Undefined 0 (2015) 1–0
IOS Press

A Five-Star Rating Scheme to Assess Application Seamlessness

Editor(s): Name Surname, University, Country
Solicited review(s): Name Surname, University, Country
Open review(s): Name Surname, University, Country

Timothy Lebo *,**, Nicholas Del Rio, Patrick Fisher, and Chad Salisbury
Air Force Research Laboratory, Information Directorate
Rome, NY, USA

Abstract. Analytics is a widespread phenomenon that often requires analysts to coordinate operations across a variety of incompatible tools. When incompatibilities occur, analysts are forced to configure tools and transform or *munge* data, distracting them from their ultimate task objective. This additional burden is a barrier to our vision of *seamless analytics*, i.e. the use and transition of content across tools without incurring significant costs. Our premise is that standardized semantic web technologies (e.g., RDF and OWL) can enable analysts to more easily munge data to satisfy tools' input requirements and better inform subsequent analytical steps. However, although the semantic web has shown some promise for interconnecting disparate *data*, more needs to be done to interlink user- and task-centric, analytic *applications*. We present five contributions towards this goal. First, we introduce an extension of the W3C PROV Ontology to model analytic applications regardless of the type of data, tool, or objective involved. Next, we exercise the ontology to model a series of applications performed in a hypothetical but realistic and fully-implemented scenario. We then introduce a measure of seamlessness for any ecosystem described in our Application Ontology. Next, we extend the ontology to distinguish five types of applications based on the structure of data involved and the behavior of the tools used. By combining our 5-star application rating scheme and our seamlessness measure, we propose a simple Five-Star Theory of Seamless Analytics that embodies tenets of the semantic web in a form which emits falsifiable predictions and which can be revised to better reflect and thus reduce the costs embedded within analytical environments.

Keywords: analytics, interoperability, Linked Data, semantics, evaluation

1. Introduction

Linked Data (LD) is a large, decentralized, and loosely-coupled conglomerate covering a variety of topical domains and slowly converging to use well-known vocabularies [1,2]. To more fully reap the benefits of such diverse data, LD analysts must employ an equally diverse array of analytical tools. Meanwhile, the Visual Analytics community (VA) has been forging a science of analytical reasoning and interactive visual interfaces to facilitate analysis of "*overwhelming amounts of disparate, conflicting, and dynamic information* [3]." Although the community has produced a vast array of tools and techniques that could assist [4], it remains difficult to easily reuse those tools in evolving environments such as the world of LD analytics – perhaps because they rely on non-semantic representations that make it difficult to establish and maintain connections across analyses. Regardless of which community's approaches are adopted, the need to continually form interconnections among the triad consisting of *data*, *analyst*, and *tool* remains a costly endeavor – and to benefit from both VA and LD research, the costs need to be more clearly portrayed, assessed, and overcome.

*Corresponding author. E-mail: Timothy.Lebo@us.af.mil
**DISTRIBUTION STATEMENT A. Approved for public release; distribution is unlimited. Case Number: 88ABW-2014-5577

We attribute these analytical costs to two major factors:

- the ability to integrate software tools with arbitrary data
- the ability to reuse prior analytical materials

With respect to applying software, the flexibility afforded by new APIs such as D3 [5] has resulted in a proliferation of "one-off" visualization tools that inhibit low-cost reusability. These new visualizations regularly assume specific input data formats about specific topics that are not explicitly expressed, and can cause analysts to spend up to 80% of their time integrating these kinds of tools [6]. Even if analysts could easily use the near two-thousand cataloged D3 visualizations[1], each visualization is a sink from the standpoint of subsequent analysts. Derived results, including interactions and selections, are often not codified in forms that can be easily reused by subsequent analysts.

Given these cost factors, we formalize a "five-star theory" that explains how these costs can be mitigated. The theory combines work from VA and LD communities and explains analytical costs in terms of data evolution (i.e, VA theory) and data structuredness (i.e., LD theory). As data evolves into ordered forms that facilitate analytic reasoning, it jumps within a dichotomous space of mundane (i.e., non-semantic) and semantic forms. Figure 1 shows that our five-star theory is just one possible theory in a class of possible analytical cost theories. We believe that an analytical cost theory should be built from three major components: Member of the theory class should all contain: a model from which to describe analyses, a cost metric to assess analyses, and cost reduction strategies.

Our contributions and sectioning of this paper are also illustrated in Figure 1. As shown at the bottom of the image, Section 2 introduces an extension of the W3C PROV Ontology to model analytic applications regardless of the type of data, tool, or objective involved. Section 3 (not shown) exercises the ontology to model a series of applications performed in a hypothetical but realistic and fully-implemented scenario. Section 4 introduces a measure of seamlessness based on the cost of performing applications in ecosystems described using our application ontology. Section 5 extends the application ontology to distinguish five types

[1] http://christopheviau.com/d3list/ maintains a list of public D3 visualizations. The current count as of December 10, 2014 was 1,897 visualizations.

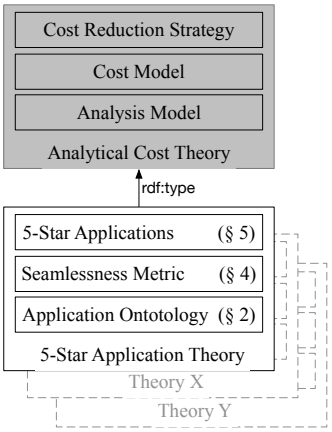

Fig. 1. A theory of seamless analytics comprises three elements: a model, a cost model, and cost reduction strategies.

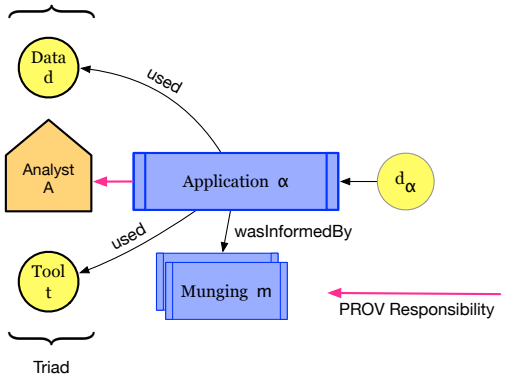

Fig. 2. Application Ontology Core is an extension of PROV. Applications use tools to generate new datasets which could include visualizations. Applications are informed by munging activities that transform data representations. Figure 9 illustrates an extension to further distinguish among five types of applications.

of applications that progressive reduce the cost of analyses. Section 6 describes past work in the area of analytical models and techniques for supporting interoperability in analytical environments. Finally, Section 7 discusses future work before concluding in Section 8.

2. An Ontology of Analytical Applications

Our core Application Ontology (AO) provides a minimal set of concepts to describe an analytical step, herein known as an **application**; a complete thread of analysis can be chained together when subsequent applications use materials from previous applications, as exemplified in Section 3. Application chains can

then be assessed using the seamlessness measure introduced in Section 4 and can be further distinguished into five sub-types using the constraints introduced in Section 5.

An application refers to an analyst's contextualized use of some dataset within a tool to achieve some implicit objective, which contrasts with prior work of modeling applications as a piece of software [7]. Our applications are a kind of PROV Activity [8], defined as "*something that occurs over a period of time and acts upon or with entities; it may include consuming, processing, transforming, modifying, relocating, using, or generating entities*." An application associates three key entities, which we refer to as the application *triad*: 1) the dataset used, 2) the performing analyst who also most immediately benefited from the result, and 3) the tool that derived a result from the dataset. Figure 2 illustrates these relations using the PROV layout conventions[2] – d_α is the result derived from dataset d by analyst A with tool t during application α.

The distinguishing aspect of our AO is the focus on **munging** activities that may be required to suit a dataset to a tool's input requirements. Munging, also known as wrangling, is the imperfect manipulation of data into a usable form and has been recognized in VA field for decades, yet continues to be a ubiquitous and costly problem [9]. We focus on munging because it persists and dominates as a cost factor for applications. The relationship between applications and munges is also shown in Figure 2 using PROV, but we further relate munging activities as also being *part of* the application[3].

As shown in Figure 3, we establish seven subclasses of munging and group them into three intermediate super-classes. These intermediate classes (*mundane*, *semantic*, and *trivial* munging) are distinguished according to a dichotomy that can be found within Tim Berners-Lee's Linked Data rating scheme [1]. Broadly speaking, Berners-Lee's scale can be used to partition data into two groups: non-RDF and RDF. Let $D_{[1,3]}$ denote the union of all data earning one, two, or three stars according to the popular scheme, and $D_{[4,5]}$ the union of all four or five-star data. We call any dataset within $D_{[1,3]}$ "*mundane*" and any dataset within $D_{[4,5]}$ "*semantic*," reflecting the perspective of the Semantic Web (SW) and LD communities that more highly rated data are easier to use or inherently provide more value.

[2] http://www.w3.org/2011/prov/wiki/Diagrams

[3] Using Dublin Core hasPart, http://purl.org/dc/terms/hasPart

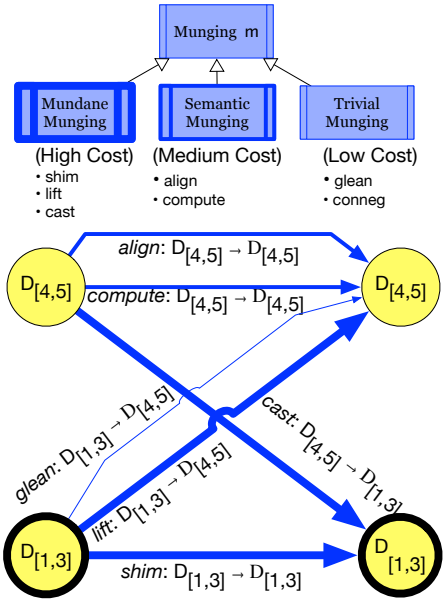

Fig. 3. Munging activities defined in terms of the Tim Berners-Lee's linked data scale. Not shown is content negotiation because it applies to all data types (an ideal situation).

Let the function tbl return the star rating of a dataset, i.e., $tbl(d) = s$. The seven sub-classes of munging (*shim*, *lift*, *cast*, *align*, *compute*, *glean*, and *conneg*) are defined in terms of using[4] data from either $D_{[1,3]}$ or $D_{[4,5]}$ and generating data from the same.

$$\text{munge} : \{D_{[1,3]}, D_{[4,5]}\} \mapsto \{D_{[1,3]}, D_{[4,5]}\}$$

Mundane munges incur the highest cost and are shown in Figure 3 with heaviest edges. Semantic munges are less expensive than mundane munges and are shown with medium weight lines. Finally, trivial munges are the least expensive of all and are shown with lightest lines. The abstract and coarse level cost is intended to reflect the ease at which data can be used within and across applications.

