WHITE PAPER

Next-Generation Data-Integration Tools for Precision Medicine:

Immunotherapy, Immunoprofiling, Covid-19 Clinical Trials, and Patient-Outcomes Research

1 Introduction

Linguistic Technology Systems (LTS) is developing a suite of software libraries which focus on native application development and hypergraph database technology.[1] This White Paper will discuss applications of these software tools to biomedical data sharing, placing an emphasis on precision medicine, even though these same tools may be used in other areas of medicine as well. We focus on engineering a "next-generation" software infrastructure for precision medicine data-sharing, combining existing libraries with new components. The resulting technological framework would be aligned with biocomputational industry trends; for example, implementing data integration solutions through property-graph common representations. LTS, however, would take such data integration solutions to the next level, by synthesizing existing paradigms into a collective — in particular, combining Property Graphs with Hypergraphs. LTS's novel "property-hypergraph" model is a step forward for NoSQL database technology because this new model permits many different data structures, encapsulated in many different kinds of application-level data types, to be encoded/serialized, queried, and synthesized in a database context. The unique advantage of LTS's property-hypergraph model is that, from the point of view of application integration, it is developer-friendly and conveniently supports heterogeneous queries across multiple application-level data types.

Precision medicine is based on the scientific realization that the effectiveness of a certain clinical intervention — for instance, immunotherapy as a cancer treatment — is dependent on factors that vary significantly between/among patients. In the case of immunotherapy for cancer, assessing the likelihood of favorable outcomes requires genetic, serological, and oncological tests which need to be administered prior to the commencement of treatment. Therefore, analysis of precision-medicine outcomes requires detailed immunoprofiling — building immunological profiles of each patient before (and perhaps during/after) the immunotherapy regimen — alongside an evaluation of how well the patient responds to the treatment, with the best outcome being cancer remission or non-progression.

The contemporary emergence of Precision Medicine is driven, in part, by advances in diagnostic equipment which enable immunological profiles to be much more fine-grained than in previous decades. This level of patient-profiling detail enables granular three-way analysis involving (1) patients' immunology; (2) treatment regimens; and (3) clinical outcomes. This three-way approach would have the goal of identifying signals in immunological profiles that indicate which therapeutic intervention(s) is most likely to engender a favorable outcome. As with any statistical analysis, predictive accuracy increases in proportion to the amount of data available. As such, further refinement of precision medicine depends in part on data aggregation: pooling a number of observational studies where patients' immunological profiles are described, in detail, alongside reports on treatment plans and patient outcomes.

[1] Further discussion of our data-integration tools can be found in the LTS White Paper, "A New Protocol for Improving Biomedical Data Sharing across Multiple Institutions," https://raw.githubusercontent.com/Mosaic-DigammaDB/CRCR/master/hgdm/HGDM-a-new-protocol.pdf, and in the LTS protocol definition paper, "The HGDM Protocol (Hypergraph Data Modeling): How to Unify Hypergraph Database Architecture for Multi-Institutional Biomedical Data Sharing," https://raw.githubusercontent.com/Mosaic-DigammaDB/CRCR/master/hgdm/HGDM-protocol-definition.pdf. Our technology is previewed in a limited (demo-style) repository at https://github.com/scignscape/ntxh on the **wip-facs** branch; files in this repository will be cited in a couple of places below when discussing concrete implementations of property-hypergraph technology.

Within the scientific research community, organizations such as the Society for Immunotherapy of Cancer (**SITC**) and the Parker Institute for Cancer Immunotherapy (**PICI**) have developed programs and tools to share immunotherapy research data. Given their relative novelty, such resources are, however, much smaller in scale than comparable (but less cutting-edge) projects, such as Cancer Commons or the **RSNA** (Radiological Society of North America) Image Share program. Next-generation data-integration tools for precision medicine, such as those developed by LTS, would offer larger-scale immunotherapy data sharing that can be advanced by curating software components and a software-development ecosystem that would enable more effective multi-disciplinary and multi-institutional collaborations.

Similar to immunotherapy, in the Covid-19 population we are also presented with multidimensional patient data. As such, data-integration tools must, likewise, be designed to accommodate the many heterogeneous data types corresponding to diagnostic/clinical information about Covid-19. Such information traverses assessing patients' viral load, analyzing patients' blood samples for antibodies or genetic material suggesting prior Covid-19 infection, profiling patients' immunological fitness prior to the start of a treatment for Covid-19, quantifying each patient's immunological responses once they are administered treatment, tracking the genomic evolution of SARS-CoV-2 as the virus mutates into different strains, assessing plasma viscosity (**PV**) for detection of unusually high levels of fibrinogen — leading to atypical blood clots (that are refractory to standard anticoagulant therapy), and so forth. One area where data-integration is significant for Covid-19 research is that of protein biomechanics, where many different proteins are therapeutically relevant to Covid-19.[2] For that matter, each of these proteins can be connected to molecular data, biassay information, and, in many cases, to clinical trials testing therapies where the specific protein acts as a target for inhibiting SARS-CoV-2. In combination, aggregating all of this information yields a heterogeneous (but interconnected) data space, characterized by data derived from multiple scientific disciplines and multiple laboratory methods. In Covid-19, in particular, our need for flexible/adaptable data-sharing tools to be able to accommodate such heterogeneous data is paramount because, as we have seen in the past nine months this disease process continues to surprise us in its broad-ranging effect on multiple organ systems, which can be disabling both in the short-term and long-term. Data-integration tools must, therefore, be flexible enough to accommodate these new kinds of data that continue to emerge in both the course of clinical trials and routine care in the Covid-19 patient population.

2 Overcoming Impediments to Data-Sharing Initiatives in Precision Medicine

Given the highly specialized nature of software used in most biomedical diagnostic and investigative libraries, raw laboratory data tends to get excluded from data-sharing initiatives. Instead, the information which is shared between hospitals or research centers tends to be a simplified overview, with brief summaries of diagnoses, treatments, and/or outcomes — but without any sort of in-depth data sharing. Ironically, detailed lab data sometimes does get preserved and analyzed within research publications, where scientists can deposit raw data on general-purpose data-hosting platforms (such as Dataverse, Dryad, or Open Science) or domain-specific platforms (such as the Flow Repository for **FCS** data, **GenBank** for genomics, or PhysioNet for physiology and cardiology). However, even while such data-hosting platforms are available to scientists, these platforms are not typically incorporated into data-sharing initiatives. For example, frameworks such as **OMOP** or **CDISC** do not include standard protocols for accessing data platforms' **API**s. Nor do data-sharing initiatives usually implement their own platforms for sharing most kinds of diagnostic/lab data;

[2] For instance, Qiongqiong Angela Zhou *et al.* identify 64 different proteins which are therapeutically relevant to Covid-19; *see* "Potential Therapeutic Agents and Associated Bioassay Data for COVID-19 and Related Human Coronavirus Infections," *ACS Pharmacology and Translational Science* Vol. 3, No. 5, 2020, pages 813-834, https://pubs.acs.org/doi/10.1021/acsptsci.0c00074.

consequently, the only data which researchers can access from a data-sharing initiative tends to be summarial and lacking in patient-centric detail.

These limitations have been identified as obstacles diminishing the value of data-sharing initiatives. As one example, in a paper discussing research funded by the Swiss National Science Foundation into the sequencing of immunoglobulin repertoires (Ig-seq), Simon Friedensohn *et al.* comment: "A major challenge when performing Ig-seq is the production of accurate and high-quality datasets [because] the conversion of mRNA ... into antibody sequencing libraries relies on a number of reagents and amplification steps ... which potentially introduce errors and bias [that] could alter quantitation of critical repertoire features." The authors opine that one way to confront this problem is "by implementing synthetic control standards, for which the sequence and abundance is known prior to sequencing, thus providing a means to assess quality and accuracy."[3] We see here that the underlying problem in this context is how different laboratories may use different techniques and protocols to achieve similar diagnostic/investigative goals. Consequently, when data is merged from multiple hospitals, it is likely that the raw data derives from different labs, which can lead to situations where protocol variations across each site can introduce errors and bias that contaminate the aggregate data-sharing results.

