WHITE PAPER

Next-Generation Data-Integration Tools for Precision Medicine

Improving Personalized Immunotherapy
and Patient-Outcomes Research

Executive Summary

LTS is developing a suite of software libraries centered around 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 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.

Precision medicine is based on the scientific realization that the effectiveness of a certain clinical intervention — for instance, immunotherapy as a cancer treatment — is heavily dependent on factors that vary significantly between 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 (prior to treatment), (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.

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 pro-

[1] Further discussion of our data-integration tools can be found in our White Paper introducing our novel Hypergraph Data Modeling Protocol at https://raw.githubusercontent.com/Mosaic-DigammaDB/CRCR/master/hgdm/HGDM-a-new-protocol.pdf and in our protocol definition paper at https://raw.githubusercontent.com/Mosaic-DigammaDB/CRCR/master/hgdm/HGDM-protocol-definition.pdf.

grams 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 and immuno-oncology 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.

The Need to Design Data-Sharing Tools in response to the Interdisciplinary Nature of Immuno-Oncological Data

Granted, some of the data related to immunological profiles may be sociodemographic or part of a patient's medical history — which is the kind of data that can fit within conventional Clinical Research Network models. That kind of data is distinct from the granularity of contemporary immunoprofiling which is powered by diverse kinds of highly specialized diagnostic equipment, 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 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.[2]

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 outlined in the table above, then, immunological profiles draw on a diversity of data formats and lab/data-acquisition modalities (this table is not intended to be a complete list of criteria or

[2] 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).

file formats, but rather to indicate the range of diagnostic technologies and data formats which are relevant to immunoprofiling).

As evidenced in the table, immunotherapy/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, researchers, and/or laboratories, because institutions are known to have their own indigenous software.

Data-Sharing Solutions for Immunoprofiling and Covid-19

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 simplified overviews, with brief summaries of diagnoses, treatments, and/or outcomes — without 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; 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 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. [O]ne way to address this 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 according to Friedensohn *et. al.* "introduce errors and bias" that contaminate the aggregate data-sharing results.

[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 title "B-Cell biology"), June 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.

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." The authors argue that "[i]ncorporating predictive enrichment strategies can help to identify and thus target specific phenotypes, potentially raising the possibility of positive trial outcomes [while] enrichment of studies can be done by selection of patients with a strong likelihood of a response to an intervention, thereby reducing study noise, sample size, and study-associated harm."[4] The authors go on to cite several factors — factors which are highly variable among patients — that can serve as criteria for the kinds of therapeutic interventions most likely to be beneficial to the patient. These criteria range from cardiovascular factors (measured by **EKG**s) to immunoprofiling factors — such as elevated levels of the Interleukin-6 cytokine — that are identified by blood analysis and biochemical assays. Shrestha, *et. al.* call for "large multicenter trials." They base this recommendation on the fact that there are "many patients that meet eligibility criteria for individualized therapies." This creates, according to the authors, "the potential of getting large cohorts of patients in a shorter time period." They further argue: "A robust data infrastructure developed by combined efforts of clinicians, researchers, and data scientists [can] facilitate '1-stop shopping' at the point of care for the evaluation of different therapies [and] an attractive and appropriate option at the present context." The authors make the point that "the scientific community [should] unite and meticulously design trials fostered by the concept of precision."

Assuming Shrestha and coauthors' advice were heeded, Covid-19 trials would be structured in part by precision-medical profiles of Covid-19 patients which would steer different patients into different trials based on crucial patient-specific data. Rigorous immunological, cardiovascular, and general medical-history data would therefore be obtained for each patient prior to their involvement in a trial, which implies that such data should also be available for subsequent analysis of trial results. Consequently, Covid-19 data-sharing initiatives should include fine-grained evaluations such as waveform analysis of **EKG** data, or cymtometric/chromogenic/flourogenic calculations (or their variations) for antigen assays.[5] Covid-19 data sharing should therefore identify data structures which convey detailed lab findings — for example, biochemical fluorescence charts or **EKG** signal-processing feature vectors. In the context of **EKG** signals, Lyon *et. al.* summarize that "[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."[6] 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.

