving/predicting new values from integrated datasets and for that presents a widely-applicable data integration, enrichment and publication pipeline evaluated on a set of more than 200 indicators.

8.1. Numerical Data in Databases

Siegel et al. [37] introduce the notion of semantic values – numeric values accompanied by metadata for interpreting the value, e.g., the unit – and propose conversion functions to facilitate the exchange of distributed datasets by heterogeneous information systems.

Diamantini et al. [38] suggest to uniquely define indicators (measures) as formulas, aggregation functions, semantics (mathematical meaning) of the formula, and recursive references to other indicators. They use mathematical standards for describing the semantics of operations (MathML, OpenMath) and use Prolog to reason about indicators, e.g., for equality or consistency of indicators. In contrast, we focus on heterogeneities occurring in terms of dimensions and members, and allow conversions and combinations.

As a basis for sharing and integration of numerical data, XML is often used [39]. XML standards such as XCube fulfil requirements for sharing of data cubes [40] such as the conceptual model of data cubes, the distinction of data (observations) and metadata (dimensions, measures), a network-transportable data format, support for linking and inclusion concepts, extensibility, conversion capability and OLAP query functionality. Other advantages include that XML allows to define a schema (XML Schema), there are data modification and query languages for XML such as XSLT and XQuery, and there are widely-used XML schemas for representing specific information, e.g., XBRL for financial reports, SDMX for statistics, DDI[43] for research studies. Another XML-based exchange standard for ETL transformations and data warehouse metadata is the Common Warehouse Metamodel (CWM)[44] by the Object Management Group (OMG).

However, the integration of data across different standards is still an open issue. CWM – but also other interfaces and protocols to share multidimensional datasets such as XML for Analysis and OLE DB – lack a formal definition making it more difficult to use such formalism as a basis for integration [41]. XML schemas are concerned with defining a syntactically valid XML document representing some specific type of information. Yet, XML schemas do not describe domain models; without formal domain models, it is difficult to derive semantic relationships between elements from different XML schemas [42]. Often, the domain model for an XML schema is represented in a semi-formal way using UML documents and free text. In contrast, schemas described as an OWL or RDFS ontology such as QB have a formal domain model based on logics.

Conceptually, we distinguish the global-as-view (GAV, also known as source-based integration) approach of data integration where the global schema is represented in terms of the data sources and the local-as-view (LAV) approach that requires sources to be defined as views over the global schema [43–45]. We use the GAV approach and define the global cube in terms of single data cubes using the drill-across operation. With GAV, queries over the global schema can easily be translated to queries over the data sources [44]. The advantage of LAV is that the global schema does not need to change with the addition of new data sources. The advantage of GAV is that queries over the global schema can easily be translated to queries over the data sources.

8.2. RDF Data Pipelines

Within the Semantic Web community there is extensive work around triplification and building data pipelines and Linked Data wrappers for publicly available data sources on the web, where for instance the LOD2 project has created and promoted a whole stack of tools to support the life cycle of Linked Data, i.e. creating maintainable and sustainable mappings/wrappers of existing data sources to RDF and Linked Data, a good overview is provided in the book chapter by Auer et al. [46, 47]. All this work could likewise be viewed as an application of the classical ETL (Extract-Transform-Load) [48] methodology extended to work on the web, based on open standards and Linked Data principles [13]. Our work is not much different in this respect, with the difference that we apply a tailored architecture for a set of selected sources around a focused topic (city data), where we believe that a bespoke combination of rule-based reasoning methods in combination with statistical machine learning can provide added value in terms of data enrichment. This is a key difference to the above-mentioned methods that rather focus on entity linkage and object consolidation in terms of semantic enrichment. However, this focused approach is also different from generic methods for reasoning over Linked Data on the web (cf. e.g. [49] and references therein for an overview), solely based on OWL and RDFs which (except very basic application of `owl:sameAs` (for consolidating different city identifiers across sources) and `rdfs:subPropertyOf` reasoning (for combining overlapping base indicators occurring within different sources).

Other work tries to automatically derive new from existing data. Ambite and Kapoor [50] present Shim Services providing operations for accessing remote data, integrating heterogeneous

[43] http://www.ddialliance.org/
[44] http://www.omg.org/spec/CWM/

data, and deriving new data. Workflows of operations are automatically created based on semantic descriptions of operators. Subsumption reasoning is included to match inputs of services to outputs of other services. To avoid the infinite execution of operations, a limit is defined to the depth of nested operations of the same type. In contrast to the automatically constructed workflows, our pipeline consists of a fixed set of processing steps. Instead of "shim services" that act as stand-alone components accessible via the network, we base the computation on local formulas and use a vocabulary to represent the formulas.