In the context of Covid-19, Shrestha *et al.* argue for "precision-guided studies" to be prioritized "[r]ather than conducting trials using the conventional trial designs and poor patient selection." To accomplish this, the authors argue for "large multicenter trials" which incorporate "predictive enrichment strategies ... to identify and thus target specific phenotypes [patient-profile characteristics], potentially raising the possibility of positive trial outcomes."[4] Given that there are many variable factors among patients that serve as distinct criteria for the kinds of therapeutic interventions most likely to be beneficial, this can unduly narrow the cohort size when trying to find patients that fit those specific criteria. However, by bringing together multiple institutions a larger pool of potential subjects may be brought into the framework of the study.[5] As integral to this process, the authors argue for a "robust data infrastructure" which would combine the efforts of clinicians, researchers, and data scientists so that a wide panoply of therapies customized to a patient's profile can be made readily available to the patient because the institution at the point-of-care would now be a participant in a vast network of health-care institutions that would be involved in the multicenter trial. In fact, in gauging public sentiment, patients during the Covid-19 pandemic are increasingly reluctant to participate in conventional trials where they run the risk that instead of being assigned to an experimental group they are assigned to a control group where they might lose the opportunity to receive life-saving treatment. An approach such as Shrestha *et al.* would prevent the loss of subjects partaking in important clinical trials inasmuch as participation in precision-guided studies would guarantee treatment.

Biases that are likely to arise from preselecting patients can be miminimized by conducting concurrent multiple trials, which can either take place within one institution proper, or across multiple institutions — as in the multicenter paradigm discussed above. And in fact, in such cases, though there is an absence of a control group, comparative analysis would still be able to be performed,

---

[3] *See* Friedensohn S, Lindner JM, Cornacchione V, Iazeolla M, Miho E, Zingg A, Meng S, Traggiai E and Reddy ST (2018) *et al.*, "Synthetic Standards Combined With Error and Bias Correction Improve the Accuracy and Quantitative Resolution of Antibody Repertoire Sequencing in Human Naive and Memory B Cells", *Frontiers in Immunology* (special issue titled "B Cell Biology"), June 20, 2018, page 2 (https://www.frontiersin.org/Journal/10.3389/fimmu.2018.01401/full). This was work was funded by Swiss National Science Foundation Systems Antibody RTD project.

[4] *See* Gentle Sunder Shrestha, Hem Raj Paneru, and Jean-Louis Vincent, "Precision medicine for COVID-19: a call for better clinical trials", *Critical Care*, Vol. 24, 2020, (https://ccforum.biomedcentral.com/articles/10.1186/s13054-020-03002-5).

[5] In addition, as pointed out by Shrestha and his coauthors, such large multicenter trials can produce "large cohorts of patients in a shorter time period."

because the results of all the concurrent trials would be compared against one another. In such a scenario, rigorous data modeling for each individual trial is especially important. Using specialized data-integration tools to accommodate heterogeneous patient populations, such as those proposed by LTS, researchers would be able to perform large-scale analyses, encompassing all the aggregate trial data. Pragmatically, using such integrative data-modeling tools Covid-19 trials would be able to be structured, in part, by precision-medicine profiles of Covid-19 patients, which would steer different patients into different trials based on crucial patient-specific data. These tools, as mentioned above, allow for the integration of *heterogeneous* data — in Covid-19 trials, that would include immunological, cardiovascular, neurological and other general medical-history data which would be obtained for each patient prior to their involvement in a trial, and likewise made available for subsequent analysis. Customized software that provides flexible and adaptive code libraries, such as LTS's **ConceptsDB** database engine, would allow for trial enrollment criteria — across multiple trials/multiple institutions — to be continually updated so that patients could receive the best treatment plan tailored to their defining profiles, which may change over time.[6]

Using such tools, Covid-19 data-sharing initiatives would be able to include fine-grained evaluations, such as waveform analysis of **EKG** data or cytometric/chromogenic/flourogenic calculations (or their variations) for antigen assays.[7] This would allow Covid-19 trials to identify data structures which properly convey these sorts of detailed diagnostic findings — for example, biochemical fluorescence charts or **EKG** signal-processing feature vectors. With regard to fine-grained data structures, Lyon *et al.* summarize that for **EKG** signals "[m]orphological features include the coefficients of the Hermite transform, the wavelet transform or the discrete cosine transform ... that aim to model the ECG beat instead of extracting features from the raw data."[8] As such, feature-vectors extracted from **EKG** waveforms are amongst the kinds of heterogeneous data structures which would be useful for Covid-19 trials.[9] To illustrate this point, the LTS demo includes versions of several acoustic-processing libraries, including code for **EKG** analysis, engineered to interoperate with a property-hypergraph database (linked to acoustic data sets with signals encoded in binary files).[10]

2.1 The Need to Design Data-Sharing Tools to Match the Complexity of Immuno-Oncology Data

While some of the data related to immunological profiles may be sociodemographic or part of a patient's medical history (fitting nicely within conventional Clinical Research Network models) that kind of data is, however, distinct from the granularity of contemporary immunoprofiling that is powered by *diverse* kinds of highly specialized diagnostic equipment and methods — which require special-purpose file formats and software. For instance, one dimension of immunological profiling is "immune repertoire"; the more robust a person's repertoire, the wider variety of antibodies they can produce to fight off pathogens. Immune repertoire is often measured by studying genetic diversity in B-cells; in recent years, this has been done using "Next Generation Sequencing" (**NGS**), which produces files in formats such as **FASTQ**. Another dimension of immunological profiling is

[6] More information about ConceptsDB is available at https://raw.githubusercontent.com/Mosaic-DigammaDB/CRCR/master/hgdm/ConceptsDB.pdf.

[7] LTS will review Covid-19 data sharing methodologies and initiatives — and publish computer code demonstrating the relevant methods — in *Data-Integration and Conceptual-Space Models for Covid-19* (Elsevier, 2021).

[8] *See* Aurore Lyon, Ana Mincholé, Juan Pablo Martínez, Pablo Laguna, and Blanca Rodriguez, "Computational techniques for ECG analysis and interpretation in light of their contribution to medical advances", *Journal of the Royal Society Interface* January 2018 (https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5805987/).

[9] Because machine-learning and other statistical analytic techniques operate on **EKG** feature vectors, these vectors serve as intermediate representations of **EKG** waveforms from which diagnostic findings are derived. Such vectors are, therefore, a good example of the fine-grained information that should be included in data-sharing initiatives. In particular, this information should be integrated into trial data made available for subsequent analysis.

[10] This LTS demo is designed to provide an example of how a property-hypergraph database can serve as an entry-point and management tool for data sets composed of multiple binary files. The database can perform feature-extraction on individual binary files and store the resulting feature vectors as database records.

quantifying the proportions of different sorts of blood cells in a patient's blood sample, which is typically done via Flow Cytometry or Mass Cytometry, yielding **FCS** (Flow Cytometry Standard) files. The immunological evaluations which are the goal of these methods are sometimes called Cell-Type Classification or "Automated Cell-type Discovery and Classification" (**ACDC**). Yet another dimension of immunological profiling in the context of cancer treatments is the Immune Environment, or "Tumor Microenvironment," which concerns the interplay between a tumor and surrounding cells, such as blood/lymph vessels.

With respect to immunological profiling of the tumor microenvironment, this environment is sometimes surveyed using high-powered contemporary bioimaging techniques, such as "Confocal Microscopy" — a kind of layered imaging that acquires **2D** scans at different depths, creating a **3D**-like model. Image analysis is necessary to derive oncological findings from these images, looking not only for "Distinguished Regions" but also for linear paths and patterns (e.g., connective tissues or blood vessels). As a result, Confocal Microscopy data involves both raw images and image-annotations that summarize image-analysis/Computer Vision results. **DICOM** and **OME-TIFF** are examples of common bioimaging formats.[11]

As outlined in the table, below, immunological profiles draw on a diversity of data formats and lab/data-acquisition modalities, though this table is not intended to be a complete list of criteria or file formats, but rather to indicate the range of diagnostic technologies and data formats which are relevant to immunoprofiling.

