Data Integration for Systems Biology

by

Elizabeth D. Trippe

(Under the Direction of Juan B. Gutierrez)

Abstract

Bioinformatics is an integral part of systems biology studies, yet many large scale multi-omic studies fail to produce meaningful, actionable results. These experiments produce data sets that are often massive, contain many different types of data in a variety of formats and are often analyzed with a specialized set of tools. Also, scientific and clinical studies often incorporate data sets that cross multiple spatial and temporal scales to describe a particular phenomenon. In this work, these challenges are addressed though the development of a novel analytical framework, Scientific Knowledge and Extraction from Data (SKED), incorporating standardized quantitative data formats, called data primitives, and an extensible object-oriented schema to manage analysis steps. The SKED framework was used to manage analysis of diverse data types from different hosts (three species of non-human primates) and tissues (whole blood, bone marrow, and blood plasma) to investigate molecular mechanisms and interventions to promote host resilience to *Plasmodium* infections. Molecular targets that may influence the host response, as well as FDA-approved modulators of these targets, were identified using information from the Pharos database (from the Illuminating the Druggable Genome project) and the Drug-Genome Interaction database. One of these modulators, imatinib, is known to have multiple targets, which were also found here, and the evidence supporting the re-purposing of this drug to promote a resilient host response is presented. This work shows that the SKED approach is able to produce biologically meaningful and verifiable results. The SKED framework is flexible and can be easily extended in the future to new data types, new analysis methods, and other experimental systems.

Index words: Malaria, *Plasmodium*, bioinformatics, data integration, genomics, transcriptomics, proteomics, non-human primates

Data Integration for Systems Biology

by

Elizabeth D. Trippe

B.S., Kennesaw State University, 2004

M.S., University of Illinois Urbana-Champaign, 2006

A Dissertation Submitted to the Graduate Faculty
of The University of Georgia in Partial Fulfillment
of the
Requirements for the Degree

Doctor of Philosophy

Athens, Georgia

2018

Data Integration for Systems Biology

by

Elizabeth D. Trippe

Major Professor:	Juan B. Gutierrez
Committee:	Jonathan Arnold Jessica Kissinger David Peterson

Electronic Version Approved:

Suzanne Barbour
Dean of the Graduate School
The University of Georgia
December 2018

Dedication

This thesis is dedicated to my family.

Acknowledgments

In particular, I would like to acknowledge the contributions of the following people. Sanaz Haghani assisted with Figure 1.1. Jacob B. Aquilar provided the formal definition of data primitives. Mustafa V. Nural implemented the relational schema, improved the architectural design of the solution, and wrote the software necessary to query the relational database. Jeremy D. DeBarry worked to curate and annotate the data. Diego M. Moncada assisted with interpretation of results. Rabindra Tirouvanziam and Mary R. Galinski provided expert knowledge and contributed to interpretation and understanding of results in the context of malaria. Jessica C. Kissinger helped develop the concept of data primitives, and their implementation. Juan B. Gutierrez conceived the concept of data primitives.

This project was funded in part by Federal funds from the US National Institute of Allergy and Infectious Diseases, National Institutes of Health, Department of Health and Human Services under contract #HHSN272201200031C, 2012-2017, PI Mary R. Galinski, which supports the Malaria Host-Pathogen Interaction Center (MaHPIC). This work was also supported by the Defense Advanced Research Projects Agency and the US Army Research Office through the program Technologies for Host Resilience - Host Acute Models of Malaria to study Experimental Resilience (THoR's HAMMER), DARPA contract #W911NF-16-C-0008, 2016-2019, PI Mary R. Galinski. All experiments were performed at the Yerkes National Primate Research Center and are supported in part by the National Institutes of Health, Office of the Director (P51OD011132). The content of this work is solely the responsibility of the authors and does not necessarily represent the official views of the National Institutes of Health or other funding organizations.

I would like to thank all the members of the MaHPIC-HAMMER consortium for their help and advice.

Table of Contents

Page

List of Figures

List of Tables

Chapter 1

Literature Review

1.1 Introduction

The bottleneck in the biological discovery pipeline is now the generation of meaningful results and models from a complex dataset rather than its production and dissemination. The cost of a generating a genome is now less than the cost to store it and the results of analysis may be more than can reasonably be tested or confirmed.

The difficulties of repurposing even the simplest quantitative modeling tools (e.g. linear regression) from one dataset to another, and the difficulties in integrating just two levels of biologically complexity (e.g. transcriptome and metabolome), invite one to look for similarities and commonalities in data structures that could be exploited for re-usability of analysis methods between biological data types.

In order to address the problems associated with management of data analysis, the Scientific Knowledge Extraction from Data (SKED) framework was developed as a part of this work. Quantitative data are harmonized into data primitives, which are JavaScript Object Notation (JSON) formatted files that contain the metadata and data associated with a data primitive. The formats of each data primitive are described in detail in Chapter 3 and the Appendix. The SKED object-oriented schema provides a means to easily re-use quantitative pipelines and functions for different data types.