8.3. Data Modelling and Representation

Besides the RDF Data Cube Vocabulary (QB) that we are using in this work there are other vocabularies available to publish raw or aggregated multidimensional datasets. For instance, there are various OWL ontologies available for representing multidimensional datasets [51]. Also, several light-weight ontologies have been proposed, such as SCOVO [52] and SCOVOLink [53]. Other vocabularies for statistical data are the DDI RDF Vocabularies[45], several vocabularies inspired by the Data Documentation Initiative, and the StatDCAT application profile[46] (StatDCAT-AP) to express in a structured way the metadata of statistical datasets which are currently published by the different agencies in the European Union. In comparison to these approaches, we see the following reasons for choosing QB:

QB, as a W3C recommendation, is an established standard for aggregating and (re-)publishing statistical observations on the web, with off-the-shelf tools to process and visualise QB data. QB's wide adoption is an important factor for data integration use cases, as sources already represented in QB can be integrated more easily than sources in other representations. Further, QB has shown applicability in various use cases [54], and exhibits the necessary formality to allow efficient and flexible integration and analysis of statistical datasets. The multidimensional data model of QB allows to make explicit different dimension – dimension value combinations, e.g., `_:obs cd:unit "km2".` and `_:obs dcterms:date "2010".` which is important for interpreting the semantics of values and for integration purposes [53].

8.4. Modelling of Equations

Modelling the actual numerical data and the structure of that data captures only a part of the knowledge around statistical data that can be represented in a machine-interpretable manner. Equations in particular are a rich source of knowledge in statistical data. Lange [55] gives an extensive overview of representations of mathematical knowledge for the Semantic Web. We first cover representation of equations for layout purposes, and then cover representations that permit the interpretation of the formulas by machines.

Non-RDF based representations of mathematical knowledge include MathML [56] and OpenMath [57] and use XML for serialisation and focus more on a semantic representation of mathematical entities and are not directly useful for reasoning. Although GeoSPARQL [58] uses MathML in XMLLiterals and OpenMath provides preliminary integration into RDF,[47] these representations are hard to reuse for RDF tools and still are not suitable for an RDF QB setup.

The OWL ontologies QUDT [28], OM [59], and SWEET [60] provide means to describe units and to some extent model conversion between these units, but do not specify a concrete machinery to perform these conversions. Our approach is orthogonal to these efforts in that it provides not only a modelling tool for unit conversions but more general equations and also gives a semantics to automatically infer new values.

Semantic Web rule languages and systems often implement numerical functions – for example RIF uses numerical functions from XPath [61]. Other examples for rule languages and systems include SWRL and Apache Jena rules. Converting equations to rules naively can lead to a set of recursive rules which often lead to non-termination even for one equation alone (cf. [8]).

To add reasoning over numbers Description Logics were extended with *concrete domains* (cf. [62]). A concrete domain is a domain separate from the usual instance domain of the model based semantics. Examples for concrete domains include different sets of numbers or strings. A specific concrete domain extension defines predicates over the concrete domain, e.g., *greater than* for numbers, or *substring* for strings. Often also a limited set of functions (for computation) can be supplied. Racer [63] implements concrete domains with numbers. But computed values are only used during reasoning and are not available to the user afterwards. OWL Equations [64], a concrete domain extension carried over to OWL, allows comparing numerical values – even computed values; still the same limitations apply.

8.5. Missing Value Imputation

Several books provide information on handling missing values from the perspective of statistics as well as from social sciences, e.g. cf. [65, 66]. Within the Semantic Web community, a main focus on value completion has been in the prediction of generic relations, and mainly object relations (i.e. link-prediction) on the object level rather than on numerical values, cf. [67] for an excellent survey on such methods. The usage of numerical values is a a rather recent topic in this respect. Along these lines, but complementary to the present work, Neumaier et al. [68] (as well as similar works referenced therein) have discussed methods to assign bags of numerical values to property-class pairs in knowledge graphs like DBpedia (tailored to finding out relations such that for instance a certain set of numbers could possibly be "population numbers of cities in France"), but not specifically to complete/impute missing values.

Our method rather uses fairly standard, robust, and well-known methods (KNN, linear regression, and random forest) for numerical missing value imputation based on principle components [20].

This could be certainly refined to more tailored methods in the future, for instance using time-series analysis; indeed our

[45] http://www.ddialliance.org/Specification/RDF

[46] https://www.europeandataportal.eu/de/content/statdcat-ap-wg-virtual-meeting

[47] http://www.openmath.org/cd/contrib/cd/rdf.xhtml

predicted values, while reasonably realistic in the value ranges often show some non-realistic "jumps" when raw data for a certain indicator is available over a certain sequence of years, but missing for only a few years in between. Since the missing value imputation component in our architecture is modularly extensible with new/refined methods, such refinements could be added as future work.

9. Conclusions

In this paper we have presented the *Open City Data Pipeline*, an extensible platform for collecting, integrating, and enriching open city data from several data providers including Eurostat and UNSD. We have developed several components including wrappers, a data crawler, an ontology-based integration platform, and a missing value prediction module, which relies on both statistical regression methods and ontological inference over equational background knowledge: since we deal with very sparse datasets, the prediction (or, as it is often referred to, imputation) of missing values is a crucial component. For this, we have developed two approaches, one based on basic regression methods, and one based on exploiting known equations.

As for the former, we predict target indicators from components calculated by Principal Components Analysis (PCA). We applied three basic regression methods and selected the best performing one.