Immunological Profile Dimension	Lab/Clinical Method	File Format
Sociodemographic	Scan Patient Records	CSV, SQL
Medical History	Scan Patient Records	CDISC, OMOP, PCOR
Immune Repertoire	NGS	FASTA, FASTQ
Cell Type Classification	Flow/Mass Cytometry	FCS
Immune/Tumor Microenvironment	Confocal Microscopy	DICOM, OME-TIFF, AXFi

As evidenced in the table, immuno-oncology data sharing is inherently more complex when compared to data-sharing initiatives which focus primarily on clinical outcomes — such as the Observational Medical Outcomes Partnership (**OMOP**) or the Patient-Centered Outcomes Research Network (**PCORNET**). This is so, because immunoprofiling data is more *heterogeneous* and multi-disciplinary than ordinary clinical data. Formats such as the **OMOP** Common Data Model (**CDM**), the **PCORNET CDM**, or the Clinical Data Interchange Standards Consortium (**CDISC**) specifications promote data sharing primarily through **SQL**-style tables, where data analysis and extraction can be achieved via conventional **SQL** queries. The situation is very different, however, when the data that must be exchanged derives from specialized hardware and software, which demands special-purpose file formats, parsers, and query engines. This problem-space is accentuated when preparing multi-site sharing that can span dozens of hospitals, research centers, and/or laboratories, because individual institutions are known to have their own indigenous software that may not interoperate well with the software platforms of other institutions.

2.2 Fostering Workable Software Alignment across Multiple Hospitals and Laboratories

Data-sharing initiatives need to pay particular attention to the logistics of hosting/accessing data in contexts where granular patient-specific information is important, such as immunoprofiling. Ques-

[11] Robust image-annotation formats are not particularly well standardized (at least to a degree comparable to **FCS** or **FASTQ**), though some annotations can be expressed via formats such as **DICOM** Presentations, **COCO** (Common Objects in Context), **VOC** (Visual Object Classes), or LTS's own **AXFi** (annotation exchange format for images).

tions which need to be addressed include (1) where data is to be hosted; (2) how participating institutions should submit data to a central repository; (3) how participating institutions and/or outside investigators should access previously-deposited data; (4) how to ensure anonymization of patient-specific records; (5) how to ensure that different labs used by different hospitals are utilizing compatible protocols, so that results from multiple labs/hospitals can be seamlessly merged in a shared data commons; (6) how to ensure proper alignment between software employed at different institutions; and (7) how to incorporate data curated within the context of a multi-institutional data-sharing initiative into scientific papers documenting research findings. Each of these areas of concern are technically demanding because of the complex and heterogeneous nature of immunological profiles.

One must take into consideration the fact that data sharing networks span multiple hospitals, with multiple teams of scientists and programmers at each facility. Consequently, the software that powers each data-sharing initiative across multiple institutions may need to be run on dozens, if not hundreds, of computers, which may differ from one another. Nevertheless, to prevent chaos and to foster a workable software alignment across this large nexus of medical institutions, each hospital has the ability to enforce restrictions on how software is installed. For example, making use of software for a specific data-sharing initiative should not entail making any changes to the host computer (all the files needed for the initiative should be centralized in one separate folder or in multiple subfolders). As a result, all the software which powers the initiative should be able to run in a "sandbox" environment, using code and files exclusively located within a dedicated project folder. In particular, the software should not rely on "system installs" — installing libraries and other dependencies into common areas shared by multiple applications, including those unrelated to the data-sharing initiative. Such installs could (potentially) adversely affect the operation of other software (for example, by overwriting the version of a library on which some other application depends). For this reason, hospitals and clinics should enforce particularly strict rules to ensure that essential medical software is not adversely affected by other applications (such as those connected to a specific data-sharing initiative).

Overall, then, one should strive for the software powering data-sharing initiatives to be self-contained — with few or no external dependencies — and to run smoothly in a "sandboxed" environment.[12] What this means is an environment that is isolated from *other* data and applications on the host computer. Naturally, developing software components which adhere to these strict requirements demands a conscientious software development methodology, but even more so, one which can get beyond the hindrances that stem from conventional practices. That is, the majority of open-source

[12] External dependencies are, for the most part, code libraries which must be installed on a developer's or user's computer to compile and/or run an application. Many operating systems, particularly UNIX variants, have "package managers" that allow developers and users to manage external dependencies by downloading and installing additional software — typically with only one or two commands — when such components are needed. For example, when compiling a program which requires the "boost" C++ libraries, programmers might be instructed to issue a command such as **sudo apt-get install lib-boost-all-devel** so that these libraries are available for the program to use. Sometimes external libraries must be present only on the developer's computers to be compiled into an application, but often the libraries also have runtime components that need to be installed on users' computers as well; in these cases, either users have to install these dependencies themselves, or external libraries are bundled together with the main application into a "package." In any case, package managers make working, at times, with external dependencies deceptively easy, which can lull programmers into ignoring the downsides of depending on too many external components. While installing dependencies is *sometimes* straightforward, there are many potential problems, such as mismatch between the version of a library which the application uses and the currently available package; version mismatches that cause a newly installed library to overwrite an older version in a way that breaks some other application (potentially crashing the computer); and wasted disk space caused by installed libraries that are rarely used. Because of these issues, developers should try to avoid external dependencies when they prove unnecessary, or at least make them optional — that is, users and developers should not be forced to install dependencies for relatively minor or supplemental features which are not essential to running an application. For instance, a science application employing **Qt GUI** components would probably not need to depend on **boost** libraries, because most of the more popular **boost** capabilities — such as **boost.filesystem** and **boost.regex** — are already available within **Qt**, which is not an external dependency because it is the framework within which the application is principally developed.

biomedical/bioinformatics software, while often scientifically sophisticated, adheres to software-development paradigms that drastically limit their usefulness in data-sharing contexts: these software projects do not adequately prioritize minimizing external dependencies and, as a result, the software is often relatively difficult to compile and install. Unfortunately, what happens as a result of this difficulty with compiling and installing is that sophisticated software which could actually add information-granularity and UX enhancements to data-sharing initiatives often gets excluded from the process. The inevitable consequence of excluding *software* is that potentially valuable *data*, particularly data that evinces the granularity of information that is sought, is excluded as well.

There are practical considerations that contribute to the lack of fine-grained data sharing despite its clinical/research importance. For example, sharing raw lab data (which can prove critical for inclusion in many data-sharing initiatives) is often hindered by the lack of software components that can be used outside the laboratory proper. This is so, because the high-end equipment found in diagnostic and research laboratories usually connect to vendor-specific software, which albeit useful for initial data acquisition/analyses, it is nevertheless wholly insufficient for sharing data across multiple sites. On the other hand, it is too daunting to conceive of each institution hosting sophisticated and costly commercial software for each immunoprofiling-related discipline so that each institution could share fine-grained raw data with one another. However, LTS addresses these constraints by noting that software which is shared across a data-sharing initiative can be widely deployed because it need not fully replicate the capabilities of the initial data-acquisition software. Instead, each initiatives' shared software can have functionality focused specifically on operations which would be invoked across the data-sharing network, such as visualization, accessing metadata, and analytics which are not too computationally intensive. This suggests that software design that would accommodate data-sharing across institutions should focus on components that can be widely *shared* and *reused*, and not just on optimizing performance alone.

In fostering workable software alignment, however, one must be careful to consider some of the hidden pitfalls. For instance, in scientific computing it is very common to achieve software sharing and sandboxing through "containers" or "virtual operating systems," such as Docker or VMWare. These technologies create a self-contained environment for running applications, often by digitally emulating a different operating system than the one that is actually installed: a typical example would be a Windows computer running Linux programs within a Docker container based on Ubuntu, a popular Linux variant. This kind of virtualization/containerization, however, is an imperfect solution because (1) "emulating" an operating system slows down every operation, which can noticeably degrade the performance of applications running in a virtual sandbox or container, particularly applications with GUI front-ends; (2) Docker, VMWare, or similar software has to be installed on the host computer; and (3) Docker (in particular) can consume large amounts of memory — even when containerized applications are not actually running — so that computers hosting Docker and other virtualization/containerization platforms would require periodic maintenance. In short, installing and using programs such as Docker or VMWare potentially alters, to a noticeable extent, the computers where virtual or containerized software is executed.