Fostering Workable Software Alignment across Multiple Hospitals and Laboratories

Assuming that a data-sharing network spans multiple hospitals — and multiple scientists or programmers in each hospital — the software which powers each initiative may easily need to be run on dozens or hundreds of different computers. Each hospital might enforce restrictions on how software is installed. In general, 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 subfolders). All the software which powers

[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] 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).

[6] *See* Aurore Lyon, *et. al.*, "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/).

the initiative should therefore 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, the software powering data-sharing initiatives should be self-contained, with few or no external dependencies, and should run smoothly in a "sandboxed" environment — an environment that is isolated from other data and applications on the host computer. Consequently, developing software components which adhere to these strict requirements demands a conscientious development methodology, one which can get beyond the hiderances that stem from conventional practices. That is, the majority of open-source biomedical/bioinformatics software, while often scientifically sophisticated, adheres to software-development paradigms that limit their usefulness in data-sharing contexts: these projects do not adequately prioritize minimizing external dependencies, and they are often relatively difficult to compile and install. Unfortunately, sophisticated software which could actually add information granularity and **UX** enhancements to data-sharing initiatives often gets excluded. The inevitable consequence of excluding *software* is that potentially valuable *data* frequently gets excluded as well.

Despite the importance of fine-grained data-sharing — for instance, including certain raw lab data into data-sharing initiatives — data-sharing at this level is mostly hindered by the lack of software components that can be used outside the laboratory proper. Diagnostic and research laboratories tend to require expensive equipment which connects to special-purpose, vendor-specific software. Although such software may be a prerequisite for initial data acquisition and analysis, in the context of multi-disciplinary/multi-institution initiatives spanning dozens of hospitals and research sites it is impractical to expect each location to host commercial software for each immunoprofiling-related discipline. However, LTS addresses these constraints by noting that software which is shared across a data-sharing initiative need not fully replicate the capabilities of the initial data-acquisition software. Instead, each initiatives' shared software can have functionality focused 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 should focus on components that can be widely *shared* and *reused*, not just on optimizing performance.

It is very common in scientific computing 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 which 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 need periodic maintenance. In short, installing and using programs such as Docker or **VMWare** potentially alters the computers where virtual or containerized software is executed to a noticeable extent.

LTS, by contrast, has worked on curating biomedical and bioinformatics tools which are sandboxed by virtue of development methodology rather than by virtualization. These tools run as normal applications within the normal operating system; however, 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. These 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 are not necessarily 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 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.

Challenges for Immuno-Oncology Data Sharing

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. Questions 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.

The problem of software alignment requires particular attention, because — despite a pervasive biomedical emphasis on clinical trials and multi-site data-sharing — the overall field of bioinformatics software is spread across numerous different research communities and computational ecosystems. As a result, software tools tend to become entrenched in one research community, often rendering them incompatible with the tools preferred by other communities. 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.

Proposing Software Engineering Principles/Tools Optimized for Data Sharing

To redress data-sharing problems that are caused by software incompatibility, LTS proposes a series of immunotherapy and immunoprofiling software tools that would be based on existing solutions, in many cases, 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 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 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.

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.[7] 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).

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 function 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. 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 **EKG**s, **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).

Introducing Property-Hypergraph Traversal in Practice: Covid-19 Examples

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 whose effectiveness is investigated by the relevant clinical trials. Moreover, toward the goal of translating precision medicine to Covid-19 clinical practice, it should be possible to construct a space of patient-profile signals (antecedent to trial commencement) so as to precisely quantify the intervention's favorable-outcome probability (relative to patient-profile priors) based on the trial data.