As for the latter, we have shown that the predictions computed this way can be further improved by exploiting equations, where we have estimated and verified the assumption that this combination improves prediction accuracy overall, in terms of the number of filled-in values and estimated errors.

The created prediction values are fed back into our triple store and are accessible via our SPARQL endpoint or Web UI. Here, we additionally publish provenance information including the used prediction methods and equations along with estimated prediction accuracy.

9.1. Lessons Learnt

In the wrapper component, integrating cities and indicators for a new dataset (often CSV tables) is still a slow manual process and needs custom scripting. Particularly, entity recognition for cities can only partially be automated and needs manual adaptation for each wrapper. Also, mapping indicators is still a largely manual process, where in the future, we plan to apply instance based mapping learning techniques used in ontology matching (cf. [69]). We emphasise here, that in fact such approaches could rely on and extend similar regression techniques as we used for imputing missing values.

As for our enrichment approach (combining missing value prediction techniques and equations), we have improved over the past 1 1/2 years significantly [7], not only refining our methods, but also by proving the conjecture that the more data we collect, the better the predictions actually get: by applying the PCA-based prediction approach, using *standard* regression techniques without customisation, we reach a good quality for predictions (overall RMSE% of 0.55%) and are able to fill large gaps of the missing values, which can be further improved by the combination with equational knowledge.

The republication of our enriched dataset as Statistical Linked Data [3], using the standard QB format shall allow combining and integrating our dataset with other datasets of the Global Cube [16].

9.2. Future Work

Our future work includes extensions of the presented datasets, methods, and the system itself.

Currently, the data sources are strongly focused on European cities or demographic data. Hence, we aim to integrate further national and international data sources, in particular the U.S. Census Bureau statistics and the Carbon Disclosure Project.

The U.S. Census Bureau [70] offers two groups of tabular datasets concerning U.S. statistics: *Table C-1 to C-6* of [70] cover the topics Area and Population, Crime and Civilian Labor Force for cities larger than 20 000 inhabitants; *Table D-1 to D-6* of [70] cover Population, Education, Income and Poverty for locations with 100 000 inhabitants and more. Contrary to the UNSD or Eurostat datasets, the U.S. Census Bureau datasets have a low ratio of missing values ranging from 0% to 5% for a total of 1267 cities. The data includes 21 indicators, e.g., population, crime, and unemployment rate.

The Carbon Disclosure Project (CDP) is an organisation based in the U.K. aiming at "[...] using the power of measurement and information disclosure to improve the management of environmental risk"[48]. The *CDP cities* project has data collected on more than 200 cities worldwide. CDP cities offers a reporting platform for city governments using an online questionnaire covering climate-related areas like Emissions, Governance, Climate risks, Opportunities, and Strategies.

Many cities operate dedicated open data portals. The data from these individual city open data portals (e.g., New York, Vienna) could be added and integrated. This is surely a large effort on its own, as we would require a unified interface to many data portals. Either we would have to write wrappers for every cities' portal, or standardisation efforts on how cities publish data would have to succeed.

Apart from that, we could include sources like DBpedia or Wikidata in a more timely fashion, e.g. recording changes in values, through projects such as the DBpedia wayback machine [71], to collect also historical data from DBpedia.

Compared to [7] we have completely refurbished our crawling framework and architecture to automatically and regularly update the integrated sources dynamically. We therefore expect new lessons learnt from more regular updates; e.g.; Eurostat only since very recently updates its datasets monthly, instead of annually only,[49] which we expect to benefit from.

We also plan to connect our platform to the Linked Geo Data Knowledge Base [72] including OpenStreetMap (OSM) data.

[48] https://www.cdp.net/en-US/Pages/About-Us.aspx

[49] cf. Section 8.1 at http://ec.europa.eu/eurostat/cache/metadata/de/urb_esms.htm: "From 2017 new data will be published the first day of every month."

Based on such data, new indicators could be directly calculated, e.g., the size of public green space by aggregating all the parks. Some preliminary works on integrating indicators extracted from OSM with our Open City Data Pipeline have been presented in [73].

As we integrate more sources and datasets, another future direction we should pursue is to revisit cross-dataset predictions of missing values in more detail,[50] i.e., how predictions can be made from one data source to another. This is particularly important, as typically different data sources have only a handful (if any) of overlapping indicators.

We further aim to extend our basket of base regression methods with other well established methods. Promising candidates are Support Vector Machines [74], Neural Networks, and Bayesian Generalised Linear Models [75].

Moreover, we plan to publish more details on the best regression method per indicator as part of our ontology: so far, we only indicate the method and estimated RMSE%, whereas further details such as used parameters and regression models would be needed to reproduce and optimise our predictions. Ontologies such as [26] could serve as a starting point.

Furthermore, we are in the process of improving the user interface to make the Web application easier to use. For this we investigate several libraries for more advanced information visualisation.

Acknowledgements Part of the work was carried out with the support of the German Federal Ministry of Education and Research (BMBF) within the project "WIRE" (Grant 01lS16039B).