1.2 Systems Biology Needs New Data Management and Integration Approaches

Biological sciences are undergoing a rapid increase in the amount of data gathered during an investigation; not only have experiments expanded from studying and measuring one part (e.g. a gene) of an organism to studying all structurally and functionally similar parts (e.g. a genome) but we have also expanded the data types and levels of organization being studied (e.g. epigenome, immunome, etc.). Raw output from next-generation sequencing (NGS) for research has surpassed Moore's Law of performance improvements in information storage and computation, so that even the simplest, most basic experimental designs now produce larger datasets than ever before [1]. The further expansion of these -omic technologies from research to personal use and clinical practice(i.e. 23-and-me) has compounded the problem, as has the increased availability and use of real-time biomedical and consumer health monitoring devices [2, 3]. This expansion in technology has enabled research to shift from reductionist approaches which focus on finding and determining the roles of components, to systems approaches which focus on the dynamic interactions of the system's constituents [4].

Systems approaches transcend the traditional boundaries across disciplines and frequently rely on informatics approaches that involve complex datasets describing a system at multiple spatial and temporal scales [5]. Systems chemistry, which aims to holistically understand complex chemical systems and predict the outcomes of novel chemical systems relies on intensive calculations for structural analysis [6]. Systems pharmacology, which aims to holistically understand drug mechanisms to improve drug efficacy and clinical outcomes, relies on data sets from multiple scales which may include chemical, cellular, physiological and environmental measurements [7]. Systems biology is the focus in this work but the approach should be applicable to other types of systems. Systems biology studies may incorporate many levels of biological organization including population and ecosystem information while systems medicine focuses on the improvement of human health and treatment of disease [8–10]. Such systems approaches have resulted in advances in drug resistance reduction, cancer therapy and cardiovascular disease. However powerful these approaches might be, they frequently use large, heterogeneous, complicated and often sparse data and models to reach conclusions.

Over time, many public data resources have been developed that make biological databases, standards, and analysis tools easily available and more easily used by the wider scientific community, yet the re-organization and homogenization of data for different projects is still a monumental task requiring specialized expertise. For example, the developers of the Findable, Accessible, Interoperable, and Reusable (FAIR) data sharing principles have approximately 950 sets of standards and approximately 940 databases documented and listed on their website (FAIRsharing.org) [11]. Also, the goals of the COmputational MOdeling in Biology and NEtwork (COMBINE) initiative include the development of standards for the new fields of systems and synthetic biology, and the improvement of the interoperability of current standards and tools [12]. These initiatives and others like the BD2K initiative, have made great contributions and are very useful sources of raw data [13].

1.2.1 Motivation for Data Primitives

Philosophers working on epistemological and scientific problems have used mathematical notations to depict their ideas and model the natural world. Notable among these have been Immanuel Kant, according to whom reality consists of knowledge of processes in time and space [14]. Believing that these notions exist as real entities outside of human perception, Leibniz and Newton argued whether time and space were relative or absolute quantities [15]. Later, in a move from studying physical systems to biological systems, Robert Rosen and subsequently A. H. Louie, used set theory to describe the complexity of a living system [16–18].

In contrast to historic models depending on imprecise and sparse data sets, modern models rely on precise and dense measurements of complex systems. Therefore, categories of data structures are needed in order to reduce this complexity. According to the Kantian view, reducing such data to a minimalistic representation would result in data structures that represent time, space and associated information. According to Newton and Leibniz, the absolute or relative notions of these measurements would be required for a reduced data framework. Anticipatory systems, as described by Rosen, would require data representations that encompass all possible measures of physical and biological entities with ways of encoding states or outcomes.

Efforts to generalize data types and formats for quantitative analysis have resulted in minimalistic data structures called in this work "data primitives". These structures provide the basis of the SKED framework to organize, combine, re-purpose and analyze large and small datasets. Quantitative scientists now have basic structures to easily utilize different types of data from different sources at different scales. Their usefulness, formats and implementation are shown here.

1.3 Bioinformatics and Programming

1.3.1 Functional and Object-Oriented Programming in Bioinformatics

Functional programming connects inputs and outputs of a series of code expressions to create a program. In contrast, object-oriented programming (OOP) uses data structures, associated methods and interactions between objects to create a program. Functional programming is most useful when the formats of input and output data structures are rigidly defined and cannot be easily changed, while OOP is most useful when there is a variety of input data formats. In many cases the data format does not even need to be known in advance [19].

Today, most bioinformatics programming uses functional programming pipelines to analyze data. Genome and transcriptome assembly are examples of this. A limited number of raw sequence file formats (ex. FASTQ, BCL) are assembled into a small number of aligned sequence file formats (ex. BAM/SAM). This creates complications when working on integrative analyses, where information from more than one -omics technology is being used. Each data type has its own particular set of tools and functions that may not be compatible with the set of tools used to analyze a different data type.

OOP uses objects to encapsulate the data properties and methods associated with the object. Classes define types or kinds of objects. There are many advantages to using this type of programming. First, real-world and abstract objects are easily modeled by object classes in OOP. Second, classes reduce complexity and hide implementation details. This can significantly reduce the complexity of a program [20]. Other advantages include scalability, compatibility, re-usability, extensibility and platform independence[21].

1.3.2 Good Programming Practices: The Importance of Testing

To ensure that this project and the code written for this work is re-useable and can be built upon in the future, good programming practices were followed as outlined by Martin [21]. These include well-commented code along with unit-testing for consistency and accuracy of program features. The unit tests were designed using the built-in testing framework in MATLAB 2018b. With unit tests to ensure the accuracy of individual code blocks, integration tests were also performed to ensure that functions and methods produced consistent outputs. These two levels of testing allow us to easily confirm the reproducibility and proper operation of the SKED classes.