2.1. Mundane Munging

Three kinds of munging activities are common in that they all require the analyst to understand *both* the structure *and* semantics of mundane datasets ($D_{[1,3]}$).

Shimming ($shim$): generates $D_{[1,3]}$ from $D_{[1,3]}$; it is any data transformation that does not involve

[4] We continue to follow PROV terminology to describe activities.

RDF and is the kind of activity that the LD community is working to ameliorate.

Lifting ($lift$): generates $D_{[4,5]}$ from $D_{[1,3]}$; it creates RDF from non-RDF and has occupied the LD community's attention for most[5] of the past decade [10,11,12].

Casting ($cast$): generates $D_{[1,3]}$ from $D_{[4,5]}$; it creates mundane forms from RDF and, unfortunately, is regularly performed by many Linked Data applications today, typically by using SPARQL to create browser-friendly HTML or SVG.

2.2. Semantic Munging

Two kinds of munging activities are common in that they require the analyst to understand *only* the semantics of datasets ($D_{[4,5]}$).

Aligning ($align$): generates $D_{[4,5]}$ from $D_{[4,5]}$; it derives new relationships from RDF and can often be achieved using ontological mappings [13].

Computing ($comp$): generates $D_{[4,5]}$ from $D_{[4,5]}$; it derives new information from RDF that is itself also expressed in RDF. While aligning is a special kind of computing, there are many other kinds of computing that are not aligning. Computing is relatively less common in current practice but can be found in a few works such as Linking Open Vocabularies[6] and SPARQL-ES [14].

2.3. Trivial Munging

Two kinds of munging activities are common in that they *do not* require the analyst to understand any of the dataset's structure or semantics.

Gleaning ($glean$): generates $D_{[4,5]}$ from $D_{[1,3]}$; the GRDDL[7] and RDFa recommendations are both approaches that can be used to glean RDF from non-RDF representations without the need for contextual knowledge.

Content Negotiation ($conneg$): generates $D_{[1,5]}$ from $D_{[1,5]}$ and "*refers to the practice of making available multiple representations via the same URI.*"[8]

[5] http://triplify.org/challenge
[6] http://lov.okfn.org/dataset/lov/
[7] http://www.w3.org/TR/grddl/
[8] http://www.w3.org/TR/webarch/

3. An Analytical Scenario: Space Junk

This section presents two representative analyses modeled according to our application ontology presented in the previous section. Both analyses are centered on the broad topic of Earth's artificial satellites, e.g., their locations, type distribution, and associated launch sites. As our two analysts perform applications and inspect generated results, they will incrementally and serendipitously gain insight, formulate new questions, and perform subsequent applications to address their new inquiries. Collectively, our two analysts exemplify a "subsequent analyst" setting, where results generated by a prior analyst are reused by a different analyst with a different objective.

We also use this scenario to highlight the intersection between the Visual Analytics (VA) and Linked Data (LD) communities. The VA community understands how data evolves into ordered forms that facilitate analytical reasoning [15,16,17]; order is subjective and defined by the analyst. The LD community understands how structuredness impacts data usage in terms of discovery and integration [1,18]. We describe our representative analyses from both perspectives. As information evolves into more ordered forms, it spans various levels of structuredness.

Ironically, the ordered forms generated by applications are typically unstructured. This "irony of use" is exemplified by certain munging anti-patterns that generate mundane results. We focus on three such patterns that we name "flatlines," "house tops," and 'hill slides" due to the signatures they exhibit when they are visualized. Additionally, we use the munging anti-patterns to explain certain analytical pain points [19,9] that have been documented by the VA communities:

[pp1] : decoding semantics of mundane data
[pp2] : reusing prior application results
[pp3] : avoiding redundant work
[pp4] : obtaining different representations of data
[pp5] : understanding tools' input data requirements
[pp6] : developing or modifying t's code base
[pp7] : obtaining the provenance of results

The applications described in this section are *instances* of the application class described in the previous section. To identify these application instances, we use subscripts; for example, α_1 denotes the first application an analyst performs. We also use subscripts to identify the result instance generated by a specific application; for example, d_{α_1} denotes the result generated by the first application. Finally, we use a pair

of subscripts to identify intermediate result instances; for example, $d_{1,2}$ denotes the dataset generated by the second munge of the first application. To disambiguate applications and datasets across multiple analyses, we will specify the analyst's name, for example: Amy's α_1 or Bart's $d_{1,2}$.

3.1. Amy's Analysis

Amy, a student enrolled in a physics course, is learning about satellite launch trajectories and becomes curious about the extent of equipment launched into space. Although her professor mentions there are more than 2,000 satellites launched by various countries, she remains curious about the location and type of these satellites and seeks some visualizations to gain perspective.

3.1.1. Application 1 (α_1): Where are the Satellites Located?

Amy's professor provides her with a URL to a Keyhole Markup Language (KML) dataset that contains all satellites currently in orbit[9]. She uses a Geographical Information System (GIS) tool to plot the geolocation of satellites contained in the KML dataset.

Amy's activities are represented by the provenance trace labeled α_1 in Figure 4. The provenance trace is expressed in terms of munges and describes how Amy transformed the KML dataset, $d_{1,1}$, into a geospatial map result, d_{α_1}. She used a GIS tool, such as Google Earth to perform the shim.

Amy's provenance trace for α_1 exemplifies the "flatline" anti-pattern, which is labeled in Figure 4. Amy shimmed a mundane KML dataset, $d_{1,1}$, into another mundane form, d_{α_1}, which may be difficult to reuse in subsequent applications since it resides as unstructured data **[pp2]**.

Amy's interactive map, presented at the top of Figure 5, shows the location of over 50,000 satellites scattered throughout Earth's orbit. The map also provides an interactive legend that lets Amy toggle between the visibility of certain kinds of satellites. Amy notices that satellites are grouped according to whether they are *Rocket Bodies*, *Debris*, *Active*, or *Inactive*, and becomes inspired to calculate the efficiency of active satellites to "space junk", i.e., all satellites that are not active. Unfortunately, the geospatial map does not provide a count for each satellite type from which Amy could use to calculate the efficiency ratio.

[9] http://apps.agi.com/SatelliteViewer/

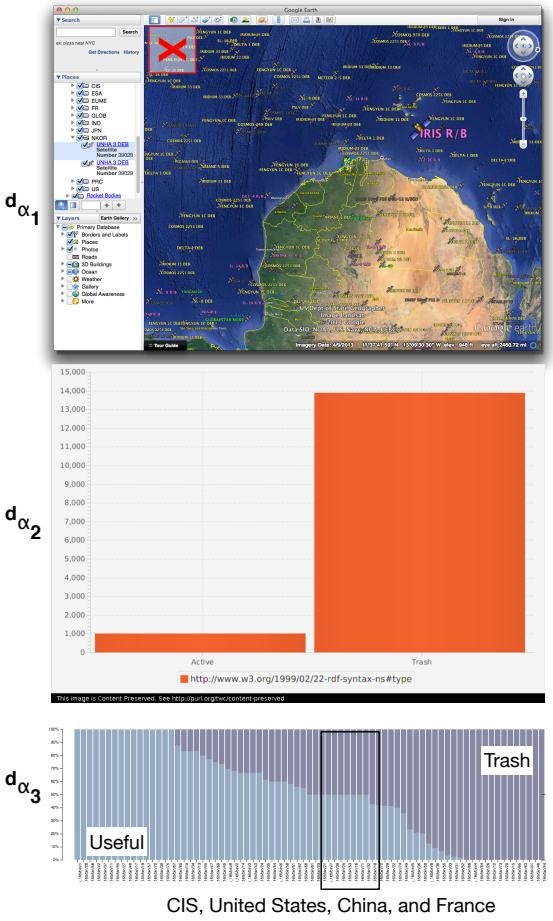

Fig. 5. Amy's application results.

Amy is also unable to select and export any subset of satellites painted on screen, such as the group of 'junk" satellites. The best she can do is take a screen shot or keep the map visualization window open on her desktop so she can reference it in later applications. In this case, Amy would serve as the communication channel between the map result and subsequent applications, which is inefficient. Fortunately, the map visualization displays the URL of the KML dataset being rendered, thereby providing Mary with a kind of natural provenance **[pp7]**.

3.1.2. Application 2 (α_2): What is the Efficiency of Satellites Launches?

Unlike her previous effort, Amy can begin her second inquiry using materials generated by her first application:

$d_{1,1}$	a URL to a KML dataset of satellites
d_{α_1}	a geospatial map of "useful" satellites

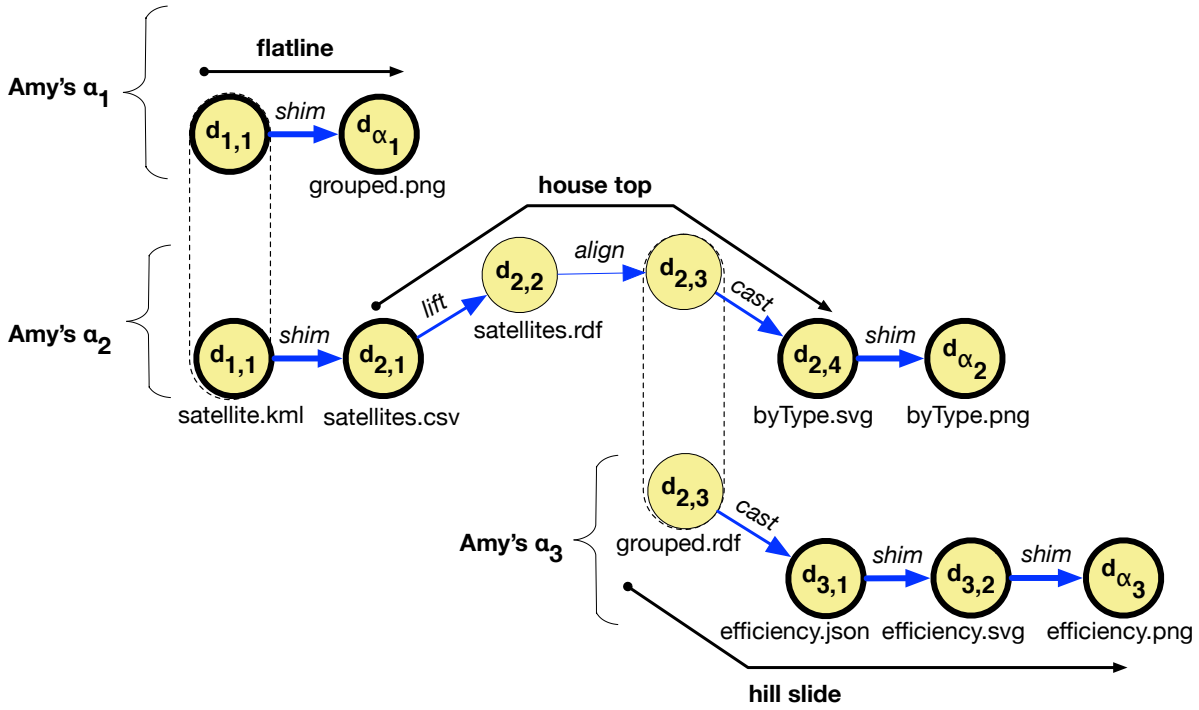

Fig. 4. Amy's analysis described using munge glyphs.