References

[1] Economist Intelligence Unit (Ed.), The Green City Index, Siemens AG, 2012.
[2] S. Neumaier, J. Umbrich, A. Polleres, Automated quality assessment of metadata across open data portals, ACM Journal of Data and Information Quality (JDIQ) 8 (1) (2016) 2.
[3] B. Kämpgen, S. O'Riain, A. Harth, Interacting with Statistical Linked Data via OLAP Operations, in: 9th Extended Semantic Web Conference (ESWC) Satellite Events, 2012.
[4] R. Cyganiak, D. Reynolds, J. Tennison, The RDF data cube vocabulary, W3C Recommendation, W3C (Jan. 2014).
[5] T. Lebo, D. McGuinness, S. Sahoo, PROV-O: The PROV ontology, W3C recommendation, W3C (Apr. 2013).
URL http://www.w3.org/TR/2013/REC-prov-o-20130430/
[6] L. M. A. Bettencourt, J. Lobo, D. Helbing, C. Kühnert, G. B. West, Growth, innovation, scaling, and the pace of life in cities, Proc. of the National Academy of Sciences of the United States of America 104 (17) (2007) 7301–7306.
[7] S. Bischof, C. Martin, A. Polleres, P. Schneider, Collecting, integrating, enriching and republishing open city data as linked data, in: The Semantic Web - ISWC 2015 - 14th International Semantic Web Conference, Bethlehem, PA, USA, October 11-15, 2015, Proceedings, Part II, 2015, pp. 57–75.
[8] S. Bischof, A. Polleres, RDFS with Attribute Equations via SPARQL Rewriting, in: Proc. of ESWC 2013, Springer, 2013, pp. 335–350.
[9] S. Stadtmüller, S. Speiser, A. Harth, R. Studer, Data-Fu : A Language and an Interpreter for Interaction with Read / Write Linked Data, in: 22nd International Conference on World Wide Web (WWW), 2013.
[10] G. Schreiber, Y. Raimond, F. Manola, E. Miller, B. McBride, Rdf 1.1 primer, W3C working group note, W3C (Jun. 2014).
URL https://www.w3.org/TR/rdf11-primer/
[11] S. Harris, A. Seaborne, Sparql 1.1 query language, W3C recommendation, W3C (Mar. 2013).
URL http://www.w3.org/TR/sparql11-query/
[12] P. Gearon, A. Passant, A. Polleres, Sparql 1.1 update, W3C recommendation, W3C (Mar. 2013).
URL http://www.w3.org/TR/sparql11-update/
[13] T. Berners-Lee, Linked Data, W3C Design Issues, from http://www.w3.org/DesignIssues/LinkedData.html; retr. 2010/10/27 (July 2006).
[14] J. Gray, S. Chaudhuri, A. Bosworth, A. Layman, D. Reichart, M. Venkatrao, F. Pellow, H. Pirahesh, Data Cube: A Relational Aggregation Operator Generalizing Group-By, Cross-Tab, and Sub-Totals, Data Mining and Knowledge Discovery 1 (1).
[15] B. Kämpgen, A. Harth, No Size Fits All – Running the Star Schema Benchmark with SPARQL and RDF Aggregate Views, in: 10th Extended Semantic Web Conference (ESWC), 2013.
[16] B. Kämpgen, S. Stadtmüller, A. Harth, Querying the Global Cube: Integration of Multidimensional Datasets from the Web, in: 19th International Conference on Knowledge Engineering and Knowledge Management (EKAW), 2014.
[17] A. Polleres, A. Hogan, R. Delbru, J. Umbrich, RDFS & OWL reasoning for linked data, in: Reasoning Web. Semantic Technologies for Intelligent Data Access (Reasoning Web 2013), Vol. 8067 of LNCS, Springer, Mannheim, Germany, 2013, pp. 91–149.
[18] I. H. Witten, E. Frank, Data Mining: Practical Machine Learning Tools and Techniques, 3rd Edition, Morgan Kaufmann Publishers Inc., 2011.
[19] J. Han, Data Mining: Concepts and Techniques, 3rd Edition, Morgan Kaufmann Publishers Inc., 2012.
[20] S. T. Roweis, EM algorithms for PCA and SPCA, in: Advances in Neural Information Processing Systems 10 (NIPS 1997), 1997, pp. 626–632.
[21] Office for Official Publications of the European Communities, Urban Audit. Methodological Handbook (2004).
[22] H. Paulheim, J. Fürnkranz, Unsupervised generation of data mining features from linked open data, in: Proc. of WIMS 2012, ACM, 2012, p. 31.
[23] R. Isele, J. Umbrich, C. Bizer, A. Harth, LDspider: An Open-Source Crawling Framework for the Web of Linked Data, in: 9th International Semantic Web Conference (ISWC) Posters and Demos, 2010.
[24] L. B. Statistics, L. Breiman, Random forests, in: Machine Learning, 2001, pp. 5–32.
[25] R Development Core Team, R: A Language and Environment for Statistical Computing, R Foundation for Statistical Computing, 2009.
[26] C. M. Keet, A. Lawrynowicz, C. d'Amato, A. Kalousis, P. Nguyen, R. Palma, R. Stevens, M. Hilario, The data mining OPtimization ontology, Web Semantics: Science, Services and Agents on the World Wide Web 32 (0) (2015) 43 – 53.
[27] D. Esteves, D. Moussallem, C. B. Neto, T. Soru, R. Usbeck, M. Ackermann, J. Lehmann, MEX vocabulary: a lightweight interchange format for machine learning experiments, in: Proceedings of the 11th International Conference on Semantic Systems, SEMANTICS 2015, Vienna, Austria, September 15-17, 2015, 2015, pp. 169–176.
[28] R. Hodgson, P. J. Keller, Qudt - quantities, units, dimensions and data types in owl and xml (2011).
URL http://www.qudt.org/
[29] A. Polleres, F. Scharffe, R. Schindlauer, SPARQL++ for mapping between RDF vocabularies, in: OTM 2007, Part I : Proceedings of the 6th International Conference on Ontologies, DataBases, and Applications of Semantics (ODBASE 2007), Vol. 4803 of LNCS, Springer, Vilamoura, Algarve, Portugal, 2007, pp. 878–896.
[30] T. Eiter, G. Ianni, R. Schindlauer, H. Tompits, A uniform integration of higher-order reasoning and external evaluations in answer-set programming, in: Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence (IJCAI-05), Edinburgh, Scotland, UK,, 2005, pp. 90–96.
[31] A. Polleres, J. Wallner, On the relation between SPARQL1.1 and answer set programming, Journal of Applied Non-Classical Logics (JANCL) 23 (1–2) (2013) 159–212, special issue on Equilibrium Logic and Answer Set Programming.
[32] R. Angles, C. Gutierrez, The expressive power of sparql, in: International Semantic Web Conference (ISWC 2008), Vol. 5318 of LNCS, Springer, Karlsruhe, Germany, 2008, pp. 114–129.
[33] P. R. Bevington, D. K. Robinson, Data Reduction and Error Analysis for the Physical Sciences, 3rd Edition, McGraw-Hill, Boston, 2003.

