> ## SDRF (Scientific Data Repository Framework) Supplemental Archives
>
> ### *Improving Data Sets' Machine Readability and Interoperability with Published Research*

A New Protocol for Sharing Scientific Data

Linguistic Technology Systems (LTS) is developing SDRF to encompass data models, publishing guidelines, and code libraries for deploying open-access research data sets associated with scientific publications. Nowadays there are many general-purpose and domain-specific portals hosting scientific data; there are also several available formats for describing and encoding scientific data, such as Research Objects, **schema.org/Dataset**, Digital Curation Center (DCC), SCIDATA, BIOCODER, and MIBBI (Minimum Information for Biological and Biomedical Investigations). The purpose of SDRF is to merge these different data-set formats into a unified, overarching standard which can be adapted to different publishing houses and pipelines.

Shaping Protocols to Conform to Current Specifications in Publishing

In order to conform to current specifications — such as FAIRsharing or the Bill and Melinda Gates Foundation guidelines for authors (https://gatesopenresearch.org/for-authors/data-guidelines) — proper protocols must be implemented at several stages of the publishing process, in particular (1) publications should provide clear descriptions of accompanying data sets and where that data is hosted; (2) publication repositories should make data-set links and important metadata clearly visible on web pages where documents are read or previewed; (3) data sets themselves need to include metadata and supporting files which help researchers properly access, visualize, and reuse the data; and (4) data sets need to be connected with software which has the correct features to load and display the relevant raw data files. SDRF will include technology applicable to each of these four facets of the publishing and data-sharing pipeline in order to conform to current standards.

It is important to emphasize that data sets are only truly valuable if they are machine-readable and seamlessly integrated into domain-specific software ecosystems. Scientists who examine and reuse published data sets are generally researchers doing technical work in a field closely related to that of the original authors; in many cases there are specialized software applications, computational methods, algorithms, and research protocols which are endemic to the relevant subject areas. When sharing research data, accordingly, publishers/authors should make it as easy as possible for scientists to examine the data within the digital ecosystem that they utilize for their own research. This often implies that document viewers — e.g., PDF viewers and/or HTML pages on publisher portals — should be ideally interconnected with scientific applications so that scientists, when reading books/articles, can seamlessly launch domain-specific software and visualize/examine associated data sets. Unfortunately, most scientific software does not incorporate code libraries to parse metadata (summarizing file types, download instructions, etc.) describing open-access data sets. This can be addressed by providing plugins or extensions that add data-set-accession capabilities to existing scientific software, so that publisher repositories and data-hosting repositories can be made truly interoperable with scientific applications.

To demonstrate how such plugins can work, as well as other facets of the data-publication process facilitated via SDRF, this paper will review two case-studies addressing research in the academic

3.2. Data preprocessing and Feature extraction

Since the data is extracted using different signal processing methods, it ranges diversely. This contributes to inadequate learning procedures. Consequently, to get started with the task, we apply rescaling or in a more common term, min-max normalization. Using this method, the data is scaled in a specified range, and here we scale the features to the [0, 1] range.

Table 1: A summary of reviewed datasets.

Data type	Description	Study
Brain MRI	In this retrospective study, we enrolled 56 patients and 28 healthy control subjects.	[9]
GAIT	This database contains measures of gait from 93 patients with idiopathic PD (mean age: 66.3 years; 63% men), and 73 healthy controls (mean age: 66.3 years; 55% men).	[10]
GAIT	303 subjects were recruited from the "Incidence of Cognitive Impairment in Cohorts with Longitudinal Evaluation-GAIT" (ICICLE-GAIT) study.	[11]
Vocal Features	UCI Parkinson's Disease Classification.	[13]
Vocal Features	The dataset range of biomedical voice measurements from 31 people, where 23 people are showing Parkinson's disease. -UCI Parkinson Speech Dataset with Multiple Types of Sound Recordings Data Set	[14]
Vocal Features	The database consisted of 23 columns and 197 rows. The dataset was created by Max Little of the University of Oxford, in collaboration with the National Centre for Voice and Speech, Denver, Colorado, who recorded the speech signals. This dataset is composed of a range of biomedical voice measurements from 31 people, 23 with Parkinson's disease (PD). Each column in the table is a particular voice measure.	[15], [17]

The authors provide citations for data sets used in their analyses ...

Figure 1: Table Listing Analyzed Data Sets in the Parkinson's Article

literature, each involving papers that have been linked with multiple data sets (data which was either reused or newly created during the course of the research described).

First Case Study: "Parkinson's Disease Diagnosis: The Effect of Autoencoders on Extracting Features from Vocal Characteristics," by Ashena Gorgan Mohammadi, Pouya Mehralian, Amir Naseri, and Hedieh Sajedi (International Journal of Speech Technology, pending review)

This case study demonstrates a scenario where an article reuses multiple pre-existing data sets. The article examines Parkinson's Disease symptomology from multiple perspectives, including gait (loss of motor function), speech impairment, and bioimaging (MRIs). The authors apply Machine Learning to data sets focused on these three different diagnostic areas, in an effort to advance research to refine Parkinson's predictors and diagnoses. The overall information can be summarized as follows:

1. **Gait Data**: this data was primarily drawn from a PhysioNet data set (https://physionet.org/content/gaitpdb/1.0.0/) obtained via sensors attached to subjects' feet as they walked (the study includes both Parkinson's patients and healthy controls). This sensor data is provided as a collection of text files, each file corresponding to one patient (or control subject), with each line in a file representing a single time snapshot. The lines are divided into space-separated columns, each representing force exerted on a single sensor, plus two additional columns calculating total force on the left-foot and right-foot sensors respectively. This data set also includes demographic and clinical information for each patient in a spreadsheet format. The authors also use an additional source of gait data derived from a more recent (2019) study whose data is available only upon request (see https://www.nature.com/articles/s41598-019-53656-7#MOESM1).

2. **Speech Data**: this data was primarily drawn from a data set hosted by the University of Cali-

fornia Irvine Machine Learning Repository (https://archive.ics.uci.edu/ml/datasets/Parkinsons). The central information is a **CSV** file, where each line represents a single voice recording from one of 31 subjects (consisting of 23 Parkinson's patients and 8 healthy controls). Each subject made multiple recordings. The individual lines in the **CSV** file present a quantitative model of subjects' speech via a collection of acoustic features/attributes. A similar "telemonitoring" data set from the same archive was used to study how the analysis of Parkinson's-related speech data may be applicable to samples obtained via devices such as smartphones.

3. **Radiological Data**: this data is not immediately available for reuse, but requires special (non-commercial) authorization from the Parkinson's Progressive Markers Initiative (https://www.ppmi-info.org/) or making a request of corresponding authors of referenced papers introducing the relevant data (https://www.frontiersin.org/articles/10.3389/fnins.2019.00874/full#h6).[1] Although this data is derived from bioimaging (so that the underlying raw data files are radiological) the authors of the IJST paper under review utilize the data in a more structured form, building off of image-feature extraction already performed when the data sets were first published by the prior authors. However, the republished unified data set itself (to appear in conjunction with the IJST submission) might include code allowing researchers to reproduce the original analytic workflow if desired: for instance, among other things, we can enable the implementations published via https://biomedia.doc.ic.ac.uk/software/malp-em/ to be embedded as a **CAPTK** module (see in particular https://cbica.github.io/CaPTk/tr_integration.html#tr_cppIntegration).

4. **Python Code Repository**: the authors also provide Python code, hosted on GitHub, which they used to analyze these various data sources.

[8] Sakar, C. Okan, Gorkem Serbes, Aysegul Gunduz, Hunkar C. Tunc, Hatice Nizam, Betul Erdogdu Sakar, Melih Tütüncü, Tarkan Aydin, M. Erdem Isenkul, and Hulya Apaydin. "A Comparative Analysis of Speech Signal Processing Algorithms for Parkinson's Disease Classification and the Use of the Tunable Q-factor Wavelet Transform." *Applied Soft Computing* 74 (2019): 255-63. doi:10.1016/j.asoc.2018.10.022.