The geospatial map is most in sync with Amy's current understanding of satellites types; there exists a set of satellites painted on screen that Amy regards as useful and another set that she regards as junk. She could use the map to calculate the efficiency ratio, but her imposed groupings only exist in the form of pixels on the screen, which are difficult to reuse **[pp2]**. To reuse the map, Amy would need to employ expensive image processing to map specific groupings of pixels to data elements [20]. That route, however, would require her to use new experimental tools which would still keep data at the mundane level **[pp1]**.

The satellite KML dataset, on the other hand, is structured and therefore easier for Amy to process. Unfortunately, the KML dataset is devoid of Amy's imposed satellite groupings; the data only tags satellites as *Rocket Bodies*, *Debris*, *Active*, or *Inactive*. If Amy reuses the satellite KML, she will have to reestablish her "useful" and "junk" satellite groupings and thereby redo work she performed while interacting with the geospatial map **[pp3]**. This kind of inefficient reuse is commonplace in enterprise settings where analysts prefer the actual source data (e.g., databases) over the derived results (e.g., CSV files extracted from spreadsheets) [9]. A common challenge for analysts is to find materials that are both easy to munge and representative of the analyst's mental schema [15].

With this tradeoff in mind, Amy falls back and uses the satellite KML dataset to generate a histogram depicting the type distribution of satellites. Histograms implant qualitative information into the visual plane [21], which allow analysts to easily assess ratios. She first partitions satellites into her imposed groupings: "useful" and "junk". Amy then uses an RDF visualization tool to generate the histogram representing launch efficiency.

Amy's second application is described by the provenance trace labeled α_2 in Figure 4. The application reuses the satellite KML data, $d_{1,1}$, as indicated by the dashed lines in the figure. Amy first used a custom script to shim the satellite KML to a CSV file denoted as $d_{2,1}$. She then used a Linked Data converter [11] to lift the CSV file into an equivalent RDF representation, $d_{2,2}$. Once she obtained RDF, Amy used an ontology mapping tool [13] to align her satellite RDF into a new dataset, $d_{2,3}$, which classifies rocket bodies, debris, and inactive satellites as *trash*. She controlled the mappings by specifying the following RDFS subclass axioms:

```
SubClassOf( :Debris nfo:Trash )
```

```
SubClassOf( :RocketBody      nfo:Trash )
SubClassOf( :Inactive        nfo:Trash )
```

Amy finally used an RDF visualization tool to cast the RDF dataset of satellite groups into an SVG histogram, $d_{2,3}$, and then a PNG image, d_{α_2}, that shows the distribution of satellites by type (i.e., useful or junk). In practice, Amy could use a tool such as Sgvizler [22], which requires that developers annotate HTML with instructions on how to execute SPARQL queries. When using these kinds of tools, analysts must play the role of developers and thereby incur addition costs **[pp6]**.

The munge segment from $d_{1,2}$to $d_{2,2}$ is a stop-gap approach to obtain linked data, which is tolerable since standards for converting to linked data are relatively new[10]. Ideally, Amy would have obtained RDF using low cost techniques, such as content negotiation [23], GRDDL, and RDFa processors **[pp4]**. The greater issue is that Amy's derived semantic results fall back down to the mundane level when she generates the histogram, or any result for that matter. Once again, rich semantic connections describing the satellite data were lost during the cast to SVG. If the labels are not informative and the context of the histogram generation is forgotten, it may be difficult for subsequent analyst understand what the graphics represent **[pp1]**. We refer to this lift-then-cast anti-pattern as a "house top", as seen in Figure 4.

The histogram, shown in the center of Figure 5, provides Amy with an easy, side-by-side comparison of relative bar lengths, which depict the number of useful and junk satellites. Amy can clearly see an order of magnitude difference between useful satellites and trash, which leads her to think that countries are inefficient when launching space materials. She does not know, however, which countries are most responsible for the resulting environmental condition. She performs the next application to explore launch efficiency on a per-country basis.

3.1.3. *What is the Efficiency of Satellite Launches per Country?: Application 3 (α_3)*

Amy can begin her next inquiry using materials generated by her two previous applications (shown are materials from α_2):

- $d_{2,1}$ a CSV file of satellites
- $d_{2,2}$ an RDF representation of the satellites
- $d_{2,3}$ an RDF representation of the satellites, grouped as useful or junk
- $d_{2,4}$ an SVG histogram showing satellites distribution by type
- d_{α_2} an PNG image of the histogram

Amy reuses the grouped satellites $d_{2,3}$ to generate a stacked bar chart that shows launch efficiency on a per-country basis. She chose to reuse $d_{2,3}$ so that she could avoid expensive munges associated with SVG and PNG histogram representations of the satellite data **[pp1,pp2]**. Additionally, $d_{2,3}$ contains her imposed satellite groupings.

As presented by the munges in Figure 4, Amy used a custom script to cast the RDF data into a JSON file, $d_{3,1}$, which conforms to the structure required by the stacked bars tool. Visualization widgets, such as D3 stacked bars[11] often impose custom format requirements which are not explicitly or formally described. This lack of documentation forces analyst to inspect source code in order to infer the ingestion requirements of a tool. In Amy's scenario, the stacked bars tool only provided an example input CSV dataset, such as the one show below:

```
State,5 Years,5 to 13 Years,Over
AL,310504,552339,259034
AK,52083,85640,42153
AZ,515910,828669,362642
```

After tediously inspecting the example dataset and the widget's JavaScript code, Amy was able to decode the input semantics. She realized that each row in the table corresponds to a composite bar. The first column specifies the label of the composite bar and the following columns specify the sizes of the sub-bars. She also realized that the widget accepts JSON versions of the CSV file, and that the input data can specify an arbitrary number of sub-bins with the caveat that all rows must provide data for every column **[pp5]**.

Armed with the knowledge about what the stacked bars accepts, Amy produced a compliant JSON version dataset, $d_{3,1}$, shown in the snippet below:

```
{owner:"United States","Active":259,"Trash":3696}
{owner:"France","Active":114,"Trash":5677}
```

Amy used the stacked bars tool to cast the JSON into a SVG file embodying the stacked bars visualization. Since stacked bars tool is web-based, the application

[10] R2RML `www.w3.org/TR/r2rml` is a more recent standard for mapping relational data to RDF.

[11] http://bl.ocks.org/mbostock/3886394

also exhibits the SVG to PNG transformation pattern between $d_{3,2}$ and the final result d_{α_3}.

Amy's provenance trace for α_3 exemplifies the hill slide anti-pattern, which is labeled in Figure 4. Amy cast the RDF dataset, $d_{2,3}$, to a mundane JSON table, which she then continually shimmed. When analysts cast semantic data to the mundane level, explicit interconnections maintained at the semantic level are lost or become implicit. This places an undue burden on analysts to remember the cypher between the mundane representations and the corresponding semantics **[pp1]**.

Amy's stacked bar chart, shown at the bottom of Figure 5, shows that most countries launch space junk, to some degree. The stacked bars visual is normalized and therefore conveys the relative efficiency of satellite launches. This perspective allows Amy to see that the Common Wealth of the Independent States (CIS), United States, China, and France all launch a large percentage of junk. She shows her friend Bart the stacked bar chart PNG image, d_{α_3}, and conveys her concerns about the inefficiency of satellite launches. She asks Bart to figure out if the United States allows any of other junk-launching countries to launch from its facilities and hands him the stacked bars image as a starting point.

3.2. Bart's Analysis

Amy enlists her friend Bart to figure out which other countries launch from facilities located in the Untied States. She initially only provides him with the PNG image showing the junk efficiency on a per-country basis.

3.2.1. Application 1 (α_1): What other Countries Launch Space Junk with the Help of the United States?

Bart quickly realizes that the PNG image, d_{α_3}, although up-to-date with Amy's current perspective, does not provide adequate information for him to perform his analysis. He would have to decode meaning from column labels and extract structured information from pixels **[pp1,pp2]**. Furthermore, Bart is unable to recover the raw materials that were used to create the image since the image does not contain any watermark or metadata describing its derivation provenance **[pp7]**. He therefore requests for all of Amy's materials, including both intermediate data and application results from d_{α_1}, d_{α_2}, and d_{α_3}, and decides to reuse the RDF dataset, $d_{2,3}$, which groups satellites according

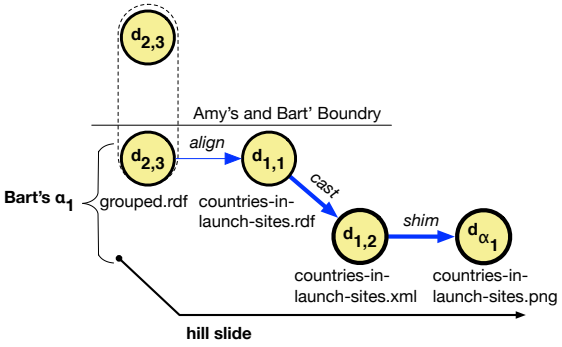

Fig. 6. Bart's analysis described using munge glyphs.

to Amy's dichotomy, i.e., her slightly older perspective that is devoid of group sizes. In practice, reviewing all of Amy's materials without any context would be a daunting task; context and meaning of the results were lost. Application results do not typically stand on their own and require the expertise of the originating analyst to fill in the lost context, through person-to-person discussion. Ideally, application results should serve as the the only interface between two different analyses.