LTS, by contrast, has worked on curating biomedical and bioinformatics tools which are sandboxed by virtue of its development methodology rather than by virtualization. Though these tools run as normal applications within the normal operating system, they are operationally sandboxed by minimizing external dependencies and by aggregating data and code libraries into folders that are isolated from other resources on the host computer. Such tools, implemented in a self-contained and cross-platform manner, can be selectively integrated into data-sharing initiatives, with identical components shared across all participating institutions. The relevant tools have not necessarily been developed by LTS from scratch — in some cases LTS has modified and incorporated pre-existing open-source biomedical software, with the end result being components which are both

self-contained and interoperable. LTS's currently available tools (which cover a range of biomedical and computational use-cases including genomics, cytometry, signal processing, scripting, data persistence, **PDF** viewers and generators, and image analysis/annotation) are designed to effect maximum interoperability among the various software platforms across multiple healthcare institutions and research laboratories.

3 Proposing Software Engineering Principles/Tools Optimized for Data Sharing

In proposing new ways to optimize data sharing, one cannot ignore a common artifact that has emerged in the bioinformatics software ecosystem. Basically, because bioinformatics software traverses so many different research communities and computational environments, software tools used in such settings have become almost inexorably embedded in their respective research communities. As a result, software tools that become entrenched in one research community are rendered incompatible with the tools preferred by another community. This is so, even though there may be an overlap in goals and methodology between them: tools tend to be associated with particular aspects of computational analysis and particular scientific disciplines for historical (not necessarily technical) reasons, such as **R** for statistics, **PYTHON** for machine learning, **C++** for image analysis and Computer Vision, Matlab for data analytics, or **JAVA** and **JVM** languages for database implementations. These historical factors accentuate the siloing of vendor-specific and/or domain-specific software components, which inhibits integration of heterogeneous data sources derived from multiple institutions, thereby hindering multi-institutional data-sharing initiatives. As a result, data-sharing initiatives are hampered by flaws and incompatibilities in software tools. While some sophisticated open-source bioinformatics libraries have been developed — as in the BioConductor suite (based on **R**) and CytoLib (a **C++ FCS** library) for Cytometry, parsers for **FASTQ** and **FASTA** in many languages, **DICOM** clients such as **DCMTK**, and the Insight Toolkit (a bioimaging segmentation and general-purpose **C++** image-processing suite) — these tend to be relatively difficult to compile and install, with complex chains of external dependencies. Moreover, the tools for different branches of biomedical computations are not particularly interoperable.

To redress data-sharing problems that are caused by software incompatibility, LTS proposes a series of immunoprofiling and immunotherapy software tools that would be based in many cases on existing solutions, but would implement components according to a development methodology which prioritizes interoperability and the minimization of external dependencies. The governing software engineering principles for such a compilation would include:

Prioritize standalone, self-contained components with few external dependencies Avoid using data formats or (as much as possible) numeric or analytic libraries which introduce external dependencies and may diminish interoperability. For example, Google's Protocol Buffers are a popular data exchange format. However — aside from being an added dependency — it is not uncommon for two different components to use incompatible versions of Protocol Buffers, causing software to crash when both components try to interoperate.

Focus on Versatile User Experience (UX) from the ground up Anticipate that data in the system will be viewed by users who are best served with responsive, desktop-style **GUI** components. Assume that most users will interact with this data via native **GUI** applications, rather than with web pages or web applications.

Prefer solutions that can be deployed entirely via source code This often implies code written in **C** or **C++**, because packages in other languages (such as **R** or **PYTHON**) usually include **C** (or sometimes **C++**) libraries that have to be compiled behind the scenes. In these scenarios, LTS's preferred alternative is to distribute the **C**/**C++** directly, with the relevant source files bundled into

whatever application depends on the code library. Source-based deployment also entails avoiding dependencies on libraries which require system-specific build steps, such as "autoconfigure." For example, **HDF5** is a popular scientific data format — used, for instance, by CytoLib, which thereby depends on a **C++ HDF5** parser (specifically, "libhdf"). However, **libhdf** is engineered in such a way that compiling the library requires a pre-compilation configuration step, which means that including **libhdf** source files into application projects is not sufficient to introduce **HDF5** capabilities (or, by extension, CytoLib). Along with Protocol Buffer incompatibility problems (alluded to earlier), these external dependencies are a significant limitation for employing CytoLib as a general-purpose **FCS** parser that can be bundled into software driving immunoprofiling data-sharing projects. CytoLib is, therefore, a good example of a component that should be rewritten as part of an immunoprofiling toolkit.

Develop a multi-disciplinary image-annotation framework While image annotation is essential for describing research powered by image processing, a broad-based image-annotation framework would also have use-cases outside image annotation proper. For instance, the mathematical representations and verbiage of image annotations can also capture dimensional transformations which govern the proper interpretation of **FCS** files. Consequently, image-annotation code can be employed as an alternative to components such as **HDF5** and Protocol Buffers both of which lead to intractable external dependencies in CytoLib.

Use Property Hypergraphs as a generic data-representation model Property graphs (which are graph models with key/value attributes that may be associated with vertices and edges) and hypergraphs (which allow edges to connect three or more vertices) are emerging as popular representation formats for bioinformatics and **AI**. LTS's "**ConceptsDB**" database engine and software-development tools take these industry trends to the next level, synthesizing the benefits of property graphs and hypergraphs.

3.1 A Data-Integration Case Study: Property-Hypergraph Traversal with AngelScript

The University of Pennsylania's Carnival project (which achieves data integration via property graphs) synthesizes heterogeneous biomedical data by translating information from disparate sources into a common property-graph representation and then querying this data with the Gremlin Virtual Machine. Gremlin is a "step-based" virtual machine where "steps" between potential focus elements in a property graph play the role of primitive processing instructions; querying and traversing property graphs involves executing a series of Gremlin steps.[13] Most Gremlin implementations are based on the Java programming language and the Java Virtual Machine (**JVM**), so that queries themselves are written in a **JVM** language (Groovy, in the case of Carnival). LTS's "property-hypergraph" technology uses a similar architecture to query and traverse property-hypergraphs (which are a generalization of both Gremlin-style property graphs and of **HyperGraphDB**-style hypergraphs).[14]

[13] The theoretical foundations of step-based Virtual Machines are presented in Marko Rodriguez, "Stream Ring Theory," February 14, 2019 (https://zenodo.org/record/2565243#.X3vzqS4pDeQ).

[14] The challenge for any database engine which employs a relatively complex data-representation strategy — such as a hypergraph, property-graph, tuple-store (a collection of records with varying numbers of fields) or a multi-dimensional (possibly sparse) array — is to efficiently map the high-level data structures manipulated within the database itself to the lower-level memory units which are stored to disk. Lower-level data structures are typically be modeled via simpler database constructions such as key-value stores, memory cells, or relational tables, so so there must be a translation pipeline between high-level structure (properties, hypernodes, hyperedges, and so forth) and low-level points (record cells, shared memory address, key-value values, etc.) For purposes of demonstration, LTS's property-hypergraph database implementation utilizes **WhiteDB**, an in-memory tuple-store style **NoSQL** engine, for low-level data persistence (*See* http://whitedb.org/). The LTS engine encapsulates access to each **WhiteDB** instance by providing a series of multi-record constructions that encode property-hypergraph formations within groups of **WhiteDB** records. Different class methods allocate, populate, and query these record-groups, although from the client-code perspective the record-groups are treated as integral data structures rather than as aggregates of disparate records. The record-groups provide a tableau, or what the library code refers to metaphorically as

LTS's technology is primarily implemented in C++, rather than Java, which calls for a C++ Virtual Machine for manipulating property-hypergraphs. The relevant computational techniques can be demonstrated via AngelScript, which is a C++-based scripting language. One difficulty when employing AngelScript is that all C++ functions which must be exposed to the scripting engine have to be individually registered, using different procedures depending on the exposed functions' signatures and calling conventions. It would be prohibitively complex to register all functions used by property-hypergraph libraries and emulators directly. Step-based Virtual Machines offer an alternative solution: by factoring graph-traversal into a collection of basic operators, it is only necessary to expose procedures implementing these operators themselves; queries may then be composed by chaining together multiple scripts.[15] Moreover, any C++ class which implements a small group of the most important "step instructions" can emulate a property-hypergraph, in that its data instances can be queried and traversed as if they were represented in memory as property-hypergraphs. This includes classes encapsulating access to database instances, but also classes whose objects represent a single file or file-group, which may be a digitization of lab/diagnostic data — FCS files for Flow or Mass Cytometry, MIT-BIH headers and signal binaries for EKGs, DICOM for bioimaging, and so forth. Via these techniques the Carnival pipeline can be emulated in C++ (with AngelScript playing the role of Groovy) and extended to a broader spectrum of biomedical data (with binary files sometimes replacing SQL tables as data sources).