[50] Some preliminary work along these lines have been presented in [7].

[34] H. H. Ku, Notes on the use of propagation of error formulas, Journal of Research of the National Bureau of Standards 70 (4).

[35] G. Ianni, A. Martello, C. Panetta, G. Terracina, Efficiently querying RDF(S) ontologies with answer set programming, J. Log. Comput. 19 (4) (2009) 671–695.

[36] H. Santos, V. Dantas, V. Furtado, P. Pinheiro, D. L. McGuinness, From Data to City Indicators: A Knowledge Graph for Supporting Automatic Generation of Dashboards, 2017, pp. 94–108.

[37] M. Siegel, E. Sciore, A. Rosenthal, Using Semantic Values to Facilitate Interoperability Among Heterogeneous Information Systems, ACM Transactions on Database Systems (TODS) 19 (2).

[38] C. Diamantini, D. Potena, E. Storti, A Logic-Based Formalization of KPIs for Virtual Enterprises, in: Advanced Information Systems Engineering Workshops, 2013.

[39] J. M. Pérez, R. B. Llavori, M. J. Aramburu, T. B. Pedersen, Integrating data warehouses with web data: A survey, IEEE Trans. Knowl. Data Eng. 20 (7) (2008) 940–955. doi:10.1109/TKDE.2007.190746.

[40] W. Hümmer, A. Bauer, G. Harde, XCube – XML for Data Warehouses, in: 6th ACM International Workshop on Data Warehousing and OLAP (DOLAP), 2003.
URL http://portal.acm.org/citation.cfm?id=956060.956067

[41] P. Vassiliadis, T. Sellis, A Survey of Logical Models for OLAP Databases, ACM SIGMOD Record 28 (4).

[42] M. Klein, D. Fensel, F. van Harmelen, I. Horrocks, The relation between ontologies and schema-languages, LinkÃűping Electronic Articles in Computer and Information Science 6 (4).

[43] D. Calvanese, G. De Giacomo, M. Lenzerini, D. Nardi, R. Rosati, Data Integration in Data Warehousing, International Journal of Cooperative Information Systems 10 (3).

[44] A. Cal, D. Calvanese, G. D. Giacomo, M. Lenzerini, Data Integration Under Integrity Constraints, Information Systems 29 (2).

[45] M. R. Genesereth, Data Integration: The Relational Logic Approach, Morgan & Claypool Publishers, San Rafael, California (USA), 2010. doi:10.2200/S00226ED1V01Y200911AIM008.

[46] A. N. Ngomo, S. Auer, J. Lehmann, A. Zaveri, Introduction to linked data and its lifecycle on the web, in: Reasoning Web. Reasoning on the Web in the Big Data Era - 10th International Summer School 2014, Athens, Greece, September 8-13, 2014. Proceedings, 2014, pp. 1–99.

[47] S. Auer, L. Bühmann, C. Dirschl, O. Erling, M. Hausenblas, R. Isele, J. Lehmann, M. Martin, P. N. Mendes, B. V. Nuffelen, C. Stadler, S. Tramp, H. Williams, Managing the life-cycle of linked data with the LOD2 stack, in: The Semantic Web - ISWC 2012 - 11th International Semantic Web Conference, Boston, MA, USA, November 11-15, 2012, Proceedings, Part II, 2012, pp. 1–16.