From $d_{2,3}$, Bart generates a clustered visualization that groups countries according to the launch sites they use. This transformation sequence is described in the provenance trace labeled α_1 in Figure 6. Bart first used an ontology mapping tool to align Amy's satellite RDF into a new dataset, $d_{2,3}$, which specifies the `:inCategory` property between countries and the sites they use. He controlled the mappings by specifying the following OWL property chain:

```
SubObjectPropertyOf
(
   ObjectPropertyChain
   (
      ObjectInverseOf( acl:owner )
      prov:wasDerivedFrom
   )
   :inCategory
)
```

The property `acl:owner` specifies the country that owns a satellite. The property `prov:wasDerivedFrom` specifies the site from where a particular satellite was launched. Bart then used a custom script to cast dataset $d_{2,3}$ into an XML file that conforms to the structure required by the cluster visualization tool, such as Aduna ClusterMap[12]. He finally used the cluster tool to generate the result of the application, d_{α_1}.

[12] http://www.aduna-software.com/technology/clustermap

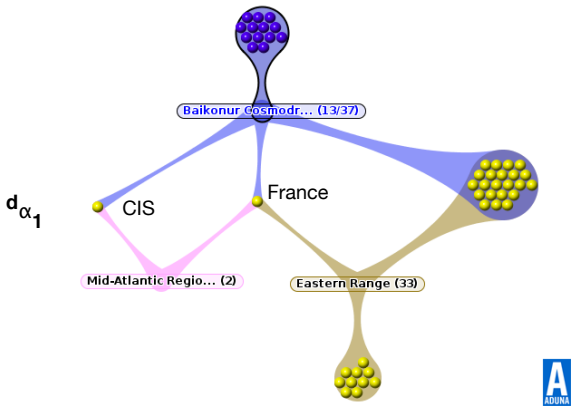

Fig. 7. Bart's application results.

From the application result, presented in Figure7, Bart can see that France launches from the "Mid-Atlantic Regional Spaceport," which is owned by the United States. Additionally, CIS launches from the "Easter Range" site, which is also owned by the United States. Bart sends the cluster map image to Amy, with some small text describing his findings.

3.3. Recap

Figure 8 presents an overview of Amy's and Bart's analyses. The top segment of the figure represents the actual analysis performed by Amy and Bart and the bottom segment represents an ideal scenario, where every application generates both a mundane and semantic representation of their result. The dashed lines in the figure indicate that a dataset was reused in a subsequent application.

In the actual analysis, every application generated only a mundane dataset. This is common signature of the anti-patterns and the reason why analysts are usually encouraged to fall back and use less developed, intermediate materials. We see this in the figure where no dashed lines extend from the arrow tips. This is consistent with previous reports that indicate analysts usually fall back to working with source databases and scripts rather than derived artifacts [9].

In the ideal scenario, every application would generate a mundane result as well as an equivalent, alternate semantic representation. Humans rely on their broadband visual channel to receive information and, therefore, will always need mundane representations of information such as rendered graphics. We believe, however, that analytical costs can be reduced if more tools generated mundane/semantic datasets, such as

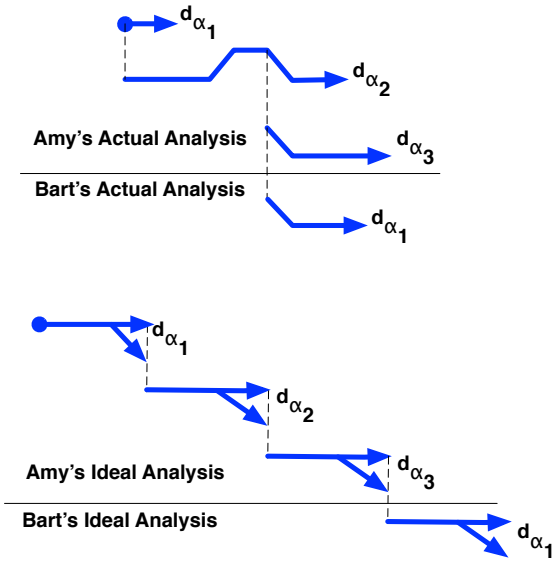

Fig. 8. Juxtaposition of an actual analysis vs. an ideal hypothetical analysis.

GRDDL and RDFa serializers. With such tools, analysts would be able to directly reuse application results and avoid duplicating prior analytical efforts.

4. A Metric for Application Seamlessness

In the previous section, Amy and Bart performed specific sequences of applications that helped them understand satellites from different perspectives. Amy generated geospatial plots and histograms, while Bart generated a a visualization that presents categorical relationships between entities. Each unique sequence of applications induces a unique *analytical ecosystem*, E. Since Amy and Bart each performed their own unique set of applications, they each induced a unique ecosystem, i.e., E_{Amy} and E_{Bart}.

Formally, an ecosystem E is defined as the set of applications that influenced[13] a particular analysis:

$$E = \{\alpha_1, \alpha_2, ..., \alpha_n\}$$

Each application α in an ecosystem is formally defined as a tuple consisting of a *non-empty* set of munges and a resultant dataset D_α:

$$\alpha = (M_\alpha = \{m_1, m_2, ..., m_m\}, d_\alpha)$$

[13] http://www.w3.org/TR/prov-dm/#term-influence

This set-theoretic definition of application α is an alternate expression of the OWL ontology described in Section 2, and is better suited for defining the seamlessness metric presented in this section. The *seamlessness* metric, S, assesses analytical cost from two different cost perspectives: (1) how easily can analysts integrate data with tools and (2) how easily can subsequent analyst reuse prior application results. The metric, thus, balances the importance of "getting the job done" with the altruistic desire to help subsequent, future analysts reuse materials.

4.1. Integration Cost

We define an equation that expresses how easily analysts can integrate data with tools. The equation is only a function of the kinds of munges performed during applications:

$$S_1(E) = \frac{\sum_{\alpha \in E} \sum_{m \in M_\alpha} cost(m)}{\sum_{\alpha \in E} \sum_{m \in M_\alpha} cost(shim)} \quad (1)$$

The numerator contains the actual cost of the ecosystem, which is calculated by summing the costs of each munge of every application in ecosystem E. The denominator reflects the hypothetical worst-case cost incurred by shimming exclusively (i.e., every munge of every application is a shim) and effectively normalizes ecosystems so that ecosystems with different munge counts can still be compared. Therefore, the equation has a range of $(0, 1]$, where lower scores indicate that analysts were able to more easily integrate data with tools. The lower bound of zero is exlusive since we do not permit munges to have a zero cost.

We see that the equation depends on a cost function that maps munge types to cost values. To bound our munge-level cost function, we first present a complete ordering of munge costs that aligns with the partial ternary ordering introduced in Section 3.

$$\begin{aligned}
cost(\alpha) &> cost(shim) \\
cost(shim) &> cost(lift) + 2\, cost(align) + cost(cast) \\
cost(lift) &> cost(cast) \\
\hline
cost(cast) &> cost(align) \\
cost(align) &> cost(comp) \\
\hline
cost(comp) &> cost(glean) \\
cost(glean) &> cost(conneg) \\
cost(conneg) &> 0
\end{aligned}$$

The horizontal lines delimit the three munge groups shown in in Figure 3; the top group corresponds with mundane munges, the middle group corresponds with semantic munges, and the bottom group corresponds with trivial munges. Therefore, the cost orderings reward applications that contain a larger proportion of trivial and semantic functions. The range of the cost function is $(0, cost(\alpha))$, which prohibits the possibility of a zero cost munges. The least expensive munge is a $conneg$ and the most expensive munge is a $shim$, which is a composite of lifting, aligning, and casting; shims incur the highest cost because they require analysts to perform mental data alignments without concrete intermediary models.

We use one such solution to the complete ordering and define a munge-level cost function, as shown below:

$$cost(m) = \begin{cases} 20 : \text{ if } shim \\ 6 \ : \text{ if } lift \\ 5 \ : \text{ if } cast \\ 4 \ : \text{ if } align \\ 3 \ : \text{ if } comp \\ 2 \ : \text{ if } glean \\ 1 \ : \text{ if } conneg \end{cases}$$

Given cost bindings, we see that he integration cost function favors applications that contain trivial and semantic munges. For example, compare Amy's application α_3 and Bart's application α_3, which both consist of only three munges. Amy's α_3 consists of one cast and two shims, which results in an integration cost of

0.75. Bart's α_1, on the other hand, consists of an align, a cast, and a single shim, which results in a cost of 0.48.

In practice, analysts should assign costs that based on different measures, e.g., man hours, lines of code, and commit frequencies. As long as the ordering constraints are satisfied, Analysts can experiment with different cost valuations and obtain new seamlessness scores that are consistent with previously computed rankings of their ecosystems. For example, given two ecosystems E_1 and E_2, where $S_1(E_1) < S_1(E_2)$ was established using cost function c, the ranking will hold under a different cost function c^prime, so long as both c and c^1 satisfy the same cost order constraints. We can consider c and c^1 as kind of scaling factors on application costs as a whole.

4.2. Reuse Potential

We define an equation that expresses how easily subsequent, analysts can reuse materials generated by prior analyses. Since this score is looking at the seams (i.e., data) between different ecosystems, the score is a function of the *kind* of results that are generated by applications. We assume that linked data, including data that can be trivially munged to yield linked data, are easier for subsequent analysts to reuse. On the other hand, mundane results such as PowerPoint slides, CSV files, and raster images are harder to reuse [9]. For example, Bart reused Amy's less contextualized RDF file instead of the PNG image result.

To embody this premise, we define the potential (pot) function that returns a set of scaling factors based on the type of D_α:

$$pot(d_\alpha) = \begin{cases} \frac{1}{cost(shim)} & : \text{ if } tbl(d_\alpha) > 3 \\ \frac{1}{cost(align)} & : \text{ if } conneg(d_\alpha) \neq \emptyset \\ \frac{1}{cost(glean)} & : \text{ if } glean(d_\alpha) \neq \emptyset \\ 1 & : \text{otherwise} \end{cases}$$

If a dataset is encoded in RDF, the pot function provides the greatest reward by returning the smallest scaling factor. If a dataset can be content negotiated to RDF, the function provides a smaller award since subsequent analysts would have to perform a content negotiation munge to acquire RDF. If a dataset can be gleaned to RDF, the function provides the smallest reward since subsequent analysts would have to perform a glean munge, which costs slightly more than content negotiation. Finally, if a dataset does not satisfy any of the above conditions, the function provides no reward; a scaling factor of 1 has no effect.