[9] Salvatore, C., A. Cerasa, I. Castiglioni, F. Gallivanone, A. Augimeri, M. Lopez, G. Arabia, M. Morelli, M.c. Gilardi [...] "[...] quantitative Machine Learning on Brain MRI Data for Differential Diagnosis of Parkinson's Disease and Progressive Supranuclear Palsy." *Journal of Neuroscience Methods* 222 (2014): 230-37. doi:10.1016/j.jneumeth.2013.11.016.

[10] "Gait in Parkinson's Disease." Gait in Parkinson's Disease V1.0.0. February 25, 2008. https://physionet.org/content/gaitpdb/1.0.0/

[11] Rehman, Rana Zia Ur, Silvia Del Din, Yu Guan, Alison J. Yarnall, Jian Qing Shi, and Lynn Rochester. "Selecting Clinically Relevant Gait Characteristics for Classification of Early Parkinson's Disease: A Comprehensive Machine Learning Approach." Scientific Reports 9, no. 1 (2019). doi:10.1038/s41598-019-53656-7.

[12] Goetz, Christopher G. "The History of Parkinson's Disease: Early Clinical Descriptions and Neurological Therapies." *Cold Spring Harbor Perspectives in Medicine*, Cold Spring Harbor Laboratory Press, Sept. 2011.

[13] UCI Machine Learning Repository: Parkinsons Disease Classification Data Set. https://archive.ics.uci.edu/ml/datasets/Parkinson's Disease Classification.

[14] Sriram, Tarigoppula & Rao, M. & Narayana, G & Vital, T. & Dowluru, Kaladhar & GK. (2013). Intelligent Parkinson Disease Prediction Using Machine Learning Algorithms. IJEIT. 3. 212-215.

[15] R. Das, "A Comparison of multiple classification methods for diagnosis of Parkinson disease." Expert Systems with Applications, vol. 37, no. 2, pp. 1568-1572, 2010.

[16] Erdogdu Sakar, Betul et al. "Analyzing the effectiveness of vocal features in early telediagnosis of Parkinson's disease." *PLoS one* vol. 12,8 e0182428. 9 Aug. 2017, doi:10.1371/journal.pone.0182428

[17] M. Peker, B. Şen, D. Delen, Computer-aided diagnosis of Parkinson's disease using complex-valued neural networks and mRMR feature selection algorithm, J. Healthcare Eng. 6 (3) (2015) 281–302

[18] Ahlrichs, Claas, and Michael Lawo. "Parkinson's Disease Motor Symptoms in Machine Learning: A Review." Health Informatics - An International Journal 2, no. 4 (2013): 1-18. doi:10.5121/hiij.2013.2401.

[19] Khoury, Nicolas, Ferhat Attal, Yacine Amirat, Abdelghani Chibani and Samer Mohammed. "CDTW-based classification for Parkinson's Disease diagnosis." ESANN (2018).

[20] Brooks, David J. "Neuroimaging in Parkinson's Disease." NeuroRX 1, no. 2 (2004): 243-54. doi:10.1602/neurorx.1.2.243.

[21] Mohammad, Roohi, and Fatima Mubarak. "Neuroimaging in Parkinson Disease." Parkinson's Disease and Beyond - A Neurocognitive Approach, 2019. doi:10.5772/intechopen.82308.

[22] A. Kazeminejad, S. Golbabaei and H. Soltanian-Zadeh, "Graph theoretical metrics and machine learning for diagnosis of Parkinson's disease using rs-fMRI," 2017 Artificial Intelligence and Signal Processing Conference (AISP), Shiraz, 2017, pp. 134-139.

[23] Shiiba T, Arimura Y, Nagano M, Takahashi T, Takaki A. "Improvement of classification performance of Parkinson's disease using shape features for machine learning on dopamine transporter single photon emission computed tomography." PLoS ONE 15(1): e0228289, (2020). doi:10.1371/journal.pone.0228289.

[24] Xu, Jiahang, Jiao, Huang, Yechong, Luo, Xu, Qian, Li, Ling, Liu, Zuo, Wu, Ping, and Xiahai. "A Fully Automatic Framework for Parkinson's Disease Diagnosis by Multi-Modality Images." Frontiers. August 05, 2019.

[25] Ting, Jiang, Lin, Wei, Wu, Ping, Zhou, Yongjin, Zuo, Wang, Jian, Yan, Zhuangzhi, Shi, Kuangyu,

> **Some data sets are directly available through the bibliography; others have to be located by reading the cited articles.**

Figure 2: Bibliography (With Data Set Hyperrefs) in the Parkinson's Article

---

[1] When the unified data set is published, we will request permission to include a copy of the original data set to spare future readers from having to request this data on their own.

Unifying the Data Sets into a Single Package

In this artice pending review, the authors summarize the data sets in a table within the main text (see Figure 1 here) and, in their bibliography, they cite these data sets either directly or by referencing publications where the relevant data sets are described (see Figure 2 here). Doing so properly credits authorship to the researchers who curated the data sets, and it gives readers a means to locate their raw data. However, accessing and working with that raw data is inconvenient from a reader's point of view without most or all of the data sets being repackaged into a *single* archive that could be hosted and downloaded as one unit. Obtaining raw data from the resources identified in the bibliography requires several steps — for instance, the PhysioNet sensor data can be downloaded as a zipped folder, while the demographic data attached to it has to be downloaded separately. Furthermore, some of the information obtained from **MRI** and speech analysis (reported in papers cited as data sources for the IJST submission) is provided as supplemental materials within the secondary papers; this requires readers to browse one at a time through each of the relevant articles so as to find downloadable links (see Figure 4). In short, piecing all the source data together puts the onus on readers to manually inspect multiple web resources and to manually interconnect a variety of files once they are downloaded.

Another complicating factor is that certain information present in the data sets is implicit within how the data sets are organized, requiring extra effort to extract this information in a machine-readable manner. For instance, the PhysioNet sensor data uses a file-naming convention which encodes several pieces of information in the file names, such as whether the file presents a Parkinson's patient or a control subject (see Figure 3). Though by examining file names it would be possible to construct a table with additional information providing context for the file contents, such information is not directly included within the PhysioNet data set; it needs to be extracted by computer code.

Constructing Machine-Readable Supplemental Archives

This collection of data sets serves as an example of how technologies such as **SDRF** can fill the gap between publication/data repositories and scientific computing, making scientific data more "**FAIR**" (Findable, Accessible, Interoperable, Reusable). Upon publication of this submission in IJST (pending peer review) the disparate open-access data from the article's secondary sources would be provided as a single **SDRF** archive. This archive would provide *machine-readable* access to information spread across multiple sources, translated into a common file format. In general, **SDRF** encourages and implements features to help data sets conform to **FAIR** and related standards, such as (1) bundling multiple data sets into a single archive; (2) migrating data to general-purpose representations wherever possible — formats such as **XML**, **HDF5**, **ARFF**, or **DICOM**; (3) providing meta-data in multiple formats (**DCC**, **schema.org/Dataset**, **MIBBI**, **BIOCODER**, etc.) to be compatible with different organizations' platforms; (4) identifying one or more "preferred applications" for examining/reusing the published data; (5) explicitly representing information encoded via file-names; (6) bundling raw data, meta-data, and (where possible) machine-readable article text into a single resource, which **SDRF** calls a "Supplemental Archive;" and (7) annotating the data sets to support microcitations that granularly link the publication to its associated Supplemental Archive.