[48] M. Golfarelli, S. Rizzi, Data Warehouse Design: Modern Principles and Methodologies, McGraw-Hill, 2009.

[49] A. Hogan, Exploiting RDFS and OWL for Integrating Heterogeneous, Large-Scale, Linked Data Corpora, Ph.D. thesis, Digital Enterprise Research Institute, National University of Ireland, Galway, available from http://aidanhogan.com/docs/thesis/ (2011).

[50] J. L. Ambite, D. Kapoor, Automatically Composing Data Workflows with Relational Descriptions and Shim Services, in: International Semantic Web Conference (ISWC), 2007.

[51] M. Niinimäki, T. Niemi, An ETL process for OLAP using RDF/OWL ontologies, J. Data Semantics 13 (2009) 97–119. doi:10.1007/978-3-642-03098-7_4.

[52] M. Hausenblas, W. Halb, Y. Raimond, L. Feigenbaum, D. Ayers, SCOVO: Using Statistics on the Web of Data, in: 6th European Semantic Web Conference (ESWC), 2009.

[53] D. Vrandečić, C. Lange, M. Hausenblas, J. Bao, L. Ding, Semantics of Governmental Statistics Data, in: Web Science Conference (WebSci), 2010.

[54] B. Kämpgen, R. Cyganiak, Use Cases and Lessons for the Data Cube Vocabulary, Working Group Note – http://www.w3.org/TR/2013/NOTE-vocab-data-cube-use-cases-20130801/, W3C, USA (Aug 2013).

[55] C. Lange, Ontologies and languages for representing mathematical knowledge on the semantic web, Semantic Web 4 (2) (2013) 119–158.

[56] R. R. Miner, D. Carlisle, P. D. F. Ion, Mathematical markup language (MathML) version 3.0 2nd edition, W3C recommendation, W3C, http://www.w3.org/TR/2014/REC-MathML3-20140410/ (Apr. 2014).

[57] S. Buswell, O. Caprotti, D. P. Carlisle, M. C. Dewar, M. Gaetano, M. Kohlhase, The open math standard, Tech. rep., version 2.0. Technical report, The Open Math Society, 2004. http://www.openmath.org/standard/om20 (2004).

[58] M. Perry, J. Herring, Ogc geosparql-a geographic query language for rdf data, OGC Implementation Standard. Sept.

[59] H. Rijgersberg, M. van Assem, J. Top, Ontology of units of measure and related concepts, Semantic Web 4 (1) (2013) 3–13.

[60] R. G. Raskin, M. J. Pan, Knowledge representation in the semantic web for earth and environmental terminology (sweet), Computers & geosciences 31 (9) (2005) 1119–1125.

[61] M. Kay, N. Walsh, A. Malhotra, J. Melton, XQuery 1.0 and XPath 2.0 functions and operators (second edition), W3C recommendation, W3C (Dec. 2010).
URL http://www.w3.org/TR/2010/REC-xpath-functions-20101214/

[62] F. Baader, The description logic handbook: Theory, implementation and applications, Cambridge university press, 2003.

[63] V. Haarslev, R. Möller, Description of the racer system and its applications., Description Logics 49.

[64] B. Parsia, U. Sattler, Owl 2 web ontology language data range extension: Linear equations, W3C working group note, W3C, https://www.w3.org/TR/owl2-dr-linear/ (Dec. 2012).

[65] E. R. Buhi, P. Goodson, T. B. Neilands, Out of sight, not out of mind: strategies for handling missing data, American journal of health behavior 32 (1) (2008) 83–92.

[66] F. S. Switzer, P. L. Roth, Coping with missing data, in: Handbook of Research Methods in Industrial and Organizational Psychology, Blackwell Publishing Ltd, 2008, pp. 310–323.

[67] H. Paulheim, Knowledge graph refinement: A survey of approaches and evaluation methods, Semantic Web 8 (3) (2017) 489–508.

[68] S. Neumaier, J. Umbrich, J. Parreira, A. Polleres, Multi-level semantic labelling of numerical values, in: Proceedings of the 15th International Semantic Web Conference (ISWC 2016) - Part I, Vol. 9981 of LNCS, Springer, Kobe, Japan, 2016, pp. 428–445.

[69] J. Euzenat, P. Shvaiko, Ontology matching, 2nd Edition, Springer, 2013.

[70] U.S. Census Bureau, County and City Data Book 2007, https://www.census.gov/compendia/databooks/ (2007).

[71] J. D. Fernández, P. Schneider, J. Umbrich, The dbpedia wayback machine, in: Proceedings of the 11th International Conference on Semantic Systems, (SEMANTICS 2015), Vienna, Austria, 2015, pp. 192–195. doi:10.1145/2814864.2814889.

[72] C. Stadler, J. Lehmann, K. Höffner, S. Auer, LinkedGeoData: A core for a web of spatial open data, Semantic Web 3 (4) (2012) 333–354.