Like the munge cost function, we expect analysts to reconfigure the pot function to best represent their work environment. Perhaps, for provenance concerns (i.e., prod:alternateOf), it is more important to keep the mundane and semantic results bundled together as gleanable datasets. In this case, pot can be reconfigured to provide gleanable datasets with the greatest reward. Or perhaps, file size is an issue and therefore gleanable datasets should return very little reward. In practice, we can update the function to "follow up" on the actual reported gains by drawing from the provenance of downstream usage [24] and providing upstream analysts with feedback regarding their impact.

Since d_α can satisfy multiple conditions, the pot function returns a set of scalars, one for each bucket d_α satisfies. In these cases, subsequent analysts can choose which facet of the dataset they want to work with. For example, an analyst may be able to glean or content negotiate a single XML dataset, if the dataset contained embedded RDF or was referenced by a URL that could be content negotiated.

The pot function rewards applications that avoid expensive anti-patterns presented in Section 3. Even applications that flatline though a large portion of the trace can be rewarded, so long as they generate semantic, gleanable, or content negotiated results.

4.3. Seamlessness Score S

We can now define the seamlessness score, S, that incorporates the integration and reuse ease expressions:

$$S(E) = \frac{\sum_{\alpha \in E} \sum_{m \in M_\alpha} min(pot(d_\alpha) cost(m))}{\sum_{\alpha \in E} \sum_{m \in M_\alpha} cost(shim)} \quad (2)$$

The score uses the pot function to scale the total cost of an application. Additionally, S uses the mimimim value returned by pot in order to provide the greatest rewards. The scaled application costs reflect future "returns on investments" for potential, subsequent analysts. Although lifting mundane data to semantic forms is expensive, multiple subsequent uses of the semantic content is relatively cheaper in the long run.

4.4. Amy's and Bart's Scores

To compute Amy's and Bart's seamlessness score, we first extracted their ecosystems by examining their application traces and logging each munge and generated result d_α:

$$E_{Amy} = \{$$
$$\{\alpha_1 = \{cast, shim, shim\}, map.png\},$$
$$\{\alpha_2 = \{shim, lift, align, cast, shim\}, hist.png\},$$
$$\{\alpha_3 = \{cast, shim, shim\}, stackedbars.png\}\}$$
$$E_{Bart} = \{$$
$$\{\alpha_1 = \{align, cast, shim\}, clustermap.png\}\}$$

Table 1 presents the costs incurred by each application contained in the ecosystems. For each application, we compute its cost from the integration cost perspective, i.e., $cost(M_\alpha)$. We also compute the scaling factor from the data reuse cost perspective, i.e., $pot(d_\alpha)$, and apply the scaling factor to the application cost, i.e., $cost \times pot$. Finally, we sum the scaled application costs to compute the seamlessness score for each analysts's ecosystem.

Table 1
Amy's and Bart's seamlessness score S. The scores are broken down into their constituent integration and reuse costs.

Analyst	App.	$cost(M_\alpha)$	$pot(D_\alpha)$	$cost \times pot$
Amy	α_1	20	1	20
	α_2	55	1	55
	α_3	45	1	45
			$S(E_{Amy}) =$	0.66
Bart	α_1	29	1	29
			$S(E_{Bart}) =$	0.48

For example, in Bart's application α_1, he *aligned* ($cost = 4$), *cast* ($cost = 5$), and *shimmed* ($cost = 20$), resulting in a total application cost of 29. This same application generated a mundane PNG image of a cluster map and therefore, did not earn any cost reductions, i.e., $pot(d_\alpha)$ = 1. To compute his seamlessness score, we divide his application cost 29 by the hypothetical worst case incurred by exclusive shims, 60, to obtain a score of 0.48. The reuse cost reduction factor did not apply to any ecosystem, since both ecosystems generated only mundane results for d_α.

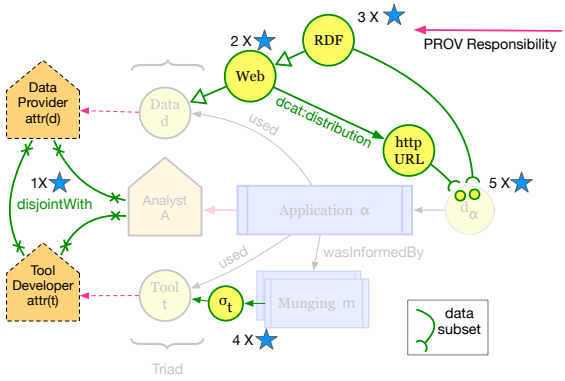

Fig. 9. An extension of the Application Ontology Core (Figure 2) to distinguish five subclasses of Application. The figure distinguishes between generic application concepts, shown in gray, and the extension concepts shown in bold-face.

5. Reducing Analytical Costs with Five-Star Applications

We propose a "5-star application rating scheme" that analysts can use to design more efficient applications that avoid anti-patterns and analytical pain points described in Section 3. The rating scheme is expressed in the form of ontological restrictions that progressively reduce the space of possible munge sequences. As the application ratings increase, the possibility of performing certain anti-patterns decrease.

We outline these ontology restrictions by extending the application ontology introduced in Section 2 to distinguish among five types of application subclasses that are illustrated in Figure 9. These subclasses are rated according their predicted cost, which is expressed as an interval. We use an interval notation to account for the fact that these application subclasses describe a set of different applications instances, each with their own scalar cost. The interval thus captures the min and max cost of the application instances in the set.

Table 2 enumerates all five application star ratings and pairs each with their associated restriction(s). Figure **??** uses this table to rate the applications performed by Amy and Bart in their analyses presented in Section 3.

5.1. One-star applications

One-star applications satisfy our fundamental restriction that the sets of analysts, tool developers, and data providers (i.e., the application triad) asso-

Table 2

A five-star rating scheme to assess analytical seamlessness of an individual application. The restrictions are specified in natural language and formally using functional and set notations. The function tbl maps the dataset D used in an application to its star rating as determined by Tim Berners-Lee's scale.

☆	Informal Restriction	Formally
1	data providers, analysts, and tool developers are disjoint	$attr(D) \cap A \cap attr(t) = \emptyset$
2	accept data (any format) via URL; cite that URL in the future	$tbl(D) >= 1 \wedge URL \in D_\alpha$
3	accept data (RDF format) via URL; cite that URL in the future	$tbl(D) >= 4$
4	use a tool's input semantics (OWL, SPARQL) when preforming munges	$used(m, \sigma_t) \wedge m \in M$
5	provide any information (RDF format) derived during use	$D \subset D_\alpha$

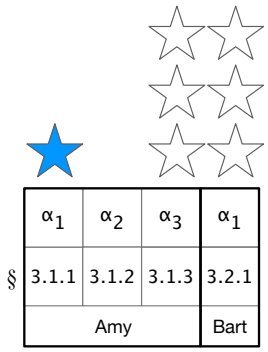

Fig. 10. Star ratings for the applications in Bart's and Amy's ecosystems. White star indicate "conditional stars."

ciated with a particular application are disjoint, i.e., $attr(d_\alpha) \cap A \cap attr(t) = \emptyset$, where A refers to the analyst and the function $attr$ supports the transitive discovery of data providers and tools developers that can be practically achieved through SPARQL queries over analytical PROV traces. Figure 9 depicts this restriction towards the left-hand side, where cross-hatched lines connecting the triad of Data Providers, Analysts, and Tool Developers represent a disjoint relationship.

In the scenario presented in Section 3, Amy's use of the GIS tool in application α_1 earns one-star since Amy was neither the provider of the satellite data nor the developer of the tool; when we state that an application instance, such as Amy's α_1, *earns* a particular start rating, this is equivalent to stating that the instance belongs to that application class. In contrast, Amy's use of the histogram tool in application α_2 does not earn one star since she was required to write HTML code. We therefore distinguish between developing munge scripts and developing the actual tool t that generates application results, where only the latter case precludes an application from earning one star.

Similarly to Amy's α_2, the use of government data mash-ups[14] and LOD metadata summaries such as Linked Open Vocabularies (LOV)[15] and SPARQL Endpoint Service (SPARQL-ES)[16] do not earn a star since the tool developers are also data providers. The LOV tree map view, for example, is immutably bound to LOV's underlying RDF store [17].

The one-star application restriction speaks more to tool developers than analysts who use the tools. If developers would design software with one-star applications in mind, they might develop more flexible tools that are not hard-coded for specific kinds of data or data sources. With such flexibility, analysts could more easily reuse tools without having to adapt the tool's source code to accept their data (pain point 6 (**[pp6]**) in Section 3).

Additionally, the one-star application class describes a set of possible munges sequences that we refer to as a *munge space*. Since the one-star application class does not place any restrictions on the structure of the data used and generated, they include application instances that consist of exclusive shims (i.e., flatlines described in Section 3) and exclusive computes. Additionally, the one-star application class allows every possible combination in between, including house tops and hill slides. We depict the one-star munge space in Figure 11. Without loss of generality, the munge space presented in the figure:

– assumes that data must *always* be munged to fit a tool

[14] http://data-gov.tw.rpi.edu/demo/USForeignAid/demo-1554.html
[15] http://lov.okfn.org/dataset/lov/
[16] http://sparqles.okfn.org/
[17] http://lov.okfn.org/endpoint/lov

– assumes *two* non-trivial munges (see Section 2 per application: one munge to get data into a tool and another munge supported by the tool itself.
– considers initial gleans and content negotiations as outside the scope and cost consideration of the application, since these munges incur such a small cost (see four- and five-star patterns in the figure)

Because each application described in this section is a class (i.e., a set of munges traces), we describe their costs in terms of intervals. The lower bound of the interval specifies the cost the cheapest possible munge sequence and the upper bound specifies the cost of the most expensive possible sequence in the munge space.