1.3.3 The Problem of Reproducibility

Reproducibility of computational research has been identified as one challenge for systems biology [22, 23]. When reproducing computational results, “forensic bioinformatics”, where a scientist must check the input and output data to determine the methods that have been used, must often be used when documentation and directions did not provide enough information [24]. One case study describes a novice user needing about 280 hours to reproduce a method [25]. With the fast pace of research and the need to make the most of valuable high-throughput experimental results, computational findings need to be reliable and easy to use [26].

Solutions have been proposed including making scientific articles “preproducible”, so that there is enough detail about the experiment or analysis for someone else to try it themselves [27]. There is a “Manifesto for Reproducible Science” [28] and “Ten Simple Rules for Reproducible Computational Research" [23] along with "Ten Simple Rules for Reducing Overoptimistic Reporting in Methodological Computational Research" [29]. The ReScience Initiative encourages the replication of already published results and all data and code must be submitted to their github repository before publication [30]. Yet none of these seems to have made a significant impact on the way computational analyses are performed and reported.

1.3.4 The Three V's of Big Data Analysis in Bioinformatics

The three V's of Big Data were first coined by Doug Laney in 2001 as Volume, Variety, and Velocity [31]. Bioinformatics databases began with genomic information and the amount of this information has expanded as different species have complete genomes assembled. The Volume of biological data continues to increase. Next, as technology develops, new biological data types will be measured and the Variety of biological data types will continue to increase. Last, the number of measurements taken throughout an experiment will continue to increase, and data growth will continue as more time points, Velocity, are recorded.

New analysis approaches are needed to enable researchers to deal with these 3V's . The SKED framework was specifically designed for this purpose and has been successfully used in the implementation shown here.

1.3.5 Data Integration is Required to make full use of multi-omic data sets

When different -omic measurements are taken over the course of an experiment, a seemingly easy, but surprisingly difficult, logical first step in analysis is to combine evidence, like transcriptomic and metabolomic data, to look for novel molecular relationships. However the information measured for each type of molecule is different, with processed transcriptomic data consisting of gene identity and associated expression level, while processed untargeted metabolomic data from mass spectrometry consists of mass-to-charge (m/z) ratio, retention time (RT), abundance level as well as putative molecular identity. The computational scientist analyzing such data must then come up with a way to combine the data in a meaningful way that is not biased toward the importance of either data type [32] and does not lose the information associated with each unique measurement.

An examination of the commonalities of several data types showed that they were all quantitative measures that changed throughout the experiment, leading to the classification of “time series”. This classification was general enough to accommodate data taken at different time scales (daily vs monthly) and different biological levels (cellular vs molecular vs clinical). We were then able to create and use a method called Massively Parallel Analysis of Time Series (MPATS) to look at the thousands of different time series gathered during one infection experiment [33] of the MaHPIC project. This approach was extended to include other basic data types including images and text.

The success of this approach led us to think of other basic data structures for raw, experimental data and to notice that these same data structures were also the most basic for reporting results. For example, annotated graphs are very often used to display the relationships between entities resulting from analysis.

The resulting reduced data structures are independent, atomic units of data representation, and their use has many advantages over current standards of data uniformization in systems biology studies. In addition to facilitating data integration at all levels, data primitives allow the standardization of the data ingestion step which significantly improves the reproducibility of the analysis. Analysis methods can thus be easily repurposed from one data type to another and used with combined data sets.

Rather than focus on the integration of only one or two data types, data primitives allow the integration of multiple data types in a modular, extensible fashion. Data primitives are the foundation of the SKED framework, in which data primitives are used for data storage to allow integration of large, heterogeneous data sets and increase reproducibility and reliability of computational analyses.

1.4 Malaria Is Still A World-wide Problem

Malaria continues to be a worldwide health burden in spite of research efforts to develop novel disease treatments and intervention strategies [34]. According to the WHO, an estimated 216 million (95% CI: 196-263 million) cases of malaria occurred in 2016, along with 445 000 deaths [34].