[73] M. Posada-Sánchez, S. Bischof, A. Polleres, Extracting geo-semantics about cities from openstreetmap, in: Posters and Demos Track of the 12th International Conference on Semantic Systems (SEMANTiCS'16), Leipzig, Germany, 2016.

[74] V. Sanchez, Advanced support vector machines and kernel methods, Neurocomputing 55 (1–2) (2003) 5–20.

[75] M. West, P. J. Harrison, H. S. Migon, Dynamic generalized linear models and bayesian forecasting, Journal of the American Statistical Association 80 (389) (1985) 73–83.

AppendixA. Complete Example of QB Equation Steps

An excerpt of the Eurostat indicator QB equations[51] in Turtle syntax as example for a QB equation: the Eurostat indicator definition for "Women per 100 men".

```
<http://citydata.wu.ac.at/ocdp/eurostat-equations#
     women_per_100_men> a qbe:Equation ;
  qbe:variable [ a qbe:ObsSpecification ;
    qbe:filter [ a qbe:DimSpecification ;
    qb:dimension cd:hasIndicator ;
    qbe:value cd:women_per_100_men ] ;
    qbe:variablename "?women_per_100_men"^^qbe:variableType ] ;
  qbe:variable [ a qbe:ObsSpecification ;
    qbe:filter [ a qbe:DimSpecification ;
    qb:dimension cd:hasIndicator ;
    qbe:value cd:population_male ] ;
    qbe:variablename "?population_male"^^qbe:variableType ] ;
  qbe:variable [ a qbe:ObsSpecification ;
    qbe:filter [ a qbe:DimSpecification ;
    qb:dimension cd:hasIndicator ;
    qbe:value cd:population_female ] ;
    qbe:variablename "?population_female"^^qbe:variableType ] ;
  qbe:hasEquation "?women_per_100_men = ?population_female *
       100 / ?population_male"^^qbe:equationType.
```

In the first step this equation is normalised to the following three rules[52].

```
ee:e4bb866b19a383a5c7ce88e853ff8bdad a qbe:Rule ;
  prov:wasDerivedFrom <http://citydata.wu.ac.at/ocdp/eurostat-
       equations#women_per_100_men> ;
  qbe:structure globalcube:global-cube-dsd ;
  qbe:output [
    qbe:filter _:b4bb866b19a383a5c7ce88e853ff8bdad ];
  qbe:input [ qbe:variableName "?women_per_100_men"^^qbe:
       variableType ;
    qbe:filter _:b4c56a2955372924bde20c2944b2b28f3 ];
  qbe:input [ qbe:variableName "?population_male"^^qbe:
       variableType ;
    qbe:filter _:b7177d26053419667c2b7deb4569a82b9 ];
  qbe:hasFunction "?population_male*?women_per_100_men/100"^^
       qbe:functionType .

_:b4bb866b19a383a5c7ce88e853ff8bdad qbe:dimension cd:
     hasIndicator ; qbe:value cd:population_female .

ee:e4c56a2955372924bde20c2944b2b28f3 a qbe:Rule ;
  prov:wasDerivedFrom <http://citydata.wu.ac.at/ocdp/eurostat-
       equations#women_per_100_men> ;
  qbe:structure globalcube:global-cube-dsd ;
  qbe:input [ qbe:variableName "?population_female"^^qbe:
       variableType ;
   qbe:filter _:b4bb866b19a383a5c7ce88e853ff8bdad ];
  qbe:output [
    qbe:filter _:b4c56a2955372924bde20c2944b2b28f3 ];
  qbe:input [ qbe:variableName "?population_male"^^qbe:
       variableType ;
    qbe:filter _:b7177d26053419667c2b7deb4569a82b9 ];
  qbe:hasFunction "100*?population_female/?population_male"^^
       qbe:functionType .

_:b4c56a2955372924bde20c2944b2b28f3 qbe:dimension cd:
     hasIndicator ; qbe:value cd:women_per_100_men .

ee:e7177d26053419667c2b7deb4569a82b9 a qbe:Rule ;
  prov:wasDerivedFrom <http://citydata.wu.ac.at/ocdp/eurostat-
       equations#women_per_100_men> ;
  qbe:structure globalcube:global-cube-dsd ;
  qbe:input [ qbe:variableName "?population_female"^^qbe:
       variableType ;
    qbe:filter _:b4bb866b19a383a5c7ce88e853ff8bdad ];
  qbe:input [ qbe:variableName "?women_per_100_men"^^qbe:
       variableType ;
    qbe:filter _:b4c56a2955372924bde20c2944b2b28f3 ];
  qbe:output [
    qbe:filter _:b7177d26053419667c2b7deb4569a82b9 ];
  qbe:hasFunction "100*?population_female/?women_per_100_men"^^
       qbe:functionType .

_:b7177d26053419667c2b7deb4569a82b9 qbe:dimension cd:
     hasIndicator ; qbe:value cd:population_male .
```