As illustrated by technologies such as Carnival, a contemporary strategy for biomedical data integration is to translate data-integration problems to graph-traversal problems. To examine this strategy in a more concrete fashion, consider the issues cited earlier with respect to setting up Covid-19 trials. As recommended by Shrestha *et al.*, Covid-19 trials should be designed to focus on specific patient groups which are more likely to benefit from the interventions that form the basis of the relevant clinical trials. Moreover, toward the goal of applying precision medicine to Covid-19 clinical practice, it should be possible to construct a quantifying domain of patient-profile signals (antecedent to trial commencement) to quantify the statistical probability that a given treatment will have a favorable patient outcome in relation to all the prior data in a patient's profile. Since researchers assume that certain factors in a patient's profile will be statistically correlated with favorable outcomes in conjunction with specific treatment plans, part of the trial's purpose is to determine which parts of the patient profile are, in fact, statistically relevant.

In practical terms, then, setting up Covid-19 trials would involve defining patient-selection criteria and implementing systems to screen for patients who may be good candidates for different trials. This would require two steps: (1) constructing a format where trial criteria can be rigorously notated; and (2) implementing software at each participating hospital to search for good candidates to register in each trial. The trial-specific software would need to query each hospital's clinical and/or diagnostic records. Because patient-specific factors for determining trial eligibility would cover a broad spectrum of data types — from sociodemographics and medical history to domain-specific

---

a "stage," where client code can map multiple high-level graph constructions (hypernodes, payloads, properties, and so forth) to persistable record-groups — the engine encodes these high-level constructions to the low-level data types and cross-referenes understood by WhiteDB. A more detailed illustration of the "stage" metaphor involved in this design can be gleaned from the **DW_Stage_Value** and **DW_Frame** classes, as well as the overarching **DW_Instance** class, in the demo repository — see the files within the **@/code/cpp/src/dbk/dgdb/dgdb-white/dgdb-white** folder from the repository's **wip-facs** branch for concrete examples, e.g. **@/code/cpp/src/dbk/dgdb/dgdb-white/dgdb-white/dw-stage-value.cpp**, where **@** stand for the project root folder (created locally when the repostory is cloned). The C++ files can also be browsed online, so for instance the **DW_Instance** header and implementation files can be previewed at https://github.com/scignscape/ntxh/blob/wip-facs/code/cpp/src/dbk/dgdb/dgdb-white/dgdb-white/dw-instance.h and https://github.com/scignscape/ntxh/blob/wip-facs/code/cpp/src/dbk/dgdb/dgdb-white/dgdb-white/dw-instance.cpp.

[15] The techniques described here are concretely demonstrated in the **facs-bridge-hgdm** and **facs-bridge-hgdm-console** projects from the demo repository via the **wip-facs** branch (the full paths are **@/code/cpp/src/xk/facs/facs-bridge-hgdm** and **@/code/cpp/src/xk/facs/facs-bridge-hgdm-console**, where "@" refers to the repository root folder for the source code and **@/code/cpp/cpp/projects/qt/qtm/isobuild/xk/facs/facs-gui/facs-bridge-hgdm(-console)** for the Qt project files). The "console" project is a command-line executable, whereas the larger project is its associated library.

lab/image results — the screening software would, therefore, need to integrate such heterogeneous data sources.

3.2 Coordinating Computational Designs with Clinical Trials for Covid-19

The value of computational methods based on property-hypergraphs in contexts such as Covid-19 is that these graphs can serve as a common intermediate representation for disparate data sources. For example, sociodemographic data would typically take the form of records or "tuples" with a common structure: as such, for each patient there would be a fixed set of attributes such as age, gender, ethnicity, and so forth. So therefore, because such attributes adhere to a fixed schema, they are perfectly suited to be represented in a property-hypergraph as hypernodes. On the other hand, some information within patient profiles would be better represented via conventional labeled graphs or via key-value properties associated with hypernodes.[16]

For instance, patient profiles usually consider the medications each patient is taking — any patient (a node) could in principle be connected to any medication (another node); some patients may be taking *no* medications, others may take just *one*, and some may take *two or more*. Also, connections between patients and medications can be the basis of further details that emerge over time (and are registered in the graph) inasmuch as medications are prescribed to patients by a specific doctor at a specific time, in a specific dosage, in response to specific diagnostic tests, and so forth. In short, information can "fan out" from the patient-to-medication connection in a relatively free-form manner. In general, then, as a subset of overall patient profiles, information about medication evinces the structural features which are, in many contexts, optimally represented via free-form labeled graphs. Meanwhile, patient profiles may also consider medical history, which can be modeled as a graph with detailed logical and temporal inter-node connections. According to this representational strategy, each patient's history is a series of events and observations which are temporally ordered — it is possible to query or traverse the graph in a manner which takes before/after relations into account — and where there are also logical or causal connections defined between nodes. For instance, an edge might assert that a given medication was prescribed to a patient (an event) *because of* the results of a given lab test (an observation).

In these examples, different sorts of clinical data — sociodemographic, pharmacological, medical-history — are modeled according to different sorts of graph structures (hypernodes, nonschematic labeled edges, temporalized graphs). The property-hypergraph model has the benefit of being able to query and traverse graphs with any of these structures. A patient-filtering system for clinical trials could, therefore, be implemented by merging multiple data sources into a common graph space; different parts of this space would have different kinds of constructions (e.g., schematic hypernodes, nonschematic frames around nodes, or causal/temporal inter-node relations). Property-hypergraphs would consequently allow queries to traverse and extract data from subgraphs employing each of these constructions. As a result, the process of selecting trial candidates on the basis of heterogeneous criteria within the "graph space" can be translated to a series of property-hypergraph queries, each query formulated according to the constructions relevant to how a partic-

[16] Computationally, at least in the implementation adopted for the LTS demo, properties are almost exactly akin to directed hyperedges with an annotation-label. Specifically, directed hyperedges are subject/predicate/object triples, where the predicate includes at least a label (a value from some controlled vocabulary of conceptual relations assertable in some domain). Likewise, properties are (in one context) key-value annotations on hypernodes, where the "key" may also derive from a controlled vocabulary. The two constructions (directed hyperedges and properties) play different modeling roles but their computational implementations are similar (the main difference being that "values" in a property-assertion are not necessarily hypernodes in contrast to the "object" in a subject/predicate/object triple). The comparison between their implementations can be seen concretely in the LTS demo; see especially the **populate_edges_or_property_record** method in the **wdb-instance.cpp** file (also the **new_property**, **add_hyperedge_or_property**, or **new_outedges_or_property_record** methods of the **DW_Instance** class). These files are part of the **@/code/cpp/src/dbk/dgdb/dgdb-white/dgdb-white** project, where **@** is the repository root.

ular data source is represented in the common hypergraph format. In short, an overarching data-integration protocol that culminates in algorithms for mapping Covid-19 patients to relevant clinical trials for which they are considered as good candidates, can be defined through a series of property-hypergraph queries. Property-hypergraph queries in this system would serve as high-level specifications that can be custom implemented for each hospital participating in a trial.