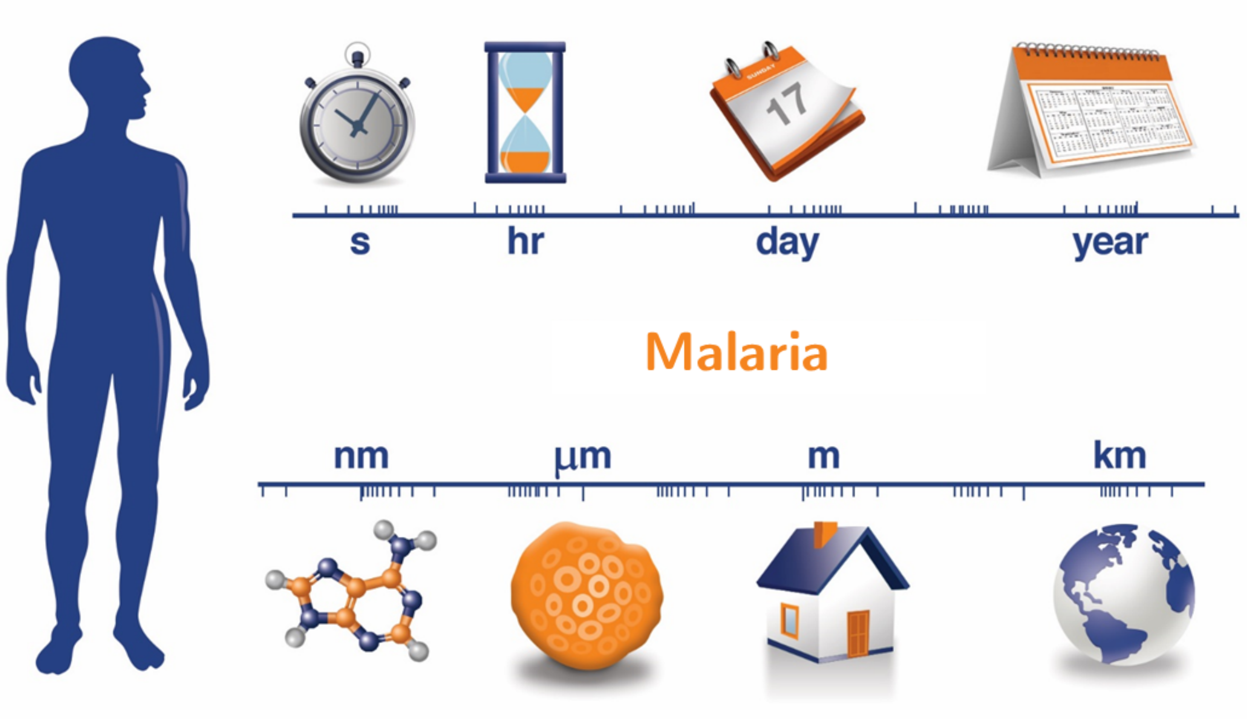


Figure 1.1: Malaria is a problem that crosses multiple time scales and multiple spatial scales.

Clinical manifestations associated with *Plasmodium* infections may be classified into asymptomatic, uncomplicated, and complicated cases with common symptoms that include fever, chills, and muscle aches [35, 36]. Asymptomatic cases most often occur in adults from regions of high malarial transmission and are characterized by the presence of circulating parasites, with parasitemias that may be up to 50,000 parasites per microliter, but no symptoms [35, 37, 38]. Uncomplicated malaria is characterized by parasitemias typically in the range of 1,000 to 50,000 parasites per microliter along with fever, sweating, chills and muscle aches but symptoms may include headache, nausea, vomiting, diarrhea, and anemia [35, 39]. Complicated or severe malaria happens more often in *P. falciparum* infections than with other infecting species and is deadly in 20% of adults and 10% of children [35, 40]. Since malaria symptoms are not always related to the level of parasitemia, other causes and treatments are needed to reduce the burden of the disease.

While in general, it can be seen that higher pathogen loads results in greater symptoms and more complications, this is not always the case as was first shown by Rberg et al. [41]. Even though pathogen load is not clearly correlated to severity of symptoms, previous intervention strategies have focused on mechanisms whereby parasite replication is prevented and the parasite is cleared [42]. This anti-parasitic immunity is in contrast with anti-disease or clinical immunity in which symptoms of the disease are prevented [42]. Anti-disease immunity, which can be quantified by tolerance curves, is associated with asymptomatic malaria cases [42–45]. An example of this is shown in Figure 1.2. In malarial infections, the hosts own immune system, with excessive inflammatory activation, can be responsible for much of the damage done by the disease [45–47].

For host-pathogen systems, exposure to the pathogen and the subsequent ability of the host to maintain health and productivity, termed resilience, is often used when quantification of the parasite load throughout the infection is not of interest or is not feasible, as with herd animals and livestock [48–54]. Tolerance, resilience and anti-disease immunity, with their focus on host processes and prevention of damaging immune responses, have been studied more in recent years [43, 44, 55–57]. This has resulted in the identification of tolerance pathways and mechanisms, including production of anti-inflammatory molecules, induction of anti-oxidant mechanisms and metabolic adaptation by the immune system, that reduce the impact of the disease without attacking the parasite [58–61].

To take advantage of both systems biology approaches and high-throughput technologies, the Malaria Host-Pathogen Interaction Center (MaHPIC) and the Technologies of Host Resilience Host Acute Models of Malaria to Study Emerging Resilience (THoRs HAMMER)