Once a Supplemental Archive has been downloaded, an important question for any **SDRF** archive is how researchers will productively access the data. Unlike the Flow Cytometry use-case discussed below, the Parkinson's archive spans several scientific disciplines; as such, there is no obvious application which could be preferred by default for examining the data files. As a fallback option, **SDRF** is designed to present data sets via **QT** Creator, a **C++** Integrated Development Environment associated with the **QT** application-development framework. **SDRF** includes code libraries to represent research meta-data as **C++** objects; these libraries can be opened as **QT** projects. These may be supplemented with separate libraries extracting and managing information specific to individual data sets. In particular, the Supplemental Archive for the Parkinson's article under peer review would provide
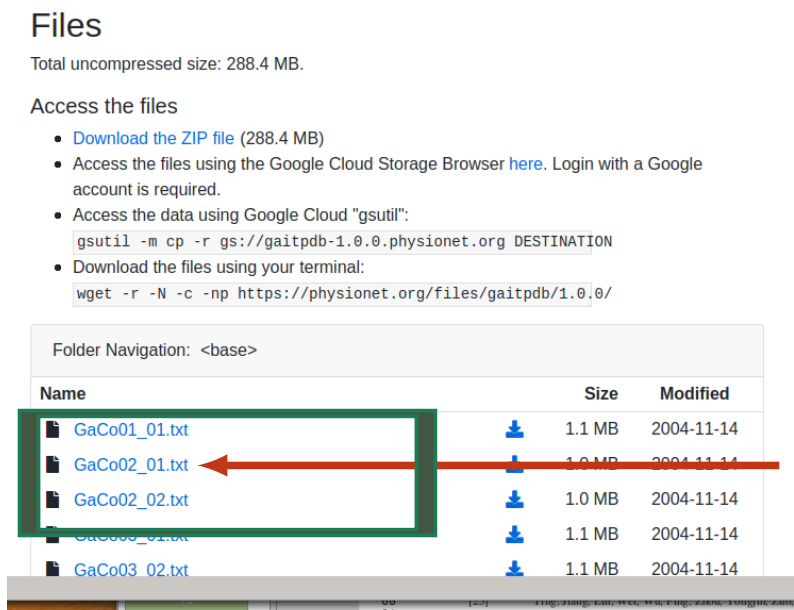
Figure 3: Extracting Information Encoded in File Names

C++ classes encapsulating spreadsheet-like data (whether originally in **.xls**, **CSV**, or space-delimited formats) republished by the Journal in the unified data set.

An additional concern for SDR**F** archives is how to properly annotate publications and data sets side-by-side. In the Parkinson's article, individual **C++** classes encapsulating tabular data serve as convenient microcitation targets: annotations within the relevant **C++** code represent anchors through which the data set may be referenced (on a more precise scale than merely citing the Supplemental Archive as a whole). In some places, individual class attributes can also be linked to lines in the authors' Python source code. On the text side, certain paragraphs within the Parkinson's article could be linked to the corresponding **C++** code annotations. This illustrates SDR**F**'s recommended annotation/microcitation system, where segments in publication texts (identified for instance via LaTeX **phantomsection** commands or **JATS** **statement** tags) are joined to annotations or comments in code and/or raw data files in the Supplemental Archive.

Second Case Study: "Marked T cell activation, senescence, exhaustion and skewing towards TH17 in patients with COVID-19 pneumonia" from nature.com, 2020

This article presents a use-case with some noteworthy contrasts to the Parkinson's publication described in the previous section. This Covid-19 paper (https://www.nature.com/articles/s41467-020-17292-4) was published along with two data sets comprising Flow Cytometry Standard (**FCS**) files hosted via the Flow Repository (http://flowrepository.org/id/FR-FCM-Z2N4 and http://flowrepository.org/id/FR-FCM-Z2N5). Links to the data sets (via **flowrepository.org** pages) are explictly provided in the publication's "Data Availability" section. However, researchers still need to perform several steps to manually download the full set of relevant **FCS** and meta-data files.

One feature of this second use-case is that the technical information in the data sets belong to a single scientific area (Flow Cytometry) and are mostly encoded in a single format (**FCS**). As such, it is straightforward to identify the kind of software which researchers need to use to visualize the raw data — basically, any application that can parse **FCS** files. There are a variety of commercial as well as open-source Flow Cytometry (**FCM**) applications which can be used to access **FCS** data. Once readers have downloaded the Flow Repository archives, they may individually load the **.fcs** files to study data which, in the original article, is summarized via figure illustrations.

This workflow nonetheless requires researchers to perform several manual steps before being able to use the published data. The core problem is that existing Flow Cytometry software does not intrinsically have capabilities to read **PDF** files, locate **FCS** data sets, and interoperate with hosting
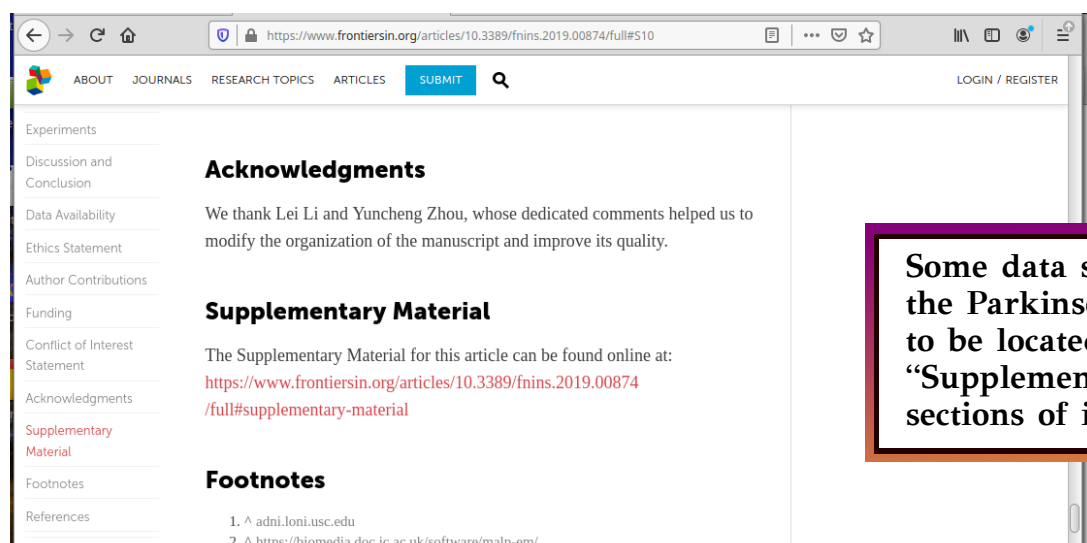
Figure 4: Indirectly Locating Data Sets from Cited Papers

platforms such as Flow Repository. Employing this use-case as an example with which to illustrate proper alignment between document viewers, publication/data repositories, and scientific software, we are developing a new Flow Cytometry application which *can* interoperate with **PDF** viewers and **SDRF** archives. This application is designed so that, when authors are reading a **PDF** file associated with an **FCS** data set, the **PDF** viewer can automatically launch and signal to the **FCM** application when a reader wishes to download and visualize **FCS** files. In short, the **FCM** application — having received data from a **PDF** viewer which implements an **SDRF** inter-application protocol — will automatically execute download and extraction steps that scientists otherwise would have to perform manually.

Note also that, although most of the relevant data for this Covid-19 article is in **FCS** form, there is, in addition, supplemental clinical information provided in other formats (e.g., spreadsheets). For cases such as these, the LTS **FCM** software includes code libraries allowing researchers to parse non-**FCS** data in standard formats such as **XML**, **HDF5**, or **ARFF**.

This use-case illustrates a general principle: that research data is most convenient for scientists when it is deployed within an infrastructure where portals, document viewers, and scientific applications can seamlessly interoperate. Wherever possible, when researchers are reading published books or articles, they should be able to *automatically* launch the proper scientific application, download data sets, and examine raw data files in the preferred software with only one or two clicks. These steps should be performed without user intervention/involvement as much as is feasible, instead of readers having to waste time with manually finding, downloading, merging/extracting, and then opening data files.