To calculate these lower and upper bounds, we can use the application cost formula presented in the previous section:

$$\sum_{m \in M_\alpha} cost(m)$$

Considering the possible munge space and the application cost formula, the cost bounds for the two-star application class is expressed by the interval:

$$\begin{aligned} cost(\alpha_\star) &= [2 \times cost(comp), 2 \times cost(shim)] \\ &= [6, 40] \end{aligned}$$

5.2. *Two-star applications*

Two-star applications accept data via URL and always cite that URL in the future. This restriction applies to any kind of data, i.e., $tbl(d) >= 1$; the function tbl maps a dataset d to its star rating as determined by Tim Berners-Lee's scale. Like the previous rating, two-star applications also fulfill the requirement that data providers, analysts, and tool developers are disjoint.

Figure 9 depicts the two-star restriction near the top, where:

– data d is available on the Web
– data d has an associated dcat:Distribution pointing to where d can be accessed
– the distribution URL is referenced by the output dataset, d_α; we use the subset relationship to denote this requirement, which is depicted by the semi-circle tail end and the embedded circle within d_α

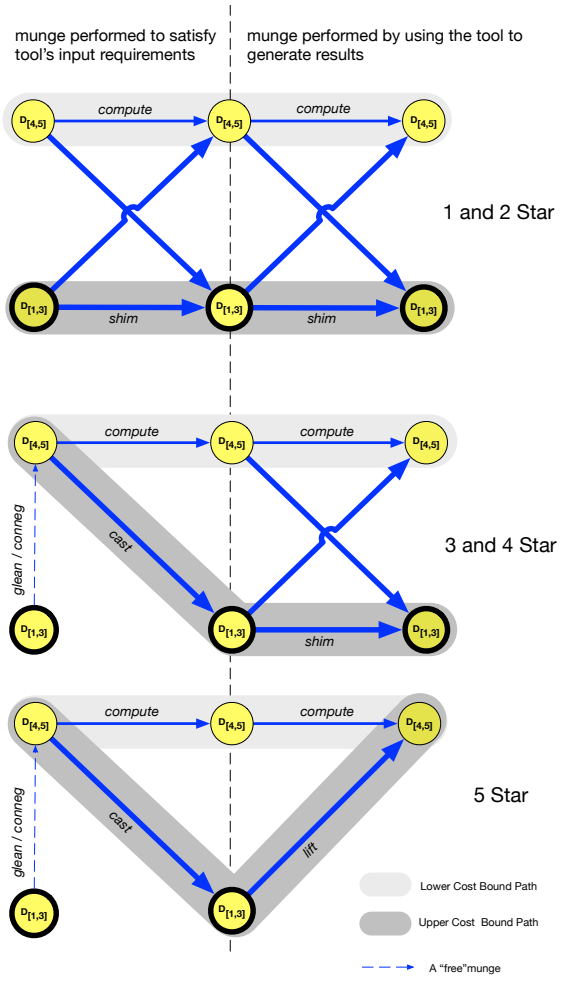

Fig. 11. Possible munge patterns associated with each application subclass. As the application restrictions increase, the space of possible munge sequences reduces.

Amy's application α_1, described in Section 3, earns two stars. The input to the application, $d_{1,1}$, was a KML satellite dataset that was available on the Web. Additionally, the resultant map, d_{α_1}, contained the URL of the input KML file, providing a simple and natural derivation provenance, **[pp7]**. We take this opportunity to reinforce the definition of an application, which is defined as a class of activities, not a software entity. Therefore, to be two-stars, an application activity must use a web accessible dataset and generate a result that cites that same dataset, despite the actual IO of the employed tool. In cases when a tool's IO does not conform with the parent application's desired star rating, the burden falls on analysts to wrap the tool invocation with munges that fulfill the desired star rat-

ing. For example, if Amy wanted to perform a two-star application, and her employed GIS tool did not generate a map citing the KML dataset, the burden would fall on Amy to inject the URL into the map, perhaps using a technique such as steganography.

In contrast, the use of LOV and SPARQL-ES would likely earn two *conditional* stars. Conditionality refers to cases when an application fulfills a particular star level requirement but fails to fulfill the immediately-preceding requirement(s). For example, although LOV and SPARQL-ES accept URLs and thus implicitly encourage analysts to use URLs for their applications, these two tools violate the one-star condition since the tool developer and data provider are the same entity.

In terms of munge space, the two-star application class is equivalent to the one-star class.

5.3. Three-star applications

Three-star applications accept RDF data via URL, i.e., $tbl(d_\alpha) >= 4$. The data can be "pure" RDF or embedded in a gleanable, mundane dataset. Like the previous rating, three-star applications must also use data available on the Web. Figure 9 presents the three-star restriction at the top, where d is an RDF dataset. The figure indicates that RDFis a subclass of Web, and thus inherits a `dcat:distribution` URL.

Both Amy's α_3 and Bart's α_1 earn three *conditional* stars. Both applications used an RDF dataset as input and both generated results that *did not* include the URLs to those input RDF datasets, i.e., the applications did not fulfill the two-star requirement. The conditional star ratings are depicted as white stars, as shown in Figure 10. Similarly, applications designed around linked data browsers [25,26,27] can earn at least three-stars iff the applicaitons meet the one- and two- star requirements. These tools accept RDF and thus encourage analysts to use RDF in their applications.

Two pain points can be alleviated when the input dataset d is RDF. Provided that d is crafted using Linked Data best practices [28], Analysts do not have to decode semantics (**[pp1]**) since these semantics are explicit and conform to community standards. Because of this affordance, analysts can more easily reuse datasets in subsequent applications (**[pp1]**). These assumptions stem from the Linked Data community, which posits that Linked Data is easier for consumers to discover, reuse, and integrate with tools.

The three-star application class defines a smaller munge space than one- and two-star application classes. If data d is encoded in RDF, it can only be computed, aligned, and cast. The three-star restriction thus removes the possibility for flatline and house top munges, although hill slides are still possible. We depict the three-star munge space in Figure 11.

The cost bounds for the three-star application class is expressed by the interval:

$$\begin{aligned} cost(\alpha_{\star\star\star}) &= [2 \times cost(comp), cost(cast) + cost(shim)] \\ &= [6, 25] \end{aligned}$$

The cost bound for the three-star application class is not only *tighter* than one- and two-star application classes, but also *lower* since the upper cost is reduced from 40 to 25.

5.4. Four-star applications

Four-star applications use a tool's input semantics (OWL, SPARQL) when performing munges, i.e., $used(m, \sigma_t) \wedge m \in M$. Like the previous rating, four-star applications also accept RDF via URL. Figure 9 depicts the four-star application restriction toward the center-bottom, where a munge m uses a tool t's input semantics σ_t during an application.

Amy and Bart did not employ any tools that provided input semantics and, therefore, neither of their ecosystems contains a four-star application. In general, four-star applications are not in widespread use, but some work in automated service orchestration relies on these kinds of applications. For example, the Semantic Automated Discovery and Integration (SADI) framework [29] requires that service providers publish input and output data requirements in the form of OWL, which provides service consumers with an unambiguous expression of the service's I/O requirements. Analysts can assess the applicability of services by inspecting formal and declarative expressions, rather than inspecting the source code or example datasets **[pp6]**, which may not be representative of the class of acceptable inputs.

In terms of munge space, the four-star application class is equivalent to the three-star class.

5.5. Five-star applications

Five-star applications output results as linked data, i.e., $d \subset d_\alpha$. Like the previous rating, five-star applica-

tions also accept RDF via URL and use a tool's input semantics during munging. Figure 9 depicts the five-star application restriction toward the right, where the RDF data used in an application is a subset of the generated result D_α.

Amy and Bart did not perform any applications that generated Linked Data, and, therefore, neither of their ecosystems contains a five-star application. Similarly, some applications analyzing the Linked Data cloud [28,30] do not earn five-stars since the results are mundane and embedded in static imagery or plain text embedded in journal articles. On the other hand, Tim Berners-Lee's tabulator [26] can also be used by analysts to perform five-star applications. Tabulator emits RDF corresponding to edits that analysts make while browsing third party data. SADI can also support five-star applications, since SADI services generate RDF that is a superset of their input.

When applications generate Linked Data, they eliminate a number of analytical pain points. For example, subsequent analysts can more easily understand prior results and more easily incorporate them in their analyses **[pp1,pp2]**. Additionally, subsequent analysts can directly reuse an application result d, rather than reusing an earlier result and rebuilding the context **[pp3]**. Finally, if an application generates both mundane and Linked Data results, it provides subsequent analysts with alternate representations to be used for different purposes **[pp4]**. Analysts can visual inspect the mundane results, but subsequent analysts can reuse the semantic alternatives.

The five-star application class defines a smaller munge space than all previous application classes. Five-star applications use RDF and generate Linked Data, or results that can be trivial gleaned to yield Linked Data; for the sake of simplicity we consider RDF and gleanable datasets to be equivalent in this section. We can therefore infer that the best case munges are exclusive computes and the worst case is a cast-lift combination or "inverted house top," as shown in Figure **??**. From the figure, we see that the five-star application class effectively eliminates the possibility for anti-patterns described in Section 3.

The cost bounds for the five-star application class is expressed by the interval:

$$\begin{aligned} cost(\alpha_{\star\star\star\star\star}) &= [2 \times cost(comp), cost(cast) + cost(lift)] \\ &= [6, 11] \end{aligned}$$

The cost bound for five-star applications is not only tighter than the three- and four-star application classes, but also lower since the upper cost is reduced from 25 to 11.

5.6. Boosting Amy's Seamlessness Scores

We will use Amy's ecosystem, E_{Amy}, from Section 3 as a baseline ecosystem to compare with an alternate, ideal ecosystem, E_{ideal}, that contains only five-star applications. We will also compare the two ecosystems in terms of their seamlessness scores in order to verify that the ideal ecosystem is more efficient. Additionally, we will provide intuition as to why the ideal ecosystem alleviates certain analytical pain points.

Ecosystem E_{Amy} contains three applications that collectively span nine munges. To facilitate a more fair comparison, we retrofitted the applications in E_{Amy} with additional lifts and gleans to produce E_{ideal}, which is five-star compliant. Therefore, our ideal ecosystem, E_{ideal}, also contains three applications that are designed around the same tools and objectives as E_{Amy}.