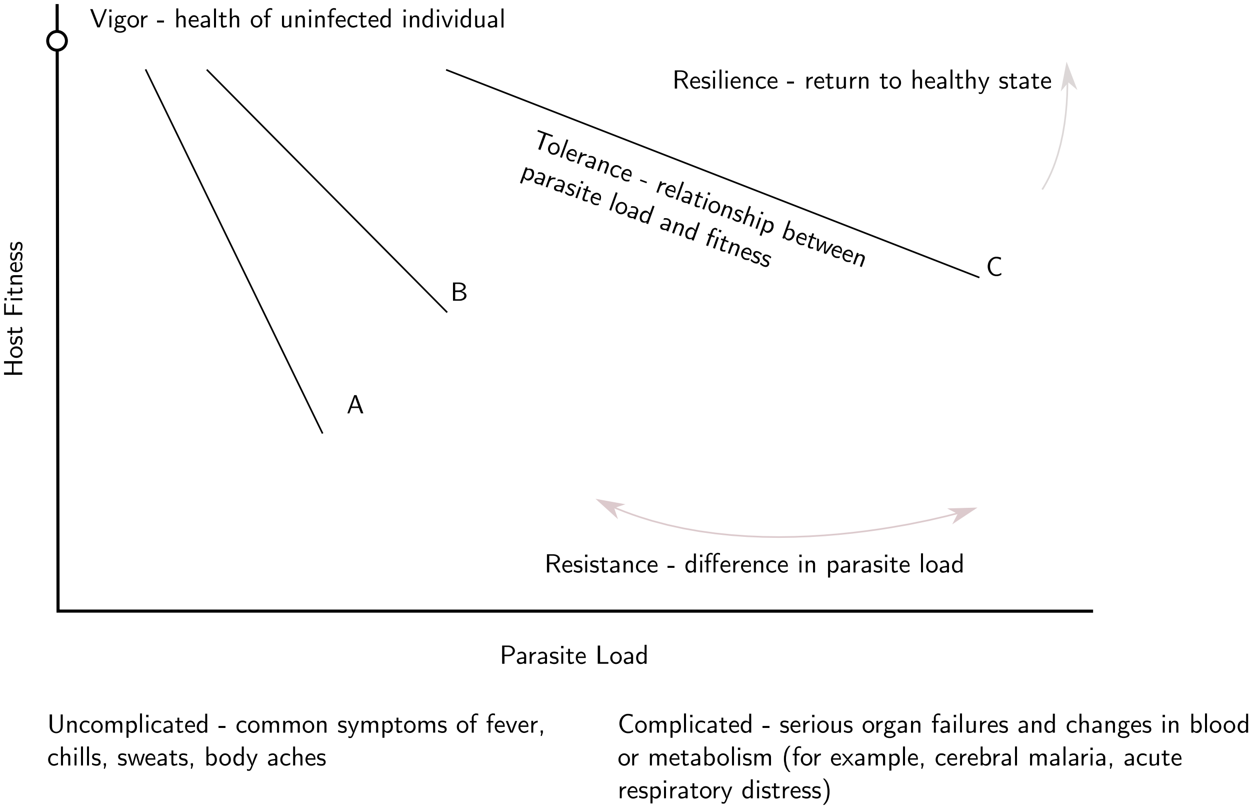

Vigor - health of uninfected individual
Resilience - return to healthy state
Tolerance - relationship between parasite load and fitness
A
B
C
Host Fitness
Resistance - difference in parasite load
Parasite Load
Uncomplicated - common symptoms of fever, chills, sweats, body aches
Complicated - serious organ failures and changes in blood or metabolism (for example, cerebral malaria, acute respiratory distress)

Figure 1.2: Tolerance curve diagram and summary

projects were designed to provide insight into malarial infections [62–64]. Systems biology approaches can provide powerful insight into the multi-level interactions of a biological system, while high-throughput technologies enable many variables to be simultaneously measured. Because the interactions between a host and its pathogens are complex, varied, and hard to study in isolation, such systems approaches will enable a more integrated understanding than studying the systems in isolation. These projects provide a rich dataset from controlled infections that provide a framework to investigate molecular, cellular, and clinical mechanisms of disease which will lead the design of future strategies for interventions in malaria.

Chapter 2

Methodology

2.1 Overview

To address the bottleneck of data analysis in the biological discovery pipeline, the SKED framework was designed. This approach takes advantage of software engineering principles that are used in complex software and data management projects. The design includes the definitions of data primitives to provide common understandings of quantitative data, the exchange of data primitives using JSON formats and encapsulation of properties and methods in an object-oriented scheme. The design improves the reproducibility and reliableness of an analysis while being scalable to larger data sets and new data types. A researcher's efforts can then shift from implementation to interpretation and discovery.

The general overview of the analysis implementation and strategy for data integration and homogenization for systems biology is shown in Figure 2.1. The analysis begins with retrieving all relevant data from SKED Database (SKEDDB) and transforming the files into time series data primitive format. While many data types were gathered over the course of these experiments, the transcriptomic and proteomic data will be the focus here as these data sets contain reliable and rich functional annotations. Information from the different experiments was analyzed using statistical methods and the results from the resilient and non-resilient hosts were compared. Additional comparisons were made to increase the power and consistency of results. These results were then combined with knowledge from databases to identify drug targets and to identify FDA-approved drugs that could be re-purposed as activators or inhibitors to promote a resilient response. The analysis pipeline was also configured to include options for targeted or guided investigation and an example is also included.

The classes, including example input data (from publicly available sources), analysis examples, and test functions, are located at https://gitlab.com/SKED.