Conclusion

The two case-studies considered here are similar in that each involve articles which are linked to multiple data sets. For maximum convenience, it is optimal for researchers to be able to access this data without performing manual download and merging/extracting actions. There are also some differences between the two case-studies: in particular, the Parkinson's data spans multiple disciplines, whereas the Covid-19 data is more rigorously grounded in Flow Cytometry. As such, the operational requirements for the Covid-19 data, from a reader's point of view, are more clearly delineated: effective integration between the publication and its accompanying data sets is defined by launching Flow Cytometry software while a researcher is reading the publication, allowing the researcher to view the data via software similar to that used to generate/analyze the data while the reported research was being conducted. In the case of the Parkinson's data, in contrast, there is no single application that would seamlessly display the spectrum of information considered in that article; as mentioned above, in such cases, **SDRF** would default to using **QT** Creator as a fallback for loading Supplemental Archives where no other software is available.
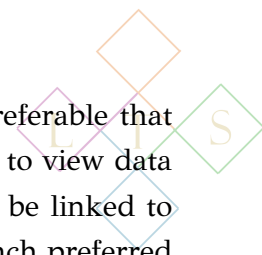
Regardless of whether one is using QT Creator or a domain-specific application, it is preferable that each SDRF archive be associated with one or more applications that researchers can use to view data (and extract information) from the archive. Moreover, these applications would ideally be linked to document viewers and also to publisher's portals, so that readers can automatically launch preferred applications and view Supplemental Archives while reading concomitant publications. In order to achieve this, scientific applications need to be augmented with plugins to parse SDRF data. To address this need, we are developing an inter-application messaging protocol so that disparate applications with SDRF plugins can interoperate. In particular, this protocol would entail PDF viewers being able to interoperate with scientific applications so that publications' data sets may be automatically downloaded and visualized via the preferred software.

Our prototype example for an application utilizing such plugins, as mentioned above, is software for Flow Cytometry. We are also working on a prototypic QT Creator plugin so that QT Creator (as the "default" SDRF software) can participate in SDRF networks using the same protocol. We will then expand the scope of this protocol via plugins for software in other domains, such as image-analysis, molecular visualization, radiology, 3D graphics, and so forth.

Future Projections: Operationalizing SDRF

This section will present a concrete outline of the steps necessary to package scientific data into an SDRF Supplemental Archive. Data Sets may be published with content and organization compatible with both SDRF and other formats, such as Research Object Bundles, so that using SDRF does not preclude adopting other formats as well. For example, a data set which provides SDRF meta-data may also include the **meta-inf** files required by Research Objects. Indeed, it is recommended that authors aim for compatibility with multiple standards, not only SDRF. This section, however, will focus on the components of Supplemental Archives that are specific to SDRF.

In this section we describe two different approaches to using SDRF. One approach employs SDRF to a limited extent, deferring to older technologies for such basic operations as encoding data or annotating publications. An alternative is to adopt SDRF more holistically: adopting experimental or under-development features in the SDRF libraries. When discussed below, these features will usually be characterized as *experimental* or *specific to SDRF code libraries*.

Formulating Scientific Data Repository Models for Individual Publishing Environments

SDRF is intended to interoperate with multiple publisher and data-hosting platforms, each of which have their own requirements and technology. Therefore, the precise contents and organization of a given SDRF Supplemental Archive will depend on specifications pertaining to the particular repository and hosting platform where the publication and data set, respectively, are to be deposited. To facilitate Supplemental Archive preparation for different environments, SDRF seeks to incorporate "Scientific Data Repository Models" (SDRMs) which are tailored to individual publishers. These SDRMs would encapsulate information (via application plugins and/or code libraries) targeted to the specific platforms where the relevant publication and data set are hosted. For example, research funded by the Bill and Melinda Gates Foundation must follow certain guidelines, which are documented whenever an article is submitted to the Gates Open Research portal. Consequently, the specific SDRF meta-data for such papers can be designed to notate how the associated research adheres to these organizational guidelines. In this type of situation, the layout and vocabulary for SDRF meta-data would be determined at least in part by the SDRM germane to the environment where the research is published.

The purpose of an SDRM is also to facilitate the implementation of technologies that interoperate with publishers' platforms — for instance, API clients which can request information from document/data portals (e.g., locating articles or data sets via keyword searches), or PDF viewers enhanced with capabilities to obtain data sets from data hosting platforms (to automatically download data sets
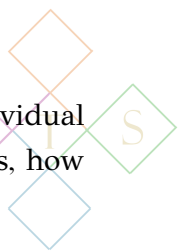
upon request from a user reading the corresponding publication). For these use-cases, individual SDRMs would include information and/or computer code about how to access publishers' APIs, how to download and extract a data set given a digital identifier, and so forth.

Because SDRF Supplemental Archives may differ somewhat according to the specific SDRM in effect, any description of these archives is provisional. To be precise, SDRF assumes a generic SDRM developed by LTS — dubbed "MOSAIC" ("Multiparadigm Ontologies for Scientific/Academic and Technical Publications") — which can be extended or modified in consultation with individual publishers, as required. As such, what is described below will apply intrinsically only to the MOSAIC SDRM; depending on the platform, the details for individual SDRF Supplemental Archives may be different than the generic model presented here.

Standard Components of SDRF Supplemental Archives

The components of an SDRF Supplemental Archive can be grouped into several facets: meta-data, raw data, and machine-readable text.

1. SDRF Meta-Data

Meta-Data Files   SDRF uses a vocabulary for describing the contents of data sets which merges the data models of several existing formats, such as DCC and **schema.org/Dataset**. SDRF recommends encoding this data in TAGML ("Text-as-Graph Markup Language"), which is a very flexible and computationally powerful representation format.[2] The SDRF-specific libraries include parsers for (an extended version of) TAGML.

C++ Files   SDRF also recommends that authors provide C++ code to initialize objects representing Supplemental Archives' meta-data. These objects may be directly constructed from the TAGML meta-data files, or (if desired) authors may choose to customize the C++ logic. For lab-based research, authors may employ BIOCODER, which adopts C++ code to notate research protocols and workflows (the SDRF libraries include a modified version of BIOCODER). Moreover, for data sets whose "preferred application" for opening raw data files is implemented in C++ (as is the case for many, if not most, scientific-computing applications), the Supplemental Archive may include plugins or extensions to these applications. In general, using C++ objects specific to Supplemental Archive meta-data as a basis, authors (or programmers coding on authors' behalf) may introduce additional C++ code demonstrating or enabling analysis, visualization, or application-integration for the Supplemental Archive's raw data.

SDRM-Specific Meta-Data   Using either data files or computer code, information or capabilities specific to the platforms where papers and Supplemental Archives are hosted can be presented in conjunction with the applicable SDRMs. For instance, C++ code distributed with the archive may include procedures for accessing APIs of the hosting service where the archive is deposited.

Qt Project Files   As explained earlier, SDRF defaults to using QT Creator as an application with which to examine data sets, if there is no obvious "preferred application" according to the research topic. Therefore, SDRF recommends that the C++ meta-data code be paired with QT project files and other assets necessary to run the code in the QT Creator IDE (Integrated Development Environment), using **qmake** as the build system. In general, there should be a QT project specific to the SDRF meta-data; in some cases, there would also be separate QT projects for working with raw data files.

2. Raw Data   As mentioned above, SDRF recommends that raw data be encoded in general-

---

[2]TAGML is "powerful" in the sense that, with suitable mappings between their disparate syntactic structures, TAGML represents a superset of XML and other common data-representation languages, such as JSON.

purpose formats such as **ARFF**, **HDF5**, **XML**, or **TAGML** (as compared to formats such as **.xls**, which are associated with specific applications). An experimental "Hypergraph Exchange Format" (**HGXF**), being developed by LTS in conjunction with S<small>DR</small>**F**, or a related query format discussed in Appendix II, may also be used.

An exception to the guidelines for general-purpose formats is when data should be presented in optimized formats endemic to a certain scientific field, such as **FCS** files for Flow Cytometry or **DICOM** archives for bioimaging. In these contexts, it is recommended to employ such formats for most of the raw data files but also to construct summaries of the data (to facilitate properly loading and accessing the domain-specific files) utilizing a more general-purpose format.