Figure 12 shows the provenance for the applications comprising E_{ideal}. We assume that an RDF version of the satellite dataset, d_{α_1}, existed prior to Amy's analysis. Therefore, from a more global perspective, Amy is a subsequent analyst that reused the results of a prior, anonymous five-star application. Amy first used a script to cast the satellite RDF dataset, $d_{1,1}$ to a satellite KML file, $d_{1,2}$. She then generated two, alternate representations of the satellite map, d_{α_1}, one semantic and one mundane. In practice, she could have used the GIS tool to generate the mundane version of d_{α_1} and then used a script to lift the KML dataset to an RDF representation that contains her imposed satellite groupings. For the sake of this exercise, we'll assume that the GIS tool generated a gleanable PNG image that contained the semantic representation. We consider gleanable XML and other embedding mechanisms, such ascontent preserved images [31], to be equivalent, since both approaches embed semantic content into mundane datasets.

Amy then used the map, d_{α_1}, as input for her next application, α_2. In E_{Amy}, Amy was not able to directly reuse the satellite map since it resided as a mundane PNG image that required expensive image processing to reuse. This gleanable version of the map in E_{ideal} , however, contains an embedded RDF dataset containing her imposed satellite groupings, as described in Section 3. She extracts the satellite groups, $d_{2,1}$, by

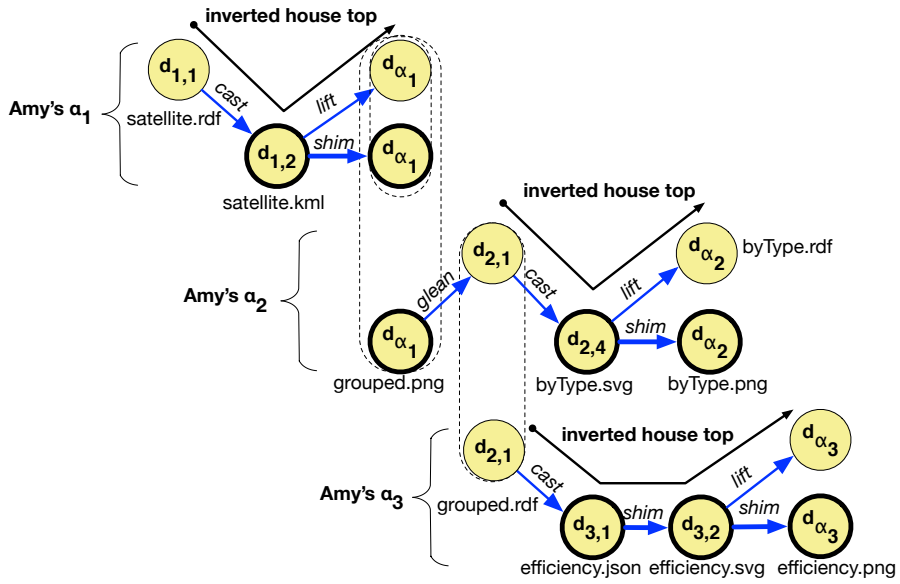

Fig. 12. Amy's ideal analysis supported entirely by five-star applications.

performing a glean and then uses the histogram tool to cast the dataset into an SVG histogram, which she then lifts to generate the application's result, d_{α_2}, i.e., an RDF representation of the SVG histogram. A snippet of the RDF histogram is shown below:

```
@prefix vsr: <http://purl.org/twc/vocab/vsr#>
@prefix sio: <http://semanticscience.org/resource/>
:bin1 a vsr:Bin;
        rdfs:label "Useful Satellites";
        sio:count "1000".
:bin2 a vsr:Bin;
        rdfs:label "Junk Satellites";
        sio:count "14000".
```

Although visually similar, a *glean*-then-cast sequence of munges is not considered a house top anti-pattern and, rather, is more akin to a hill slide, since gleanable datasets contain semantic data.

In the final application, α_3, Amy used the satellite groups, d_{2_1} as input. She used this intermediate data because it contains information about countries, whereas in ecosystem E_{Amy}, she was choose the use the group RDF only because it was in the path of least resistance in terms of munge cost. She first cast the grouped RDF into the JSON format the stacked bars tools expects. The tool also provides its input semantics describing the semantic structure of the data it expects. The SPARQL query below represents the input semantics of stacked bars, which Amy used as a conceptual munge target to generate her JSON:

```
select ?bin ?binCount ?subBin ?subBinCount
where {
  ?bin a vsr:Bin;
         sio:count ?binCount;
         dcterms:hasPart ?subBin.

  ?subBin a vsr:Bin;
         sio:count ?subBinCount.}
```

Although the SPARQL query does not include information about the particular JSON format, its conceptual description coupled with the example dataset provided by tool was enough information for Amy to produce the appropriate JSON file, $d_{3,1}$. Like applications α_1 and α_2, she generates both mundane and semantic application results.

Using the same mechanics in Section 4, we calculate the seamlessness score for E_{ideal} in Table 3. We also include the seamlessness score for the older ecosystem E_{ideal} for comparison purposes.

6. Related Work

Early visualization researchers were focused on developing models to help them understand how data is transformed into views, rather than predicting costs incurred by those transformations [32,33]. Chi devised a visualization transform model that is useful for describing how data evolves from its raw state to a view state as it passes through a four-stage pipeline of *operators* [34]. Some of these operators, for example *data stage* operators, are broadly specified and could

Table 3
The seamlessness scores for ecosystem E_{Amy} and E_{ideal}.

Analyst	App.	$cost(M_\alpha)$	$pot(D_\alpha)$	$cost \times pot$
Amy	α_1	20	1	20
	α_2	55	1	55
	α_3	45	1	45
			$S(E_{Amy}) =$	0.66
Amy′	α_1	31	0.5	15.5
	α_2	33	0.5	16.5
	α_3	51	0.5	25.5
			$S(E_{ideal}) =$	0.26

encompass munging activities. Chi's objectives, however, were centered on understanding and comparing different visualization techniques in terms of their pipeline structure, not their imposed costs [35].

In parallel, the visual analytics community has continually developed and revised analytical cost models for decades [17,16]. These models mainly consider *cognitive* costs incurred by performing user interactions [36] and visual pattern recognition. In particular, Patterson presents a cognitive model of how analysts interpret and reason with visual stimuli in order to generate responses, e.g., decisions. He presents six leverage points that visual designers should employ in order to reduce the costs imposed on analysts when interpreting visualizations [37].

When analytics researchers do consider lower level transforms, the result has been models that are specified at too high a level to easily calculate quantifiable cost predictions. Wijk proposed an economic model that considers the ratio of value (i.e., knowledge gained) to the cost incurred to generate a visualization [38]. Wijk specifically highlighted the cost to *initialize* data, which could be equated to the cost of performing munging. It is not clear, however, which specific factors influence this cost, leaving analysts with little direction as to how to better quantify and mitigate those costs.

In contrast, Kandel's work provides a more detailed description about the different kinds of data munging analysts must perform and even outlines a set of research goals [6] that paved the way for a munging tool [39]. His hierarchy of munging types and research directions were elicited from actual enterprise analysts citekandel2012enterprise, the testimonies of which provide evidence that supports our theory of analytical seamlessness. Kandel begins to discuss the use of semantic data types to address the challenges of formatting, extracting, and converting data to fit input data requirements. He mentions that these data types should be shared and reused across analyses, similarly to how the Linked Data community advocates the reuse of popular vocabularies [40]. However, we believe our seamlessness theory represents the next logical step of his work by articulating his analysts' testimonies in a form that can emit testable predictions.

Fink also outlined a set of challenges that were elicited from analysts in cyber-security settings [19]. He found that, like Kandel's enterprise subjects, analysts are limited by their capability to cheaply mitigate disparities among diverse data and tools. Some of the analysts interviewed by Fink believe that analytic environments should be as flexible as UNIX shells and allow arbitrary visualization tools to be piped together (e.g., application chains).

The models from the VA communities take a more user-centric perspective to analytics, and, therefore, typically do not incorporate the aspect of data *structuredness*; structure in VA is a more subjective and conceptual notion [15,16,17], whereas structure in LD refers to data formats. For example, the Linked Data community has long considered the potential costs and benefits associated with publishing and consuming linked structured data, but not necessarily from a analytical setting where data evolves into ordered forms that are reused by subsequent analysts. Tim Berners-Lee is a proponent of Linked Data because of the potential benefits afforded to data consumers, whom can more easily discover, integrate, and reuse distributed RDF [18]. His scheme has been useful in understanding the affordances provided to data consumers in a *client-server* setting, where data is only generated by publishers. Our work, however, uses his scheme to understand the costs and benefits in a *peer-to-peer* analytical setting, where consumers are become future publishers from the perspective of subsequent analysts.

Similarly, Janowicz and Hitzler [18] describe how the Semantic Web provides analysts with an opportunity to use third-party data in contexts not envisioned by the data provider. Analysts are able to quickly develop application-driven schema knowledge that can be used to align data into arbitrary, suitable forms required by tools. In the same spirit, Heath and Bizer describe an application architecture for linked data applications, citing data access (e.g., HTTP Get) and vo-

[18] http://5stardata.info

cabulary mapping (i.e., a kind of munging) as major components [1]. The Data Integration layer that encompasses alignment-type munges is essential for supporting the more abstract and user-driven application layer.

When "cost" is mentioned in Linked Data literature, it is referred to in an abstract manner and not usually expressed in mathematical forms from which it can be calculated. The pieces of a theory are there, however they are not consolidated and formally articulated into a framework that can be used to test and predict the community's hypotheses regarding "ease-of-use". We believe our work embodies the community's assumptions, claims, and hypothesis as a simple theory that can be used to assess, predict, and even refute (in cases when we find contrary evidence) the tenets of Linked Data that have been advertised for nearly a decade.

7. Future Work

In terms of our seamless score, we can elaborate our cost models to be more sensitive to real-world applications and consider the "user experience" factor. We can borrow from existing VA work that describes a usage cost, C_e, that denotes the "perception and exploration cost" of the analyst [38]. We can integrate this term with our application cost equation to express a more complete cost function:

$$cost(\alpha) = \sum_{m \in M_\alpha} cost(m) + C_e = cost(\alpha)$$

We can also elaborate on the distinction between mundane (1-3) and semantic (4-5) munges. Currently, our model stereotypes four- and five-star data into the same class, however, we observe significant cost differences in creating quality five-star data [28,2] Analysts must have experience in good URI design, popular vocabularies[19]. Additionally, analysts need to have some grasp of RDF patterns, such as PROV qualified associations and Semantic Science Integrated Ontology (SIO)[20], so they can understand how to more effectively anchor their RDF to existing linked data in more discoverable and recognizable ways.