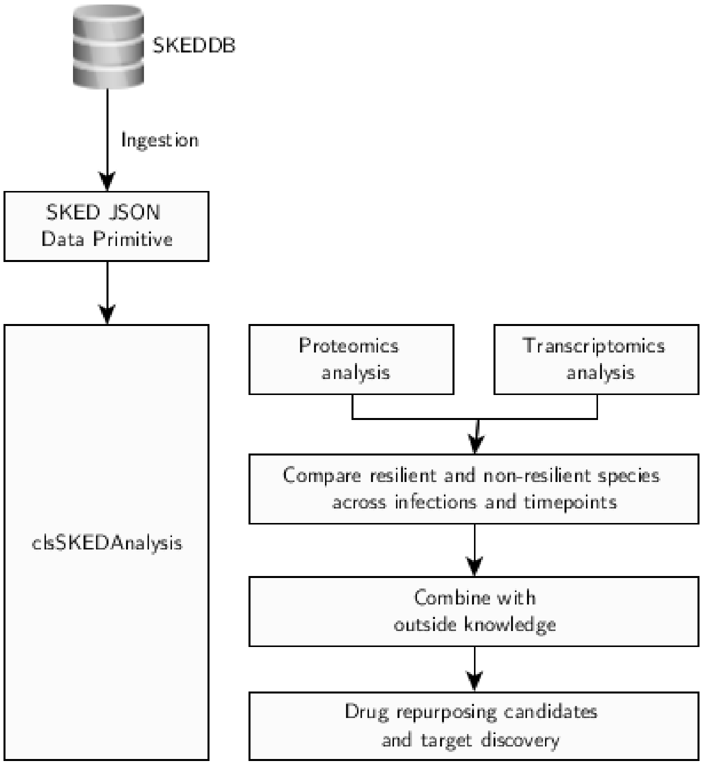

SKEDDB
Ingestion
SKED JSON Data Primitive
clsSKEDAnalysis
Proteomics analysis
Transcriptomics analysis
Compare resilient and non-resilient species across infections and timepoints
Combine with outside knowledge
Drug repurposing candidates and target discovery

Figure 2.1: General analysis overview. SKED classes provide functionality throughout the quantitative analysis steps.

2.2 Definitions of Data Primitives

Listed below is an overview of the notation and mathematical framework utilized throughout this work. Firstly, let $\mathbb{N} := \{0, 1, 2, ...\}$ be the set of natural numbers, so that $\mathbb{N}_+ := \mathbb{N} - \{0\}$ denotes the maximal strictly positive subset of the natural numbers. Additionally, for $n \in \mathbb{N}_+$ let $\mathbb{R}^n$ be the finite dimensional vector space comprised of n-tuples of real numbers. Moreover, the following convention is utilized $\mathbb{R}_+ := \{x \in \mathbb{R} | x \geq 0\}$. Let $\mathbb{C} := \{\bar{z} : \bar{z} = a + bi$ where $(a, b) \in \mathbb{R}^2$ and $i = \sqrt{-1}\}$ be the field of complex numbers, so that for $m \in \mathbb{N}_+$ the symbol $\mathbb{C}^m$ denotes the space of complex valued m-tuples. Furthermore, for $m, n \in \mathbb{N}_+$ denote the set of $m \times n$ real valued matrices by $\mathbb{R}^{m \times n}$. Finally, interval notation is to be interpreted with respect to the underlying ordering (if any) imposed on the elements in the interval.

Definition 1. Time Series Primitive. Let $i, m, n \in \mathbb{N}$, a *time series* consisting of n time points and m variables is a totally ordered set $T := \{(t_i, x_i)\}$, such that $x_i \in \mathbb{C}^m$ for $i \in [0, n]$.

Definition 2. Graph Primitive. A quadruple $G = (V, E, W, S)$ is called a *graph primitive* on V, where:

I Let $V := \{v_l\}$, $l \in [1, m] \subset \mathbb{N}_+$, be defined as a *vertex set.* Each $v_l \in V$ is known as a *vertex.*

II Let the symbol ψ denote a correspondence which assigns each element in V to an unordered p-tuple $\sigma_l \in \mathbb{R}^p$, where $p \in \mathbb{N}_+$. Denote the set $S \subset \mathbb{R}^p$ to be the image of V under the mapping ψ. In this case, each $\sigma_l \in S$ represents a set of numeric values associated with each vertex, representing e.g. gene expression, metabolite intensity, etc. The assumption is made that ψ is a surjection, i.e. for all $\sigma_l \in S$, there exists a vertex $v_l \in V$ such that $\sigma_l = \psi(v_l)$, where $l \in [1, m]$. In set notation it follows that $\psi : V \twoheadrightarrow S$.

III For $i, j, k \in [1, n] \subset \mathbb{N}_+$, the *edge set* $E := \{e_k := (v_i, v_j) \in V \times V\}$ is comprised of n unordered tuples called *edges.*

IV Let the symbol φ denote a correspondence which assigns each element in E to an unordered q-tuple $\omega_k \in \mathbb{R}^q$, where $q \in \mathbb{N}_+$. Denote the set $W \subset \mathbb{R}^q$ to be the image of E under the mapping φ. In this case, each $\omega_k \in W$ represents a set of numeric values associated with each edge, representing e.g. distance, capacity, weight, etc. The assumption is made that φ is a surjection, i.e. for all $\omega_k \in W$, there exists an edge $e_k \in E$ such that $\omega_k = \varphi(e_k)$, where $k \in [1, n]$. In set notation it follows that $\varphi : E \twoheadrightarrow W$.

Definition 3. Polygonal Mesh Primitive. A triple $\mathcal{T} = (V, E, F)$ is called a *polygonal mesh*, provided that the following three conditions are satisfied.

I Let $l \in [1, m] \subset \mathbb{N}_+$, then for all vertices $v_l \in V$, there is an edge $(v_i, v_j) \in E$ such that $v_l = v_i \vee v_l = v_j$.