Given the sheer scale of the SARS-CoV-2 pandemic, there are likely to be many candidates for almost any Covid-19 trial. However, because of the extent of the pandemic, governments and hospitals have had to establish treatment protocols rather haphazardly. As a result, there has been a fair amount of trial and error as opposed to a *systematic* framework for evaluating diagnostic regimens and treatment protocols. For example, rather than have one or two canonical SARS-CoV-2 antigen tests, used to measure virus antibody levels in patients, the US Centers for Disease Control recommends or has authorized a large list of assays performed by many different companies, using many different biochemical methods.[17] Because the format of data resulting from immunoassays depends on the specific biochemical mechanism which (within each assay) yields quantitative data, a broad spectrum of antigen tests requires a diverse array of data formats which need to be integrated. As such, whenever Covid-19 immunoassays are considered as critical factors in immunological profiles for mapping patients to appropriate Covid-19 clinical trials, querying for good trial candidates means querying across a wide spectrum of structurally different data types that correspond to this broad array of antigen tests — specifically to the mechanisms through which laboratory instruments generate quantifiable data and to the computational procedures which process such data. Here, is a good example of why specialized custom software is necessary: given the heterogeneity of Covid-19 data, synthesizing all this information calls for integrative procedures implemented directly in programming languages such as **C++** (rather than query languages such as **SQL**). Though property-hypergraphs would not obviate the need for special software, they could, nevertheless, make the initialization of custom data types and the engineering of integrative queries more convenient than alternative graph-database models.

3.3 Introducing New Software Alignment Tools for SARS-CoV-2 Phylogeny Studies

Similar to immunological studies, discussed above, analysis of SARS-CoV-2 mutations is yet another area where software alignment can prove to be important. Studies suggest that variations in the viral strain causing Covid-19 symptoms may be partly responsible for divergent immunological responses to the virus across the patient population.[18] If one patient responds either less favorably or more favorably than the average patient-response to a given treatment, clinicians need to assess whether this difference can be explained solely by the patient's prior immunological profile or whether the patient has been exposed to a genetically divergent viral strain.

Modeling SARS-CoV-2 evolution across the globe is a massive project. There have been over 41 million Covid-19 cases worldwide, in virtually every nation on earth, so a complete phylogenic picture of SARS-CoV-2 in humans would need to pool data from many different healthcare systems. Yet, even technically detailed analysis of the phylogeny of SARS-CoV-2, such as that conducted by Dearlove *et al.*, as reported recently in the *Proceedings of the US National Academy of Sciences*, only considered 27,977 patients (about 0.1% of global cases), with almost half from the United Kingdom.[19]

---

[17] *See*, for example, https://www.fda.gov/medical-devices/coronavirus-disease-2019-covid-19-emergency-use-authorizations-medical-devices/vitro-diagnostics-euas#individual-antigen.

[18] *See* for instance Osman Shabir, M.Sc., "The Phylogenetic Tree of the SARS-CoV-2 Virus," News Medical, https://www.news-medical.net/health/The-Phylogenetic-Tree-of-the-SARS-CoV-2-Virus.aspx; the author identifies one region in particular where "[c]hanges within this region may account for the differences in immune responses to SARS-CoV-2."

[19] Bethany Dearlove *et al.*, "A SARS-CoV-2 Vaccine Candidate would Likely Match all Currently Circulating Variants," *Proceedings of the National Academy of Sciences of the United States of America*, August 31, 2020, https://www.pnas.org/

Hence, achieving something resembling a holistic picture of SARS-CoV-2 mutations and how they might affect clinical treatments, would require many, many parallel studies analogous to that of Dearlove *et al.* This then raises critical questions of study alignment: this is so, because calculating viral phylogeny requires making technical decisions about how genetic sequences should be acquired and analyzed, decisions which may vary among research teams, spanning many healthcare institutions. For example, the authors of this National Academy of Sciences study outline several computational steps which they had performed both to normalize each SARS-CoV-2 genome sequence in their data set for cross-comparison and to run predictive simulations (used to estimate whether divergence between sequence-pairs are the result of localized, random mutations or, conversely, an indication that SARS-CoV-2 is evolving into several distinct strains). This and other work suggests that there are currently three genetically distinct SARS-CoV-2 strains around the world (Types A-C). Observations up to the current moment suggest that Types A-C are similar enough that Dearlove *et al.* remain "cautiously optimistic that viral diversity should not be an obstacle for the development of a broadly protective SARS-CoV-2 vaccine" (p. 9). However, such assessments are provisional, partly because one cannot predict how the virus will evolve in the future but also because of extant limitations in how scientists acquire and analyze data from past and present Covid-19 cases. Clustering SARS-CoV-2 genomes into "strains" — that is, identifying which mutations are random and which appear to be propagating to subsequent viral generations — involves making computational and biological assumptions, such as how to statistically marshal genomic data so as to quantify the prevalence of a mutation, and how to estimate whether a particular mutation confers an adaptive benefit to the viral agent (e.g., an ability to elude antibodies targeting structural proteins).

Given that modeling viral phylogeny requires certain computational assumptions and biological guesswork, data from multiple studies can only be reliably integrated if there is some degree of alignment across their methodology. As such, research teams should document their protocols in a manner that permits assessment as to whether protocol differences might compromise the resulting data. One way to achieve this is to model the protocol itself as a datatype in a programming language such as **C++**. For each study, such as Dearlove *et al.* cited above, there would then be a **C++** object encapsulating all details of the researchers' protocols and computational workflows. Assuming that research into SARS-CoV-2 phylogeny could be centralized into a publication archive, then their associated protocol-objects could be aggregated into a protocol database. The purpose of such a database would be to establish a composite picture of SARS-CoV-2 evolution by synthesizing data from multiple studies and/or multiple countries. Protocol-alignment would be one part of a common framework to quantify the epidemiological significance of SARS-CoV-2 mutations.

Thus far, only a few SARS-CoV-2 mutations have warranted further analysis, assigned names such as "P4715L" and "D614G" (Dearlove *et al.*, p. 4). Several other mutations are responsible for the differences between Type A-C SARS-CoV-2 strains. In short, a holistic global picture of SARS-CoV-2 must represent SARS-CoV-2 mutations which have been deemed phylogenically and/or clinically significant (i.e., having potential either to influence the overall evolution of Covid-19 and/or to have some bearing on clinical treatments), and must *also* represent divergent SARS-CoV-2 strains. These data-points would then be the basis of further details such as: when did a given strain and/or mutation first appear? Is the strain/mutation geographically localized? What is the proportion of different strains/mutations in a geographic area? Is there evidence that different strains/mutations affect a patient's immunological response to Covid-19 and/or the effectiveness of vaccines, antibody regimens, steroids, or other clinical interventions? How can genetic mutations within the SARS-CoV-2 virus be correlated with structures in the spike proteins encoded by the viral genes? This last question points to the importance of integrating genomic data with **3D** molecular models.

content/117/38/23652.

Whereas data structures modeling the viral genome are composed of nucleotides — and, at a higher scale, Open Read Frames (**ORF**s) — data structures describing the biophysics of glycoproteins involve **3D** geometry and chemical bonds. Analyzing how SARS-CoV-2 genes affect the production of glycoproteins, therefore, requires annotating and cross-referencing nucleotide/**ORF** data structures with molecular models encoded in formats such as Protein Data Bank (**PDB**).

Assuming that a synthesis of SARS-CoV-2 genomic research is based on a **C++** protocol object model, as suggested above, one can similarly develop **C++** data types to model SARS-CoV-2 strains and mutations. The trio of protocol, strain, and mutation objects could then serve as the basis of a unified Object-Oriented framework for representing SARS-CoV-2 evolution across the world. This would allow the results of different studies to be compared, so as to build a composite global profile of SARS-CoV-2 phylogeny. In fact, a novel "Covid Phylogeny Object Model" (**CPOM**) could serve as a nexus for merging temporal and geographical data concerning the epidemiology of SARS-CoV-2 mutations with genomic data demonstrating which mutations are significant, as well as clinical data tracking correlations between mutations and treatment outcomes. Supporting large-scale multi-trial data integration would, therefore, also introduce new requirements for clinical trial software.