3. **Annotated Manuscripts** Where authors have permission to share full-text versions of their books/articles, S<small>DR</small>**F** recommends that a machine-readable representation of these publications, in formats such as **TAGML** or **XML**, be included in the Supplemental Archive (S<small>DR</small>**F** has an experimental "Hypergraph Text Encoding Protocol" which could be adopted as well). This machine-readable manuscript may then be annotated and cross-referenced with raw data and/or code files also included in the Supplemental Archive. An experimental S<small>DR</small>**F** L^AT_EX package provides an annotation framework that not only marks locations in document text, but also encodes **PDF** viewport coordinates for use by specialized **PDF** viewers which implement an S<small>DR</small>**F** protocol for integrating **PDF** viewers with scientific applications.

4. **Representations of Laboratory Protocols and/or Computational Workflows** Formal representations of research protocols and workflows facilitate reuse of the data set and/or replication of the research work, as will be analyzed further in the next section.

Documenting Research Protocols

A systematic outline of research methods serves several different purposes, depending on the applicable scientific field. In some cases, documenting laboratory protocols promotes replication of the research and/or methodology used — in the physical environment of a laboratory, reproducing a lab protocol is dependent on precise configurations of measurements and equipment that often cannot be gleaned from the text of a research paper alone. This is the motivation for protocol-description formats such as those gathered under **MIBBI**: when seeking to reproduce the methods discussed in an article, lab technicians consult **MIBBI** files for technical information about the experimental setup (rather than the article text).

The **B<small>IO</small>C<small>ODER</small>** library supplies a variation on this form of laboratory-based protocol-description, because although **B<small>IO</small>C<small>ODER</small>** procedures generate protocol-description files, the actual protocol definition is constructed via procedural (specifically, **C++**) computer code. That is to say, **B<small>IO</small>C<small>ODER</small>**-defined protocols can be examined on two levels: through descriptions (in formats such as **HTML**) which are generated after **B<small>IO</small>C<small>ODER</small>** is executed, or through the actual **C++** code where the steps and requirements of the protocol are set forth in granular detail. Employing **C++** as a modeling language allows protocol-descriptions to be construsted in an expressive manner, utilizing programming formations (such as branching statements, objects, template/generic programming, and so forth) to represent protocol details. Modelers, as such, have the full expressiveness of a procedural programming language to work with (rather than the more restricted expressiveness of a markup language). At the same time, **B<small>IO</small>C<small>ODER</small>**'s libraries are equipped to generate (for example) **HTML** files by executing the **C++** code wherein a protocol is defined; consequently, the protocol is also expressed in human-readable form, so that consulting the underlying **C++** programming is not prerequite for studying the protocol described.

The S<small>DR</small>**F** code libraries include a modified version of **B<small>IO</small>C<small>ODER</small>** intended to expand the range of environments where **B<small>IO</small>C<small>ODER</small>** may be utilized. The key additions include:

1. The S<small>DR</small>**F** port of **B<small>IO</small>C<small>ODER</small>** uses a more modern **C++** programming style — for example, with **QT**

and **STL** (Standard Template Library) containers/algorithms instead of **C** arrays. This approach makes it easier to use **BioCoder** in conjunction with other **C++** libraries for **PDF**, LᴬTᴇX, graph visualization, application networking, and so forth.

2. Allowing templates and abstract base classes to be established as an interface for **BioCoder**-style workflow classes. Apart from concrete protocols, there can also be "protocol models" which reflect different investigative modalities — e.g., protocols for Machine Learning need a different vocabulary than protocols for lab experiments. Existing **BioCoder** global functions could in such contexts be mapped to class methods in this more general framework, so e.g. **start_protocol(...)** would become **dspin.start_protocol(...)** (here **D-SPIN** refers to an under-development **SDRF** model for image analysis).

3. Generating protocol-description files in multiple formats (not only **HTML**), and integrating protocol models with publishing software. For instance, **BioCoder**'s description generators can (depending on the **BioCoder** template selected) be extended to output LᴬTᴇX, **TAGML**, or publisher-specific **XML** dialects. Moreover, each step in a described protocol could be assigned a unique identifier that could be used to create a unique LᴬTᴇX label, permitting locations in the text of publications associated with the research work to be annotated with links to the protocol description.

4. Extending **BioCoder**'s graph functionality, using a Qt-compatible library to visualize the runtime graph, with a convenient interface for manipulating the graph structure with modern **C++**. Insofar as protocol models can be tailored to different scientific fields, different protocols would also call for different semantics for runtime graphs — that is, different property/edge schemata as would be modeled by a graph database, with edge and node-property annotations defined according to discipline-specific terminology.

5. Establishing a framework for aligning protocol models with standardized workflow vocabularies and "minimum information" checklists, such as **MIBBI**. For runtime graphs, such ontologies could provide the domain of values for edge and node-property annotations. For protocol-description code, the individual methods/functions for a protocol model could be decorated with annotations asserting that a particular function notates a research step which has a particular label in an **MIBBI** (or similar) recommendation. One could, for instance, maintain a dictionary of links between **C++** function/method pointers and label names formalized in an **MIBBI**-style controlled vocabulary.

Research Protocols and Computational Workflows

Technologies such as **BioCoder** and **MIBBI** are primarily adopted to construct human-readable summaries of research protocols. A related use-case involves explicitly notating workflows for "*in silico*" experiments or any other research which depends on digital analysis or processing. For example, investigations which depend on image-analysis may implement analytic routines wherein sample images are subject to a sequence of transformations, with a modified image produced after each step becoming input for the next step. With proper imaging software, researchers could reproduce such analyses explicitly — whether to verify the accuracy of the published analysis or to better understand the published research by visualizing its intermediary stages. In these situations, workflow descriptions are not only human-readable guidelines for re-enacting a certain investigative modality; they can be formal computational artifacts which orchestrate the execution of implemented procedures so as to allow the original computations to be repeated in different environments.

Researchers will sometimes share their analytic code in the same spirit of transparency and reusability as that which motivates sharing raw data. At a minimum, such sharing need entail only including source-code files within a data set and/or linking data sets to version-controlled archives (on GitHub, for instance, or a similar code-hosting site). However, computer code can be more tightly integrated with their containing Supplemental Archives by including code annotations and/or documentation which concretely ties programming elements (e.g. procedures, types, object member fields, or indi-

vidual typed values) to components and concepts that can be cited in the data set (and annotated in a corresponding publication). In some cases, such annotations may serve to explain why a given algorithmic implementation or sequence of operations is chosen, yielding an explication of digital protocols similar to those documented (in the lab context) via BIOCODER or MIBBI. The "PANDORE" project, for instance — a suite of image-processing utilities (called "operators") that can be combined into compound workflows — introduces an "Image Processing Objectives" ontology yielding systematically defined lexical notation for asserting the purpose of different steps in an image-processing pipeline. Workflows modeled via PANDORE may be directly executed via imaging software, and they may also be studied as human-readable emendations clarifying why the shared image-processing analyses are composed as they are.

SDRF encourages authors to utilize systematic annotation systems spread across text, code, and data as much as is possible. The details of annotation semantics depends on articles' applicable scientific field, so it is difficult to present a general analysis of how such annotations might be constructed (although some disciplines have existing standard vocabularies that can be drawn upon for source-code annotations, including the PANDORE ontology and specification within MIBBI, to mention examples cited here). LTS is also working on a more cross-disciplinary, but experimental, mechanism for unifying code and text annotations as part of THQL (discussed below).

Checklist for Authors, Editors, and Publishers

To clarify the steps needed to deploy publications and data sets according to SDRF recommendations, this section will outline the steps that would be taken by individuals fulfilling different roles in the publishing pipeline.

Authors   In most publishing environments, the process of depositing data sets for open-access hosting is entirely separate from that of submitting articles for peer-reviewed publications. Therefore, it is the responsibility of authors to ensure that these two resources — their manuscripts and Supplemental Archives — are properly interconnected. Some guidelines for ensuring that this process is carried out smoothly include: (1) cite the URL for the data set near the top of the manuscript submission; (2) cite the URL for the publication in a **readme** or similarly prominent location in the data set; (3) if there are no access restrictions, include a PDF file showing the full article in an easily findable location in the data set; (4) use labels and **phantomsection** or similar tags/commands to mark locations in the manuscript which are conceptually linked to parts of the data set; and (5) request that authors who reference the current manuscript cite both the publication URL and the Supplemental Archive URL when referencing the body of research.