II For $p \in \mathbb{N}_+$ and $s \in [1, p] \subset \mathbb{N}_+$, define the set $F := \{f_s := (v_i, v_j, \dots, v_k) \in V \times V \times \cdots \times V \ : \ v_i \neq v_j \neq \cdots \neq v_k\}$. The set F is composed of p unordered tuples called *polygons*. Making use of a slight abuse of notation, it is required that for all $(v_i, v_j) \in E$, there exists a polygon $f_s = (v_i, v_j, \dots, v_k) \in F$ such that $(v_i, v_j) \in f_s$.

III Provided two polygons intersect, i.e. $f_r \cap f_s \neq \varnothing$, then the vertex or edge responsible for the nonempty intersection is contained in V or E, respectively, in $\mathcal{T}$.

Definition 4. Image Primitive. Consider the set $\mathbb{N}_+^n := \mathbb{N}_+ \times \cdots \times \mathbb{N}_+$ (n-times). Let $(d_1, \dots, d_n) \in \mathbb{N}_+^n$ and denote the space of real-valued hypermatrices with non-negative entries as $\mathbb{R}_+^{d_1 \times \cdots \times d_n}$. A hypermatrix $H \in \mathbb{R}_+^{d_1 \times \cdots \times d_n}$ can be written as $H = [h_{k_1 \cdots k_n}]_{k_1, \dots, k_n = 1}^{d_1, \dots, d_n}$.

A hypermatrix H is regarded as an image primitive provided that each entry $h_{k_1 \cdots k_n}$ stands for the amount of color k_n in spatial location $h_{k_1 \cdots k_{n-1}}$.

Remark. The common instances are listed below.

I In the case $n = 3$, there are two spatial dimensions representing a location called a pixel and there are d_3 different colors associated with each pixel.

II In the case $n = 4$, there are three spatial dimensions representing a location called a voxel and there are d_4 different colors associated with each voxel.

Note that the terms d_3 and d_4 in the above cases represent the number of distinct colors (or frequencies of the EM spectrum) and other properties (e.g. transparency) under consideration.

Definition 5. Metadata Primitive. Let the sets of *total data*, *meta data* and *experimentally obtained data* in the form of data primitives be labeled by S_T, S_M and $S_D := \{(T_1, G_1, \mathcal{T}_1, H_1), \dots, (T_k, G_k, \mathcal{T}_k, H_k)\}$, respectively. The set S_T admits a unique mutually disjoint decomposition with respect to the analysis conducted, i.e. $S_T = S_M \cup S_D$ where $S_M \cap S_D = \varnothing$. Elements of S_M can be thought of as data that provides information about experimentally obtained data, e.g. the instruments used, the instrument operators, dates, etc. For $|S_M| = n$ and $1 \leq i \leq k$, let $\Phi := \{\phi_1, \dots, \phi_n\}$ be a family of mappings such that for each $s_{m_i} \in S_M$ and $S_{D_i} := \{(T_1, G_1, \mathcal{T}_1), \dots, (T_i, G_i, \mathcal{T}_i)\} \subset S_D$, we have that $\phi_i(S_{D_i}) = s_{m_i}$. In this case, all of the sets under consideration are countable and finite, as a result

$$S_M = \bigcup_{i=1}^{n} \{\phi_i(S_{D_i})\}.$$

Remark. A triangulated mesh is a particular case of more general structures called simplices. A k-simplex is a k-dimensional geometric object with flat sides which is the convex hull of its $k + 1$ vertices. The mesh stores the vertex, edge and face information of a given surface or data set and is a piecewise planar surface, i.e. it is planar almost everywhere, except at the edges where the triangles join. In the case where all of the faces are triangles, the mesh is called triangulated. Therefore, a triangulated mesh can be regarded as a collection of triangles in three dimensional space that are connected in a particular way (to form a manifold on the given surface, i.e. each edge is shared by no more than two faces). It is well known that any surface can be estimated by a series of triangles. Each triangle can store additional data at the faces, e.g. colors, with sharp creases stored on edges and continuously varying quantities stored at each vertex. Due to their relatively simple geometric structure, all triangles can be represented as triples. An advantage of using such a mesh lies in the ability to efficiently answer data queries (information requests from a given database), e.g. finding the vertices or edges of a particular face or finding all triangles around a vertex.

2.3 JSON Formats for Data Primitives

Data primitives were designed to be the basic building blocks of quantitative analysis and JSON file format was chosen to be the basic file format for data primitives. JSON is a data-interchange file format that is light-weight and easy to read [65]. The format is not based on a single, individual programming language and automatic parsers have already been written for most programming languages [65, 66].

The JSON format is based on the ability to create complex, nested structures and JSON files are built around two basic data structures, objects and arrays. Objects are unordered collections of name-value pairs, and are surrounded by curly braces ({}) [65]. Arrays are ordered lists of values, and are surrounded by brackets ([ ]). [65] JSON values can be strings(surrounded by double quotes (" ")), numbers, boolean, null, objects or arrays [65]. JSON is commonly used for fast, efficient browser communication in web sites.