Next the QB rules are converted to SPARQL INSERT/CONSTRUCT queries. We give here the SPARQL query for the second rule above (`ee:e4c56a2955372924bde20c2944b2b28f3`) which computes `women_per_100_men`.

```
PREFIX cd: <http://citydata.wu.ac.at/ns#>
PREFIX qb: <http://purl.org/linked-data/cube#>
PREFIX sdmx-measure: <http://purl.org/linked-data/sdmx/2009/
     measure#>
PREFIX sdmx-dimension: <http://purl.org/linked-data/sdmx/2009/
     dimension#>
PREFIX dcterms: <http://purl.org/dc/terms/>
PREFIX prov: <http://www.w3.org/ns/prov#>
PREFIX qbe: <http://citydata.wu.ac.at/qb-equations#>
PREFIX cd: <http://citydata.wu.ac.at/ns#>
PREFIX globalcube: <http://kalmar32.fzi.de/triples/global-cube.
     ttl#>
PREFIX estatwrap: <http://ontologycentral.com/2009/01/eurostat/
     ns#>

INSERT {
  ?obs qb:dataSet globalcube:global-cube-ds ;
    cd:hasIndicator cd:women_per_100_men ;
    dcterms:publisher ?source ;
    dcterms:date ?year ;
    sdmx-dimension:refArea ?city ;
    sdmx-measure:obsValue ?value ;
    prov:wasDerivedFrom ?population_male_obs, ?
         population_female_obs ;
    prov:wasGeneratedBy ?activity ;
    prov:generatedAtTime ?now ;
    cd:estimatedRMSE ?error .

  ?activity a prov:activity ;
  prov:qualifiedAssociation [
    a prov:Association ;
    prov:agent   cd:import.sh ;
    prov:hadPlan <http://citydata.wu.ac.at/ocdp/eurostat-rules#
         e4c56a2955372924bde20c2944b2b28f3> ] .
}
WHERE { { SELECT DISTINCT * WHERE {
  ?population_male_obs qb:dataSet globalcube:global-cube-ds;
    cd:hasIndicator cd:population_male ;
    dcterms:date ?year;
    sdmx-dimension:refArea ?city ;
    sdmx-measure:obsValue ?population_male ;
    cd:estimatedRMSE ?population_male_error .

  ?population_female_obs qb:dataSet globalcube:global-cube-ds;
    cd:hasIndicator cd:population_female ;
    dcterms:date ?year;
    sdmx-dimension:refArea ?city ;
    sdmx-measure:obsValue ?population_female ;
    cd:estimatedRMSE ?population_female_error .

  BIND(CONCAT(REPLACE("http://citydata.wu.ac.at/ocdp/eurostat-
       rules#e4c56a2955372924bde20c2944b2b28f3", "
       e4c56a2955372924bde20c2944b2b28f3", MD5(CONCAT("http://
       citydata.wu.ac.at/ocdp/eurostat-rules#
       e4c56a2955372924bde20c2944b2b28f3",STR(?
       population_male_obs), STR(?population_female_obs))))) AS
        ?skolem)
  BIND(IRI(CONCAT(?skolem, "_source")) AS ?source)
  BIND(IRI(CONCAT(?skolem, "_obs")) AS ?obs)
  BIND(IRI(CONCAT(?skolem, "_activity")) AS ?activity)
  BIND(NOW() as ?now)

  ## computation and variable assignment
  BIND(100.0*?population_female*1.0/IF(?population_male != 0, ?
       population_male, "err") AS ?value)

  ## error propagation
  BIND((ABS(100.0)+0.0)*(ABS(?population_female)+?
       population_female_error)*1.0/IF((ABS(?population_male)-?
       population_male_error) != 0.0, (ABS(?population_male)-?
       population_male_error), "err")-100.0*?population_female
       *1.0/IF(?population_male != 0, ?population_male, "err")
       + 0.1 as ?error)
```

[51] http://citydata.wu.ac.at/ocdp/eurostat-equations

[52] http://citydata.wu.ac.at/ocdp/eurostat-rules

```
FILTER(?error > 0.0)

## 1st termination condition:
## there exists no better observation with the same dimension
      values
FILTER NOT EXISTS {
    ?obsa qb:dataSet globalcube:global-cube-ds ;
      dcterms:date ?year;
      sdmx-dimension:refArea ?city ;
      cd:hasIndicator cd:women_per_100_men ;
      sdmx-dimension:sex ?sex ;
      estatwrap:unit ?unit ;
      sdmx-dimension:age ?age ;
      cd:estimatedRMSE ?errora .
    FILTER(?errora < ?error) }

## 2nd termination condition:
## the same equation was not used for the computation of any
     source observation
FILTER NOT EXISTS { ?population_male_obs prov:wasDerivedFrom
     */prov:wasGeneratedBy/prov:qualifiedAssociation/prov:
     hadPlan/prov:wasDerivedFrom? <http://citydata.wu.ac.at/
     ocdp/eurostat-rules#e4c56a2955372924bde20c2944b2b28f3> .
      }
FILTER NOT EXISTS { ?population_female_obs prov:
     wasDerivedFrom*/prov:wasGeneratedBy/prov:
     qualifiedAssociation/prov:hadPlan/prov:wasDerivedFrom? <
     http://citydata.wu.ac.at/ocdp/eurostat-rules#
     e4c56a2955372924bde20c2944b2b28f3> . }
}}}
```