4 The Need to Customize Clinical Trial Management Software

While there are several Clinical Trial Management Systems (**CTMS**s) on the market, custom-built software can provide flexibility to support innovative, multi-faceted trial designs. In this context, **CTMS** applications may be used both for controlled trials designed according to government regulations (such as double-blind clinical trials) and for more informal observational studies, some of which are conducted under data-sharing initiatives. In either case, the total data generated in the context of a trial/study will usually go beyond Clinical Research Network models and Electronic Case Report Forms (**eCRF**s), which are the basis of conventional **CTMS** software. Specifically, most trials involve diagnostic and outcomes assessments which rely on some form of lab analysis and specialized software. As noted earlier, trial data proper often incorporates only summarial reviews of lab findings, thereby excluding detailed lab reports. However, this potentially inhibits access to scientifically valuable data. For instance, though it is crucial in the monitoring of SARS-CoV-2 genetic diversification to extract and analyze viral DNA from the blood samples of Covid-19 patients wherever possible, summarial trial reports are wholly devoid of this data. This implies that any Covid-19 trial should, ideally, include a mechanism for identifying patients' SARS-CoV-2 strain and mutations, and subsequently submitting this data to a central repository. Such SARS-CoV-2 genomic data would then become part of the trial's overall data profile, cross-referencing viral-genomic analysis of patients' blood samples with other patient-specific data, such as immunological profiles.

As we can see, once trials embrace heterogeneous data models (which require special-purpose software for accessing some of the trial data) **CTMS** requirements become more complex. In these situations, **CTMS**s may need to model and in some cases replicate complex computational workflows, such as those employed by Dearlove *et al.* for calculating SARS-CoV-2 genomic sequences from patients' blood samples. The **CTMS** software may also need to interoperate with domain-specific applications, as in bioimaging and image analyses, signal processing (e.g. for **EKG** analysis), Flow Cytometry, biochemical assays, genomic analysis, epidemiological modeling and so forth. If possible, such applications should be configured or extended to work with the clinical trial software. For instance, if a **DICOM** (Digital Imaging and Communications in Medicine) client is used to study an image derived from a specific trial — e.g., a radiological scan of a Covid-19 patient's lungs — the **DICOM** software could be provided with a plugin that would show trial information in a separate window, which could then be juxtaposed with the main-image view.[20] In this context, software

[20] A brief graphical example of how **DICOM** plugins can enhance bioimaging technology is included in an LTS White

alignment means that all institutions participating in a trial could use the *same* plugins, so that the trial's central **CTMS** system could interoperate with special-purpose software in a consistent manner. This would also aid in establishing, as part of the trial design, protocols for depositing special-purpose data assets (such as **FCS** or **DICOM** files) alongside clinical data and **eCRF**s.

A further benefit of **CTMS** customization is that custom software adds flexibility for trial design. By definition, trials allow researchers to test some biomedical hypothesis in a controlled manner. Trials are, therefore, defined around the premise that observational information resulting from the trial is empirically significant, revealing something new about what the trial was designed to investigate. For instance, a Covid-19 trial might assess how well patients with varying prior immunological profiles respond to monoclonal antibody (**mAb**) treatments. The relevant observations in this case derive from the subsequent course of the disease for each patient, as well as potential adverse reactions, but there are, however, inherent complicating factors, such as: (1) were patients receiving other treatments as well; (2) for patients who recover, how do we know that the antibodies expedited that recovery; (3) how quickly was the recovery; and (4) did patients continue to suffer from Covid-related symptoms even when they were no longer infectious.

In addition to these post-treatment observations, situational details specific to the trial — such as each patient's **mAb** dosage level, prior Covid-19 risk factors, or viral-load change over time — need to be incorporated into the trial's unique data models. Moreover, a comprehensive investigation could well incorporate both information about the patient's unique immunological profiles and the nature of the SARS-CoV-2 variant/strain found in the patient. Patient-centric data could likewise include sociodemographic information about the patient, supplemented by critical epidemiological metrics, such as contact tracing: was the patient living or working in an environment where they were likely to have been exposed to the virus? If not, how did they contract it?

4.1 Why Flexible Data Modeling is Crucial for Trial Design

Flexibility in trial design is important because the correct protocols for modeling and integrating both pre-treatment and post-treatment data need to be worked out for each trial, depending on the trial's goals and logistics. Designers need to identify, for example, what dimensions of patients' immunological and sociodemographic profiles are likely to be consequential when analyzing treatment outcomes. It is important for trial designers to model sociodemographic data attentively. Indeed, biomedical research has increasingly been criticized in recent years for bias toward certain populations (e.g., white, middle-class non-seniors), leading to the unavoidable consequence that certain racial, age, and socioeconomic categories will be vastly under-represented in trial cohorts. This results in uncertainties as to how well trial results carry over to populations at large — that is, populations characterized by a heterogeneous mix of demographic factors. Trial designers can mitigate these concerns by demonstrating sociodemographic diversity among trial participants.

Demonstrating sociodemographic diversity, however, calls for transparency about how sociodemographic details are represented. The process of grouping patients into ethnic/racial and/or socioeconomic strata can be equivocal at times. For example, if a trial participant is a graduate student at the University of Chicago, should their socioeconomic status be assessed on the basis of their own income or that of their parents? If their residential zip code places them on the South Side of Chicago, a region with both a prestigious campus and pockets of extreme poverty, should they be demographically classified alongside permanent residents of their neighborhood? What about a varsity linebacker who appears to be in excellent health before a Covid-19 infection? Should his status as an athlete be taken to indicate being extremely fit prior to the disease, or might his back-

Paper at https://raw.githubusercontent.com/Mosaic-DigammaDB/CRCR/master/cr2.pdf; see the first two figures.

ground as a football player intimate a potential history of brain trauma which may compound his neurological damage due to Covid-19?

Similar issues of interpretation also apply to post-treatment observations. How should researchers decide which observations qualify as clinically significant consequences of Covid-19? We have seen that as the pandemic has unfolded, a fair number of cases have been cited in the professional literature describing Covid-19 patients who suffer certain cognitive/neurological effects, such as muscular fatigue or weakness, mental confusion or poor concentration (sometimes referred to as "brain fog"), or symptoms of Guillain–Barré syndrome spectrum. Almost certainly, some of these symptoms may be the product of cognitive/neurological effects due to SARS-CoV-2. At the same time, Covid-19 patients — even those who fight off the infection successfully or who test positive but remain asymptomatic — may find their lives so disrupted by the pandemic that this may (indirectly) cause cognitive and neurological problems. For instance, prolonged inactivity (for a typically active person), which commonly occurs as the result of a quarantine, may contribute to poor concentration and other diminished forms of mental acuity. Given the fact that most people's lives during the pandemic are not "normal," it may be difficult to establish which symptoms experienced by a patient are actually biological effects of the disease itself or, alternatively, indirect consequences of quarantine restrictions put into place to help stem the tide of the pandemic. This sort of ambiguity also applies to potential adverse side-effects of Covid-19 treatment. We need to ask the question how long after a treatment is administered should a patient's symptoms be considered potential side-effects of the therapy itself or, alternatively, the result of nagging uncertainties and a disrupted lifestyle imposed by the pandemic.

The fact that these questions have no predetermined answers indicates that reporting protocols, which give some structure to this kind of amorphous and *ad hoc* patient data, need to be considered as an integral part of trial design. Specific parameters should be established for the posttreatment and post-recovery window of time, where patients' symptoms might be noted as potential effects of the disease or of the administered therapies themselves; moreover these symptoms should be verified and classified according to the trial design itself lest they could be lost altogether from the medical reporting process. The methodology for doing so should be rigorous but also flexible, because it is hard to predict *a priori* the range of patients' responses to both diseases and treatments. For example, because SARS-CoV-2 was initially believed to affect lung functioning primarily, the risk of long-term cognitive/neurological damage was not widely anticipated when considering treatment over the course of Covid-19 infection. Consequently, because tests of a cognitive/neurological/physiological/radiographic (except for basic lung scans, with respect to radiology) had not been a common facet of early Covid-19 trials/observational studies, there were obviously no corresponding data structures included in those studies for capturing this full spectrum of neuropsychological and neurological data.