In terms of development, we need to explore practical approaches for developing software that helps analysts perform five-star applications. Currently, most tools do not accept and generate RDF and, thus, it is up to analysts to wrap the tool in munges that conform to the five-star application munge space. The software engineering community is missing a suitable abstraction and set of requirements that can guide visualization developers to build tools that facilitate five-star usage.

Similarly, we can explore different representations for expressing a tool's inputs semantics. In addition to OWL, input semantics may be represented using SPARQL and even Java interfaces, provided the analyst's environment is supported by a toolbox of Java APIs. We are currently developing an API that allows visualization widgets to self-describe their input semantics using either representation. In fact, if a developer decides to document a tool's input semantics using SPARQL, the API will use that same SPARQL query to ingest data. This keep the semantics of the documentation *tight* with respect to implementation of the tool.

Our hopes is to invert the current asymmetry; instead of analysts spending the majority of their time preparing views, we would rather they spend their time observing and vetting an abundance of automatically-generated visualization options – determined by matching data with the input semantics of a tool. We want to put the analyst into a "visualization zombie apocalypse" and change the paradigm to focus on *eliminating* visualizations that are not desirable from the abundance of what can be instantly available.

8. Conclusion

We forged a Theory of Seamless Analytics that predicts the cost of non-trivial analyses that span multiple applications. The theory is a conglomerate of theories from the Visualization Analytics and Linked Data communities and explains analytical costs in terms of data evolution (i.e, Visualization Analytics theory) and data structuredness (i.e., Linked Data theory). As data evolves into ordered forms that facilitate analytic reasoning, it jumps within a dichotomous space of mundane and semantic formats. The theory suggests that when data occupies the mundane space, analyses will incur high costs.

Our theory was described in three parts: a Application Ontology (AO) that describes analytic applications regardless of the type of data, tool, or objective involved; a scoring metric to assess the cost of analyses

[19] Linked Open Vocabularies (LOV) maintains a listing of crowd sourced vocabularies `http://lov.okfn.org/dataset/lov/`

[20] `http://semanticscience.org/`

described in AO; and a set of cost reduction strategies that are expressed in the form of restrictions on AO. We demonstrate the utility of the theory by comparing the actual cost and predicted cost of two analyses: one real-world example based on the current state of practice and an alternative, hypothetical analysis that employs the cost reduction strategies.

References

[1] Tom Heath and Christian Bizer. Linked data: Evolving the web into a global data space. *Synthesis lectures on the semantic web: theory and technology*, 1(1):1–136, 2011.

[2] Max Schmachtenberg, Christian Bizer, and Heiko Paulheim. Adoption of the linked data best practices in different topical domains. In *The Semantic Web–ISWC 2014*, pages 245–260. Springer, 2014.

[3] James J Thomas and Kristin A Cook. *Illuminating the path: The research and development agenda for visual analytics*. IEEE Computer Society Press, 2005.

[4] Shixia Liu, Weiwei Cui, Yingcai Wu, and Mengchen Liu. A survey on information visualization: recent advances and challenges. *The Visual Computer*, pages 1–21, 2014.

[5] Michael Bostock, Vadim Ogievetsky, and Jeffrey Heer. D^3 data-driven documents. *Visualization and Computer Graphics, IEEE Transactions on*, 17(12):2301–2309, 2011.

[6] Sean Kandel, Jeffrey Heer, Catherine Plaisant, Jessie Kennedy, Frank van Ham, Nathalie Henry Riche, Chris Weaver, Bongshin Lee, Dominique Brodbeck, and Paolo Buono. Research directions in data wrangling: Visualizations and transformations for usable and credible data. *Information Visualization*, 10(4):271–288, 2011.

[7] Ghislain Auguste Atemezing and Raphael Troncy. Towards Interoperable Visualization Applications Over Linked Data. In *Talk Given at the 2nd European Data Forum (EDF), Dublin, Ireland (April 2013), http://goo. gl/JhVrax*.

[8] Timothy Lebo, Satya Sahoo, Deborah McGuinness, Khalid Belhajjame, James Cheney, David Corsar, Daniel Garijo, Stian Soiland-Reyes, Stephan Zednik, and Jun Zhao. Prov-o: The prov ontology. *W3C Recommendation, 30th April*, 2013.

[9] Sean Kandel, Andreas Paepcke, Joseph M Hellerstein, and Jeffrey Heer. Enterprise data analysis and visualization: An interview study. *Visualization and Computer Graphics, IEEE Transactions on*, 18(12):2917–2926, 2012.

[10] Timothy Lebo and Gregory Todd Williams. Converting governmental datasets into linked data. In *Proceedings of the 6th International Conference on Semantic Systems*, page 38. ACM, 2010.

[11] LOD2 Collaborative Project. Report on Knowledge Extraction from Structured Sources. Technical report, 2010.

[12] RaphaÃńl Troncy Gabriel Kepeklian and Laurent Bihanic. Datalift: A platform for integrating big and linked data. In *In International Conference on Big Data from Space (BIDS'14), Rome, Italy, November 12-14, 2014 (to appear)*, 2014.

[13] Natalya F Noy. Semantic integration: a survey of ontology-based approaches. *ACM Sigmod Record*, 33(4):65–70, 2004.

[14] Carlos Buil-Aranda, Aidan Hogan, Jürgen Umbrich, and Pierre-Yves Vandenbussche. SPARQL Web-Querying Infrastructure: Ready for Action? In *The Semantic Web–ISWC 2013*. 2013.

[15] Gary Klein, Jennifer K Phillips, Erica L Rall, and Deborah A Peluso. A data-frame theory of sensemaking. In *Expertise out of context: Proceedings of the sixth international conference on naturalistic decision making*, pages 15–17. Psychology Press, 2007.

[16] Peter Pirolli and Stuart Card. The sensemaking process and leverage points for analyst technology as identified through cognitive task analysis. In *Proceedings of International Conference on Intelligence Analysis*, volume 5, pages 2–4. Mitre McLean, VA, 2005.

[17] Daniel M Russell, Mark J Stefik, Peter Pirolli, and Stuart K Card. The cost structure of sensemaking. In *Proceedings of the INTERACT'93 and CHI'93 conference on Human factors in computing systems*, pages 269–276. ACM, 1993.

[18] Krzysztof Janowicz and Pascal Hitzler. The digital earth as knowledge engine. *Semantic Web*, 2012.

[19] Glenn A Fink, Christopher L North, Alex Endert, and Stuart Rose. Visualizing cyber security: Usable workspaces. In *Visualization for Cyber Security, 2009. VizSec 2009. 6th International Workshop on*, pages 45–56. IEEE, 2009.

[20] Manolis Savva, Nicholas Kong, Arti Chhajta, Li Fei-Fei, Maneesh Agrawala, and Jeffrey Heer. Revision: Automated classification, analysis and redesign of chart images. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, pages 393–402. ACM, 2011.

[21] Jacques Bertin. Semiology of graphics: diagrams, networks, maps. 1983.

[22] Martin G Skjæveland. Sgvizler: A javascript wrapper for easy visualization of sparql result sets. In *Extended Semantic Web Conference*, 2012.

[23] Ian Jacobs and Norman Walsh. Architecture of the world wide web. 2004.

[24] Timothy Lebo, Patrick West, and Deborah L. McGuinness. Walking into the future with prov pingback: An application to opendap using prizms (in press). In Bertram Ludaescher and Beth Plale, editors, *Provenance and Annotation of Data and Processes*, Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2014.

[25] Jans Aasman and Ken Cheetham. Rdf browser for data discovery and visual query building. In *Proceedings of the Workshop on Visual Interfaces to the Social and Semantic Web (VISSW 2011), Co-located with ACM IUI*, page 53, 2011.

[26] Tim Berners-Lee, James Hollenbach, Kanghao Lu, Joe Presbrey, and Mc Schraefel. Tabulator redux: Browsing and writing linked data. 2008.

[27] Tuukka Hastrup, Richard Cyganiak, and Uldis Bojars. Browsing linked data with fenfire. 2008.

[28] Aidan Hogan and Jü Umbrich. An empirical survey of linked data conformance.

[29] Mark D Wilkinson, Luke McCarthy, Benjamin Vandervalk, David Withers, Edward Kawas, and Soroush Samadian. Sadi, share, and the in silico scientific mehtod. *BMC bioinformatics*, 11:S7, 2010.

[30] Marko A Rodriguez. A graph analysis of the linked data cloud. *arXiv preprint arXiv:0903.0194*, 2009.

[31] Timothy Lebo, Alvaro Graves, and Deborah L McGuinness. Content-preserving graphics. In *COLD*, 2013.

[32] Stuart K Card, Jock D Mackinlay, and Ben Shneiderman. *Readings in information visualization: using vision to think*.

Morgan Kaufmann, 1999.

[33] Robert B Haber and David A McNabb. Visualization idioms: A conceptual model for scientific visualization systems. *Visualization in scientific computing*, 74:93, 1990.

[34] Ed Huai-hsin Chi and John T Riedl. An operator interaction framework for visualization systems. In *Information Visualization, 1998. Proceedings. IEEE Symposium on*, pages 63–70. IEEE, 1998.

[35] Ed H Chi. A taxonomy of visualization techniques using the data state reference model. In *Information Visualization, 2000. InfoVis 2000. IEEE Symposium on*, pages 69–75. IEEE, 2000.

[36] Heidi Lam. A framework of interaction costs in information visualization. *Visualization and Computer Graphics, IEEE Transactions on*, 14(6):1149–1156, 2008.

[37] Robert E Patterson, Leslie M Blaha, Georges G Grinstein, Kristen K Liggett, David E Kaveney, Kathleen C Sheldon, Paul R Havig, and Jason A Moore. A human cognition framework for information visualization. *Computers & Graphics*, 42:42–58, 2014.

[38] Jarke J Van Wijk. The value of visualization. In *Visualization, 2005. VIS 05. IEEE*, pages 79–86. IEEE, 2005.

[39] Sean Kandel, Andreas Paepcke, Joseph Hellerstein, and Jeffrey Heer. Wrangler: Interactive visual specification of data transformation scripts. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 3363–3372. ACM, 2011.

[40] Max Schmachtenberg, Christian Bizer, and Heiko Paulheim. Adoption of the linked data bset practices in different topical domains. In *The Semantic Web–ISWC 2014*, pages 245–260. Springer, 2014.