Additional steps are also necessary when packaging files into the published archive. The exact details will depend on which SDRMs apply to the environments where the publication and data set are hosted, and also on whether authors wish to comply with research standardizations other than those of SDRF. For example, archives which are published as a Research Object Bundle need to use a folder layout proscribed by that specification. To work flexibly with other standards, SDRF does not demand *a priori* that authors follow any given conventions for file and folder names and hierarchies. In general, to pin the location of SDRF meta-data, SDRF recommends the following: C++ classes and **main.cpp** files specific to initializing SDRF meta-data should be marked accordingly with C++ comments; and files in formats such as TAGML, serializing SDRF meta-data, should be similarly identified. Once the meta-data source is locatable within the archive, all other information needed to fully parse the SDRF-specific data should be extracted from the initialized meta-data objects.

Editors   Because (as just outlined) an author's submissions to data-hosting portals are usually operationally separate from their article submissions, publication editors ordinarily cannot directly influence the author's preparation for publishing their data sets. However, editors can make

recommendations and double-check that the author's research (and other cited research within the article under submission) is properly referenced, ensuring that new (and pre-existing) data sets are Findable. That is, editors may confirm that (1) **URL**s and digital identifiers for concomitant data sets are prominently notated in the publication text; (2) bibliographic references to other publications which have their own data sets include citations for and hyperref links to those data sets if possible; and (3) the authors provide a brief overview of their data set when necessary to help researchers access the raw data and to help readers understand how the raw data has informed the research/findings presented in the publication. Moreover, editors can review the data sets themselves and verify that they properly reference the accompanying publication.

It should be noted that the proper **URL**s and formats for referencing publications may change as manuscripts work their way through the publishing process, from submission to publication. For example, a web link for accessing a peer-reviewed article may only be defined once the document is accepted for publication. Also, some institutions or organizations (for instance, Gates Open Research) publicly track a submission through different stages of processing (links to articles may be accessible even while they are still being peer-reviewed or proofed). When a data set links to its associated publication, such information about the current status of the submission should be duly observed in the data set.

In short, the precise publication-related data within a Supplemental Archive might need to be modified one or more times as a manuscript submission is processed, so editors should be prepared to notify authors when a change within the data set is necessary.

Publishers   Assuming that authors and editors follow the steps just outlined, articles and Supplemental Archives could be published in any environment without directly altering publishers' workflows or technologies. However, there are steps which publishers might take to enhance the usefulness of **S**DR**F** resources, and to derive additional scientific and commercial value from hosting articles that utilize **S**DR**F**. These steps include:

- **Feature Data-Set References on Article Front Pages**   The "front page" of an article is a **URL** which uniquely locates a given publication on a publisher's portal. Usually this front page includes an abstract, bibliographic citations, author names/affiliations, keywords, and other high-level info about the document; in some environments, the front page also reproduces the full text of the article (usually in **HTML** format). Even with this information set forth, however, it can be difficult for readers to identify which publications are paired with open-access data sets and to locate those data sets when they are available. Publishers can rectify this situation by including links to data sets near the top of the front page.

- **Recognize Data-Set Microcitations in Manuscripts**   Properly connecting publications and data sets requires customized LaTeX commands or **XML** attributes, so as to mark and annotate relevant locations/passages within the document. To fully utilize **S**DR**F** — or, indeed, any rigorous data-sharing protocol — publishers accordingly need to expand their in-house media for internally representing manuscripts in the pipeline with a collection of additional tags, commands, and/or attributes. The details of these add-ons, depending on the technology used, may be codified within individual **SDRM**s.

- **Recognize Alternative Manuscript and **PDF** Formats**   This applies to non-restricted publications where full text representations are allowed to be included within a Supplemental Archive. In this situation, the ideal **PDF** version of a document — as well as of the machine-readable encoding of the text — may be different from what is internally adopted by the publisher. For example, an article may appear as a chapter in a book, typeset according to the book's conventions; in the data set, however, that same article is openly shared as its own document, and might use different typesetting rules optimized for its specific content. Also, the machine-readable full-text representations presented in the data set should be optimized for Text and Data Mining (**TDM**), and could, therefore, differ from the internal encoding used

by the publisher. For these scenarios, publishers should allow for *alternate versions* of both human-readable and machine-readable publication text being incorporated into the publishing process, where these alternatives end up becoming assets within Supplemental Archives. Such alternatives may not differ in terms of textual content (although a data-set-specific version of an article could include supplemental instructions for using the data set), but they can differ in the visual formatting of the text as well as how the text is encoded.

- **Develop and Share Plugins for Scientific Applications**   Ideally, publishers should provide plugins or standalone applications for readers to use after they have downloaded data sets. While readers can open raw data files in the proper applications, those applications may not have the capabilities to understand the other files in a Supplemental Archive. Publishers should therefore provide tools or plugins so that these applications can access the whole archive, not just the raw data. If portals link to web sites where readers can download the applications for viewing raw data files, these links could be paired with instructions on how to obtain and use plugins provided by the publisher.

- **Register Cross-References between Publications and Data Sets**   Publishers' databases should be configured to clearly designate when a publication is linked to an associated data set. Moreover, certain third-party biblioinformatics services, such as CrossRef, allow publishers to introduce additional attributes describing publications. Connections between publications and data sets may, therefore, be formally declared in these third-party contexts.

- **Incorporate Data-Set Information into Publisher APIs**   Most publishing portals use APIs to support application-level queries for finding or accessing publications. These APIs can be extended to systematically honor associations between publications and data sets, e.g.: (1) return the URL for a data set when given an article's Document Object Identifier; (2) return information about an article when queried by a software component designed to manage data sets; (3) search for publications whose data sets fit given criteria (file format, programming language, subject area, etc.); and (4) given a publication, return basic information about its associated data that may help a reader decide whether to download the data set (file format, total file size, preferred application, and so forth).

  Data set info displayed visually for readers on document portals should also be incorporated into APIs. At the minimum, an API endpoint should yield data set ids given document ids. A more complete set of API queries for data sets might include:

  1. Search for publications based on attributes of their associated data sets (e.g.: file formats/programming languages; recommended data-viewing software; keywords in data-set descriptions; protocols as identified by standardization formats like MIBBI; repository host where the data is stored);

  2. Return a complete data-set description object serialized for use by an API client library, given a data-set identifier;

  3. Individual API endpoints for the most crucial metadata given a data-set identifier (e.g.: file format; size; authorship; recommended applications);

  4. Handle requests for microcitations generated by document signals, the same signals used for inter-application networking between document viewers and scientific applications;

  5. Provide API endpoints for finding and using plugins (discussed in Appendix I).

Benefits for Publishers

We believe adopting SDRF technology can derive several benefits for publishers (as well as institutions, universities, or independent journals which publish their own material). Bear in mind that the details of SDRF are adapted to different SDRMs; as such, the characterization of SDRF content presented here is provisional, and may take different forms in different publishing environments.
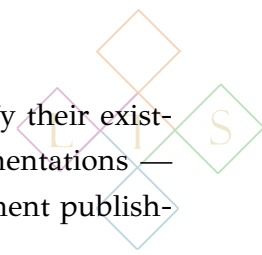
Therefore, adopting SDRF does not require publishers to substantially (or at all) modify their existing software. In general, though, incorporating certain SDRF protocols and/or implementations — whether as software extensions or simply as recommended design patterns — can augment publishers' offerings in several ways:

1. **Improving Impact Factor**   By explicitly linking publications and data sets, publishers amplify the likelihood that researchers will find the publications. Data sets present an additional/alternative route for scholars to locate articles and books, which would otherwise be found via keyword searches or bibliographic references. This would likely increase citations and downloads, because researchers would learn through data sets about publications of interest that they would not discover otherwise.