Though patient-histories for systematically tracking Covid-19's cognitive/neurological effects are very important, the complexity of this illness, nevertheless, calls for manifold methods of data aggregation rather than rely exclusively on patients' own subjective symptom descriptions. As such, patient complaints of cognitive and neurological deficits would entail the use of neurological, laboratory, neuropsychological and radiographic tests to understand the full extent of their cognitive and neurological impairments. For example, the use of **MRI**s when considering Covid-related ischemic stroke,[21] the use of electrophysiological tests, cerebrospinal fluid tests (**CSF**), or the **MRC** (Medi-

[21] *See*, for example, Johanna T Fifi and J Mocco, "COVID-19 related stroke in young individuals," *The Lancet*, Vol. 19 No. 9, September 2020, https://www.thelancet.com/journals/laneur/article/PIIS1474-4422(20)30272-6/fulltext, and Ross W Paterson *et al.*, "The emerging spectrum of COVID-19 neurology: clinical, radiological and laboratory findings," *Brain*, July 8, 2020, https://academic.oup.com/brain/advance-article/doi/10.1093/brain/awaa240/5868408.

cal Research Council) Scale for muscle-strength test when considering Covid-related Guillain–Barré symptoms,[22] or the use of neuropsychological tests, such as Trail Making Test (**TMT**), Sign Coding Test (**SCT**), Continuous Performance Test (**CPT**), and Digital Span Test (**DST**) — which measure a patient's executive abilities (letter and number recognition mental flexibility, visual scanning, and motor function) and sustained and selective attention, along with other cognitive and neurological functioning — when considering cognitive/neuropsychological impairments after a serious bout with Covid-19.[23] All in all, these cognitive/neurological/physiological/laboratory/ radiographic data structures thereby become an integral part of the information relevant to trial evaluation, because they document crucial symptoms which are presumptively attributable to Covid-19. However, in practice, prior to clinicians having been alerted to the fact that Covid-19 may cause lingering cognitive/neurological damage, Covid-19 trials were not designed to incorporate neurological or cognitive data in a systematic manner. This scenario points to how trial data models in Covid-19 studies must have the built-in capacity to be redesigned impromptu while the trial is ongoing so as to accommodate new and emerging information, such as debilitating cognitive or neurological impairment, as discussed above.

In fact, any trial-related data should be considered a starting point for subsequent analysis, which will render the data more precise and fine-grained. What this does is augment the data's value while also refining the trial's data model. Fully accounting for all these details, such as impaired cognitive and neurological functioning as in the case above, may not be feasible during an initial analysis of trial data. The point is not that trials should be designed with a goal of maximal precision in all aspects of data acquisition and interpretation. Instead, trial data should be curated in a manner which facilitates subsequent analysis, by introducing incremental levels of precision one at a time. Trial data should, therefore, be modeled within a framework which supports the relatively free-form aggregation of new information and new observations, such as a graph database, rather than the rigid schema of relational tables *ab initio*. In keeping with the way in which clinical data emerges in an *ad hoc* way during the longevity of a clinical trial, custom **CTMS** software would then allow each trial to expose its data models in a procedural, Object-Oriented fashion, using trial-specific software as a Reference Implementation illustrating how trial data is structured and curated.[24]

---

[22] *See* Samir Abu-Rumeileh, Ahmed Abdelhak, Matteo Foschi, Hayrettin Tumani, and Markus Otto, "Guillain–Barré syndrome spectrum associated with COVID-19: an up-to-date systematic review of 73 cases," *Journal of Neurology*, July 2020, https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7445716/.

[23] *See* Hetong Zhou *et al.*, "The landscape of cognitive function in recovered COVID-19 patients," *Journal of Psychiatric Research*, October 2020, https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7324344/.

[24] Similar to cognitive/neurological data, sociodemographic data, likewise, can be notoriously imprecise. It is well-known that patients of lower socioeconomic status have had higher Covid-19 infection rates and mortality rates than patients of higher socioeconomic status, but this disparity may be explained by several causal factors: more infectious workplaces, inferior post-infection treatment, poorer state of health before the onset of Covid, and so forth. Teasing apart these factors demands fine-grained analysis of the individual patient's pre-Covid history; sociodemographic generalizations can exclude important details. For example, for purposes of analysis, in the case of a graduate student with middle-class parents but no health insurance of their own, how should we quantify their degree of access to health care? How well does geographic location serve as a proxy for socioeconomic status? The case of a middle-class student living in a well-off near-campus corner of an otherwise impoverished neighborhood suggests that geospatial metrics (such as zip code or congressional district) are imperfect proxies for wealth; but other factors — such as air/water pollution or the risk of being the victim of a crime — may be statistically correlated among geographically proximate residents even if they have otherwise divergent sociological profiles. These examples illustrate that the value of sociodemographic data is proportionate to the level of statistical detail with which the data may be analyzed. Instead of a broad and vague designation, such as "low income," one may want to derive more detailed subclasses, incorporating information about patients' employment, access to health care, physical fitness, and so forth. Two patients with similar income levels, for instance, may have different levels of access to health care (depending on factors such as whether the patients have employer-based insurance or geographic proximity to healthcare facilities) or different levels of exposure to Covid-19 in the workplace, depending on the nature of their job. Even if a trial does not quantify granular sociodemographic assessments — such as patients' workplace conditions and access to health care, which might serve as a more accurate estimation of how socioeconomic status causally affects disease outcomes — subsequent researchers may determine ways to analyze or add on to data generated by a trial (e.g., through follow-up studies of enrolled patients), so as to make such sociodemographic granularity part of the quantifiable framework.

4.2 Representing Trial Data in Terms of Object Models for Software-Application Flexibility

As a concrete example of data-modeling principles (outlined in the prior two sections) consider how the novel Covid Phylogeny Object Model (**CPOM**) could serve as a nexus for integrating SARS-CoV-2 phylogenic data across multiple studies and healthcare systems. Such an Object Model may be extended in different ways for different clinical trials examining a whole range of Covid-19 treatments. For a given antibody regimen, for instance, scientists need to quantify how well the antibodies disable Covid-19 spike proteins directly and/or how well the antibodies block the virus's ability to attach to human cells. These measurements generate data which indicate how well a patient's immune response to Covid-19 is boosted by the administered antibodies, information which is usually delivered in special-purpose formats such as **FCS** or fluorescence images. A Covid-19 software ecosystem would then need to ensure that such immune-response data can be effectively parsed and integrated into Object Models describing how SARS-CoV-2 is evolving around the globe. Such data integration modeling would allow researchers to reliably assess in each individual patient their immune response to the particular acquired SARS-CoV-2 strain/variant vis-à-vis their personal immunological profile, and to competently track both current and emerging SARS-CoV-2 strains throughout the population.

Custom trial software allows trial data models and data integration to be effectuated via Object-Oriented programming techniques. Representing trial data in terms of an *Object Model* — as opposed to a tabular schema or an **RDF** ontology — provides greater flexibility in how data is represented and analyzed. This flexibility is *sine qua non* for a clinical trial to be able to easily accommodate to the *ad hoc* emergence of new parameters in the trial, such as the need to use contrast-enhanced spinal imaging to diagnose Guillain-Barré Syndrome in Covid patients who experience neurological/neuromuscular symptoms during the course of the illness. Having this kind of flexibility built right into the data models allows such models to evolve over the course of clinical trials. This is important because given that clinical trials may have fairly extensive longevity this would leave the door open to many new clinical findings which are naturally hard to predict in advance. However, Object Models need to be concretized in software, which is why trials that employ Object-Oriented data models also need to implement custom **CTMS** systems. LTS data-integration tools, which are based on procedural hypergraphs, provide a solid foundation for working with inherently adaptive Object-Oriented applications, namely because the procedural-hypergraph model has enough flexibility to accommodate the data serialization and data-manipulation requirements of Object-Oriented software.

These sorts of flexible Object Models can then facilitate trials designed according to the novel protocol proposed by Shrestha *et al.*, discussed above, where each trial would study a preselected (non random) cohort of patients for whom both pre-treatment immunoprofiling data and post-treatment outcomes data would be available, so as to compare multiple trials against one another. This would be a novel contribution because in conventional trial design pre-treatment data is less likely to be made available inasmuch as clinical trials do not commonly entail the use of custom software for analyzing pre-treatment data vis-à-vis post-treatment outcomes. As a result, Object Models customized for each trial would generate a data framework through which the causal relations between patient profiles and treatment outcomes would be investigated. Customized trial software would, accordingly, provide a Reference Implementation demonstrating each trial-specific Object Model. If adopted for multiple trials conducted across multiple clinics/hospitals, this software ecosystem could serve as a significant step forward in the fight against Covid, because the trials' data models may help doctors better understand which aspects of patient profiles are particularly significant when matching Covid-19 patients to the most salutary treatment.

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