A plausible objection to the Shrestha *et. al.* recommendations is that preselecting patients for likely success biases the trial results themselves, because patients are not selected on a random basis. This may be a benign bias, transparently built into the trial design, but it still would seem to limit the trial's value because it effectively negates the presence of a control group. To counter this

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

objection, note that Shrestha *et. al.* propose that researchers establish a suite of trials, each focused on a given sort of Covid-19 treatment. In fact, so long as these trials collectively can be compared and analyzed, each trial could serve as a *de facto* control for all the others: it is possible to measure how well one treatment yields positive results *for its target cohort* against how other treatments work for *their* target cohorts respectively. In addition, sufficiently granular patient profiling prior to each trial can supply some level of variation among trial participants. Even if trials are targeted at selected groups of patients whose profiles are similar in some ways, these profiles will not be identical: the trial results can still permit analyses mapping profile indicators to the most favorable outcomes. Combining the results of multiple analyses, one can therefore potentially concretize statistical methods to match patients with recommended treatments given a range of profile factors (which could include immunoprofiling data, demographics, medical history, and so forth). Not every trial may consider every available factor, but merging multiple trial results can result in a larger statistical framework which encompasses many factors.

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 would cover a broad spectrum of data types — from sociodemographics and medical history to domain-specific lab/image results — the screening software would therefore need to integrate heterogeneous data sources.

The value of computational methods based on property-hypergraphs in this context is that such 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: for each patient there would be a fixed set of attributes such as age, gender, ethnicity, and so forth. Because such attributes adhere to a fixed schema, they could be represented in a property hypergraph as hypernodes. On the other hand, patient profiles may also consider the medications each patient is taking, which would be more likely to be represented in the manner of an **RDF** graph — any patient could be taking any medication, so the optimal structure for representing this information is a graph whose nodes are patients and medications, and whose edges connect patients to the medications they are taking. 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 taking 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 par-

ticular 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 clinical trials for which they are 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 (when needed) 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 had to establish diagnostic and 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 treatment protocols. For example, rather than have one or two canonical SARS-CoV-2 antigen tests, 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.[8] 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 to be part of immunological profiles which are factors in mapping patients to Covid-19 trials, querying for good trial candidates means querying across a wide spectrum of structurally different data types. This is a good example of why specialized custom software is necessary; the range of data types associated with commonly used antigen tests is far too heterogeneous to be accessible to **SQL** or **RDF** queries. Instead, to accommodate the heterogeneity of data for Covid-19, data integration must be effectuated with custom data types and integrative procedures implemented directly in a programming language such as **C++** (rather than a query language such as **SQL**). Though property hypergraphs would not obviate the need for special software, they could make the initialization of custom data types and the engineering of integrative queries more convenient than alternative graph-database models.

As a final comment, note the potential benefits of custom software implemented specifically for each trial, particularly in a context as proposed by Shrestha *et. al.*, where multiple Covid-19 trials are run independently but will ultimately be interconnected by meta-analyses. According to this strategy, clinical decisions regarding which patients are registered with each trial will often coincide with a choice of treatment plans for each participating patient. This implies that the goals and protocols for every trial need to be clearly understood by medical professionals working directly with patients at the point-of-care. This is an area where custom software can help. Software for each trial would presumably be implemented while trials are being planned and organized (although the software may be refined in response to observations vis-à-vis trial data). By coordinating computational designs with clinical trial designs, doctors and researchers would be able to evaluate the trials' **GUI** components, selection criteria, data models, and research protocols, where these are all rigorously expressed in computer code. In general, familiarity with trial design — in particular, the personalized criteria for enrolling patients — can help doctors understand which aspects of patient profiles are most significant when matching Covid-19 patients to most beneficial treatments. These software components could then provide extra rigor and clarity to help doctors become familiar with trial goals and methods, leaving them sufficiently well-informed to make point-of-care (where the patient is physically located) clinical decisions "in real time."

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

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