2. **Driving Traffic to the Publication Site**   Despite standardizations such as DCC and Research Objects, the vast majority of published data sets fail to adhere to these standards. Instead, they are shared haphazardly, without proper organization or meta-data. As a resut, those data-sets which *do*, in contrast, adopt rigorous curation standards are more likely to be found by software designed for contemporary data-publishing specifications. Well-curated resources also stand out from their peers on data hosting platforms. Some hosting environments explicitly introduce a "score" to measure how well each data set is organized and documented (e.g., the MIFLOWCYT, or "Minimumum Information about a Flow Cytometry Experiment," rating on **flowRepository.org**). Accordingly, those data sets which score highly on existing or future metrics — assessing how systematically their data is organized — are more likely to attract researchers' attention, which in turn drives traffic to the publication sites where data sets' corresponding articles are indexed.

3. **Research Influence**   Articles which are paired with conscientiously curated, well-documented and reusable data sets are more likely to spur further research than publications whose raw data is not shared at all or is shared in ways which add extra effort for researchers seeking to re-use or re-examine the data. In general, scientists gravitate to raw data which can be plugged most readily into the computing environments they use for their own research. For this reason, research work — expressed both in data sets and in publications discussing the research — is likely to be more influential when scientists deem the data convenient to use and expand upon.

4. **Enhancing the Publishing Ecosystem and User Experience**   As digital media becomes an increasingly important part of scientists' use and experience of published research, publishers are expected to provide ever more sophisticated technology for visualizing data and documents. Aside from just showing PDF or HTML views of articles, publishers are gradually introducing multi-media platforms which permit interactive data access, 3D data visualization, audio and video content, and similar features of enhanced "Reader Experience." Interfacing directly with scientific applications — e.g., via publisher-specific application plugins — is a natural corollary to this trend, adding yet more computational heft to publishers' platforms. However, properly networking publications with specific software demands rigorously documented links between publications and data sets, such as provided via SDRF.

Appendix I: Use-Cases For Publisher Plugins

Publications often have graphics or multimedia content that require special viewers. More often than not these assets are visual or structured representation of data which may be placed in a data set — e.g. a table printed inline in the document is derived from a spreadsheet which is its own data file. Many document portals allow readers to examine these inline figures in separate frames and windows. To support more sophisticated multimedia content, with file types specific to different scientific fields, publishers are attempting to create browser plugins to visualize complex scientific data. In general, however, such scientific assets should really be viewed in domain-specific scientific software. This suggests that publishers' portals and document viewers should interoperate with existing applications — rather than, or at least in addition to, trying to emulate these applications via browser plugins.

One facet of a publisher's SDRM would be how their portals identify and communicate with the "preferred software" that would be used to view different Supplemental Archives.

Because the data sets linked to publications often have to be accessed via specialized scientific applications, special programming is needed to ensure that each publisher's software can seamlessly interoperate with domain-specific scientific software. Although scientific applications can open files in domain-specific formats, they do not typically know how to identify such files in the context of a published data set. For example, Flow Cytometry software has the capability to open FCS files, but not to open Research Object Bundles which contain FCS files. As a result, existing scientific applications must be extended so that they can recognize data-publication formats such as Research Objects and SDRF.

To demonstrate how such plugins could work in practice, consider the connections which exist between parts of publications (e.g., a figure illustration) and corresponding parts of data sets (e.g., .csv files). Here are some illustrative use-cases:

1. A graphic in a publication displays a segmented image. Via a context menu, this graphic can be linked to an image-processing data set which is viewed in image-annotation software (e.g. IAT, from the University of Milano-Bicocca) showing segments as annotations; or the data set is viewed in image-analysis software (such as CAPTK, from the University of Pennsylvania) where researchers can reproduce the steps taken to obtain the segmented image from the original.

2. A figure shows one rendering of a 3D manifold displaying the distribution of a high-dimensional data set under dimensional reduction. Via a context menu, this graphic can be viewed in 3D with (for instance) PARAVIEW (a popular data-visualization program).

3. A graphic shows Flow Cytometry data (generated by probing cells and cellular-scale organic materials with lasers and fluorochromes) as part of a serological assay. Via a context menu, the "event data" yielding that graphic – together with gating information, which is similar to image annotation – can be viewed in Flow Cytometry software. The event data itself is stored in FCS files (which may be accessed from web resources, such as **flowRepository**) while gating information is provided by workspace files or files (in formats such as GATING-ML) internal to the data set.

4. A graphic shows a 2D or 3D outline of the molecular structure of a protein. Via a context menu, this structure can be viewed as an interactive 3D presentation in IQMOL, a molecular-visualization application, where the raw data is obtained from the Protein Data Bank. As a variation on this use-case, every protein mentioned in a chemistry paper can be marked with a hyperlink and/or context menu launching IQMOL with the relevant **.pdb** file.

5. An archaeology paper shows images from a site which has been investigated with advanced imaging/laser equipment, e.g. point-cloud scanners or Matterport 360-degree cameras. Via context menus or links embedded in figure captions, the associated data set may be rendered in 3D graphics software such as MESHLAB, or in Panoramic Photography viewers such as Panini.

6. A computer science paper includes code samples; context menus or hyperlinks at the start of each sample can be designed to open an IDE (e.g. QT Creator) or an interactive notebook (e.g., JUPYTER) to run the sampled code, with the complete code provided by a data set.

7. A paper presenting lab results includes a schematic summary of lab protocols generated by BIOCODER. A context menu can then launch QT Creator to display the actual BIOCODER files, included in a data set.

8. A paper discussing a vaccination campaign references data sets evaluating subjects' immunological responses, geospatial visualization of the campaign's target area, and epidemiological modeling. This paper may then be linked to several different scientific applications: ARCGIS for the geospatial data; FlowJo for the immunology; QT Creator for running epidemiological simulations (modeled via the **EpiFire** library, for instance).

9. A paper on stroke rehabilitation might use **DICOM** to record both image data (e.g. **MRI**s) and audio data (e.g. documenting aphasia). These **DICOM** files need to be opened in the proper software – perhaps **MEDINRIA** for **2D** images, **ITK** Snap for **3D** images, and Audacity for audio. When the publication text describes the diagnostic significance of particular **DICOM**-encoded findings, then (perhaps via **DICOM** Structured Reporting), these text segments should be annotated to identify the software category appropriate for the specific **DICOM** files used.

For each of these scenarios, there must be a functioning integration between the document-viewers which readers use for the main publication, and the domain-specific applications they use for the data set files. One way to achieve this is for publishers to provide a suite of plugins to scientific/technical applications used to view most kinds of data files that are published in conjunction with scientific literature. Each plugin would be configured to recognize signals generated from documents created via that publisher's software and to respond to those signals properly (retrieving and opening data files from the relevant open-access data sets).

It would, of course, be impractical for every publisher to implement plugins for many different scientific applications in isolation. A more reasonable solution would be to formulate a protocol, in consultation with both publishers and application developers, stipulating how publishers' content — in particular, **PDF** files and web pages displaying research papers — should interact with scientific software. Solutions already commonly used, such as the Common Workflow Language (**CWL**), could be a starting-point for this standardization; however, the fully developed protocol should be tailored to the unique demands of triangularly connecting publications, data sets, and scientific applications.

Interactive Digital Notebooks (such as **JUPYTER**) embody another kind of technology that is sometimes employed to share research and/or to demonstrate research methods. Such notebooks typically display an interactive presentation combining computer code, explanatory text, and data visualization (e.g., charts, plots, or diagrams). A common use-case for digital-notebook-style presentations is to load a quantitative data set, perform some statistical transformation on the raw data, and visualize how the statistical operations modify the data productively. In such scenarios, the important technological elements include a coding language or library conducive to statistical analysis (such as **R** or **PYTHON**), and a **GUI** library that can convert statistical data to **2D** or **3D** graphics.

This sort of statistics-oriented analysis is only one example, however, of how data sets may be explored interactively. There are many use-cases in which analytic and visualization capabilities from diverse scientific or mathematical fields (not just statistics) are relevant. We should, accordingly, expand our notion of "Interactive Digital Notebooks" to encompass a wide range of interactive presentations constructed through the analytic capabilities and **GUI** features of scientific applications. In this sense, a Digital Notebook would be a component embedded inside a pre-existing application, and a common protocol can be specified with respect to how Digital Notebooks are initialized and interact with the host software. Scripting languages like **R** or **PYTHON** might be replaced in this context by more light-weight scripting options more conducive to **C++** embedding, such as Embeddable Common Lisp or **ANGELSCRIPT**. Digital Notebooks in this guise would be proper vehicles for formalizing a publication-to-application networking protocol.

Moreover, aside from overall links between data sets and publications, parts of a publication may be linked to parts of a data set. For example, a table, scatter-plot, or plot-graph displayed in a publication could be linked to a **.csv** file in a data set. Correlations between publications and data sets may also be based on Controlled Vocabularies and other ways of identifying concepts in publication texts. If publications and data sets are both annotated using the same Controlled Vocabulary, then links are implicitly created between the two resources, wherever a part of a data set and of a publication text, respectively, are annotated as concerning the same concept. To make these connections explicit, publication manuscripts should ideally be annotated with concepts or data structures that directly or indirectly define granular connections between publications and data sets. When publications are
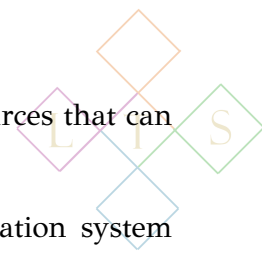
fully open-access, these annotations can be included in machine-readable full-text resources that can be freely downloaded.

Connections between publications and data sets can be asserted by using a micro-citation system within the data set and annotations within the publication. Here are a few examples: the name of a protein may be marked in publication text and also linked to a Protein Databank file in a data set; the name of a flourochrome may be marked in a publication and cited as one channel group in a Flow Cytometry file; the description of an Image Segmentation algorithm, such as a Sobel-Feldman operator, may be linked to an image in a data set resulting from that operator applied to a primary image, and also mentioned in the text as part of the description of an image-processing workflow; biochemical assays may be described in the text and also formally declared via **MIBBI**, **BioCoder**, or SpringerProtocols, with that systematic info included in the data set; and so forth.

In short, a Digital Notebook protocol can be formulated which incorporates models of document annotations and data micro-citations, identifying granular connections existing between specific locations in article text and specific facets of the data set. An Interactive Digital Notebook should allow users to switch **GUI** focus back and forth between the text (presented, say, in **PDF** form) and the windows where the data-set is visualized. Such cross-referencing, indeed, is an important aspect of interactiveness.

As is demonstrated by **SeCo** — a Digital Notebook based on **HyperGraphDB** — implementing Digital Notebooks is facilitated by Hypergraph Database technology, so a cross-application Digital Notebook protocol could profitably be centered on hypergraph database theory. In this manner a Digital Notebook protocol would be thematically related to **SDRF**'s proposed protocol for remote database access, discussed next.

Appendix II: The Transparent Hypergraph Query Language

SDRF does not overtly proscribe any particular format for encoding raw data in **SDRF** archives; as such, **SDRF** cannot make definitive recommendations for how this data may be consumed. Nevertheless, **SDRF** draws on contemporary research into database engineering, particularly the body of literature concerning Hypergraph Database design and the use of hypergraphs as multi-faceted, general-purpose formats for representing information. Hypergraph Database engines are favored in part because hypergraph schema can incorporated many families of data models, so that Hypergraph Databases can store diverse, heterogeneous data, sourced from disparate origin-points. Nevertheless — despite some common features which make Hypergraph Database engines, collectively, good architectural choices for heterogeneous and decentralized data persistence — there are several distinct hypergraph engines widely used in contemporary technology, each with slightly different data models and protocols. It is therefore a worthwhile project to define a multi-purpose hypergraph data model which merges idiosyncratic details of these various platforms, offering a common framework for accessing hypergraph data.

Furthermore, such a framework's value would be additionally enhanced by applying it to information spaces which are not, necessarily, backed by a Hypergraph Database engine but which are amenable to an intermediate software layer that translates hypergraph-oriented queries to a query language endemic to the actual database used. Although hypergraph databases are gaining a greater foothold in some computing domains (biomedical, governmental, financial, etc.), a much larger proportion of database instances rely on more traditional data-storage technology. With proper software adapters, it is possible to construct "views" onto data spaces such that they can be operationally interacted with as if they employed hypergraph models internally.

These points apply not only to databases themselves but also to any systematic data compilation, including published data sets. Therefore, one use-case for **SDRF** is to package raw data in such a way that it may be queried according to protocols established for Hypergraph Database engines (and

then extended to other information architectures).

Such is the motivation behind the "Transparent Hypergraph Query Language" (THQL), which LTS is designing as a paradigm for organizing data sets with the goal of formulating a hypergraph query model that may be extended to hypergraph databases, as well as to information spaces that can be programmed to emulate such databases. THQL is characterized as "transparent" because this technology is grounded on queries expressed and evaluated directly in C++ code, so that query-processing logic may be openly observed in source code and through a debugger (this does not preclude a distinct THQL *language*, but such would be implemented by translating query expressions to C++ procedure calls; behind the scenes THQL is treated as a C++ DSL). While oriented first and foremost to hypergraph database architecture, THQL also serves as an extension to Conceptual Space Markup Language, which was formalized (by Benjamin Adams and Martin Raubal) as an operationalization of Peter Gardenförs's Conceptual Space Theory.

In its design, THQL represents the different use-cases in which Hypergraph database technology is selected as the optimal strategy for data storage and information management or "knowledge engineering." In general, THQL recognizes four architectural paradigms which undergird the Hypergraph database model (that is, an idealized model abstracted from specific products):

Object Serialization   One appeal of Hypergraph database engines is that they minimize the biolerplate code needed to serialize (and later retrieve) "objects," or typed data structures, in the sense of Object-Oriented Programming. These engines are not "object" databases where object-persistence happens automatically, but they are designed to manage objects as first-class integral values more fluently than other database architectures (relational, hierarchical, graph-oriented, etc.). This means that, when querying a hypergraph database, it should be easy to manipulate the query results as one single typed value which is an instance of an application-specific type implemented within the application which initiates the query, or as an iterator or "recursive factory" from which such typed values can be initialized. THQL accordingly incorporates logic to manage queries and query results according to such an object-factory design pattern.

Collections-as-Values   Another beneficial feature of Hypergraph databases is that collections of similarly-typed values can be manipulated as single values, requiring less intermediate modeling than appears in SQL or RDF, for instance. THQL accordingly presents views on hypernodes which emulate collections data structures implemented in mainstream programming languages (arrays, lists, tuples, stacks, queues, deques, etc.).

Graphs and Network Models   Although "nodes" (or, more accurately, *hypernodes*) in a Hypergraph database have more internal structure than nodes in ordinary (non-"hyper") graphs, hypergraphs are also models of hypernodes connected in graph-like networks, so conventional graph queries and analyses (ignoring hypernodes' internal content and considering just connections between hypernodes) can be evaluated on hypergraphs no less than on ordinary graphs. Accordingly, THQL incorporates common features of conventional graph-query languages.

Hypernodes as Relational and Conceptual Structures   This facet of THQL integrates analytic protocols derived from various existing Hypergraph technologies (as well as AI research that is operationalized and/or facilitated by these technologies), such as **Graken.ai**, HYPERGRAPHDB, Conceptual Space Theory, Fuzzy Logic, Conceptual Role Semantics, and so forth. Each of these paradigms have distinct interpretations of the internal structure of hypernodes. To accommodate this variation, THQL allows hypernodes' internal structuration — the relationship between the overarching hypernode and its contents — to be notated according to several different systems.

LTS hopes to deploy a THQL prototype in conjunction with published SDRF data sets and also in the context of data-integration projects, including ones that will be examined in a forthcoming Elsevier volume, "Data-Integration and Conceptual-Space Models for Covid-19."

For more information please contact:
Amy Neustein, Ph.D., Founder and CEO
Linguistic Technology Systems
amy.neustein@verizon.net • (917) 817-2184

18