To be effective as a basic unit of quantitative analysis, the JSON data primitive format needs to be able to incorporate quantitative data and associated structural, administrative, and descriptive metadata. Structural and administrative metadata provide information about the data object, its origin and composition, while descriptive metadata contains more specific information about parts of the data object [67]. The basic data primitive JSON format is shown in Listing 1. The basic data primitive format contains a metadata section describing the properties and descriptors of the data primitive. This section contains elements like experiment name, subject name, subject species and location of the experiment. The "data" element contains the quantitative values and variables associated with the data primitive

(Listing 1, line 7). This element contains two sections: a header section with metadata and descriptors and another section for the quantitative elements of the data primitive.

```
 1 {
 2         "data_primitive": {
 3                 "type": "Data_primitive_type",
 4                 "metadata": {
 5                 "external_metadata_files":[ ],
 6                         "reference ontologies": [ ],
 7                         "key metadata term 1": "value metadata term 1"
 8                 }
 9                 "data": {
10                         "header": {
11                                 "key metadata term 2": "value metadata term 2"
12                          },
13                          "data primitive elements": {
14                                 "data point x": ". . .",
15                                 "key metadata term 3": "value metadata term 3"
16                         }
17                 }
18         }
19 }
```

Listing 1: Basic JSON data primitive format

The data primitive elements are different for each type of data primitive and these are summarized in Table 2.1. The precise mathematical descriptions of each data primitive may be found in [68]. Each of the data primitive elements in the right column form the basis for the corresponding section in the JSON file. The individual data elements that make up a particular data primitive JSON file are usually JSON arrays or JSON objects, but could be any other valid JSON value.

Table 2.1: Elements of different data primitives

Data Primitive Type		Data Primitive Element
📈	Time Series	T: set of time stamps X: set of values
📷	Image	V: set of vertices S: array of color values
⬠	Polygonal Mesh	V: set of vertices E: set of edges F: set of polygons
☍	Graph	V: set of vertices S: set of vertex values E: set of edges W: set of edge weights
🏷	Metadata	K: set of unique keys E: set of values

Because the time series data primitive is used throughout this implementation. An example of a short time series JSON file may be found in Listing 2, while example formats may be found in Appendix A for the remaining data primitives. The example in Listing 2 is for two variables(hematocrit and hemoglobin (hgb) levels in whole blood) measured at three times during the experiment. These measurements are part of the common clinical Complete Blood Count (CBC) and provide information about overall health [69]. Hematocrit values describe the ratio of the volume of red blood cells to total blood volume and can indicate conditions like anemia or dehydration depending on if they are high or low [69]. In this example, the metadata information about the experiment is easy to read at the top of the file and there is a reference to an external file with more information. The data header information(starting in line 16) contains properties about the variables and time formats in the subsequent time series array information.

```
{
  "data_primitive": {
    "type": "time_series",
    "metadata": {
      "id": "mahpic_E04_ME_CBC_RMe14",
      "external_metadata_files": [
        "E04M99MEMmCyDaWB_07102018-README_MULTIPL.txt"
      ],
      "experiment": "E04",
      "subject": "RMe14",
      "protocol": "ME_CBC",
      "protocol_app_id": "3",
      "summary": "E04 Clinical CBC Panel Results",
      "data_type": "Clinical",
      "protocol_description": "CBC Panel"
    },
    "data": {
      "header": {
        "term": [ "x_hematocrit_", "x_hgb_" ],
        "description": [ "Measurement Name", "Measurement Name" ],
        "unit": [ null, null ],
        "timestamp_format": "YYYY-MM-DD HH:MM:SS.S"
      },
      "time_series": [
        {
          "time_stamp":"2013-09-04 00:00:00.0",
          "value": [ ["43.0000"], ["13.6000"]
          ]
        },
        {
          "time_stamp":"2013-09-06 00:00:00.0",
          "value": [ ["41.8000"], ["13.1000"]
          ]
        }
      ]
    }
  }
}
```

Listing 2: Example of a small data primitive time series

2.4 Object-Oriented Schema Designed for Extensible and Reproducible Analysis

With SKED, the first step of data harmonization occurs using the SKED Ingestion classes, as shown in Figure 2.2. The SKED ingestion classes provide a consistent user interface to convert data from multiple source types into the JSON data primitive formats. The SKED relational database has been implemented and the classes provide a means to retrieve all quantitative data types (functional genomics, proteomics, metabolomics, etc) for results from the MaHPIC-HAMMER projects. This harmonization step ensures that all quantitative data is in the same format with the same structure. The classes have been implemented using MATLAB 2018b [70] and are available online at https://gitlab.com/SKED.

To provide consistent functionality, object-oriented analysis classes were created to access and use data primitives. These classes were implemented using MATLAB 2018b and are shown in Figure 2.3. For reading and parsing the JSON files, the `org.java` package from the `JSON-java` project are used [71]. The code for each class may be found in Appendix C Using these Java classes enables reuse of previous code and provides a mechanism to make the implementation of data primitives interoperable across programming languages and tools.

2.5 MaHPIC-Hammer Data Summary

A summary of the experiments described is found in Table 2.2. This table summarizes the host species and infecting malaria species. The table also includes information about infection type (primary or secondary) and the number of subjects. Note that the experiments involving resilient subjects are on the right side of the table. The same tissue sample types were not always collected throughout each experiment for each type of time point. Results are described only for comparable samples.

The MaHPIC datasets may be referenced with BioProject Accession number PRJNA385820 and are part of superseries GSE94274: An Integrated Approach to Understanding Host-

Pathogen Interactions. Most of these data sets are publicly available at NCBI as part of the Gene Expression Omnibus (GEO). The platforms and procedures used for sample collection, sample processing and library preparation and may be found with platform reference numbers GPL14954, GPL25689, GPL25691, GPL25692 and GPL25694. Sequencing was performed using Illumina HiSeq 1000, 2000 or 3000 with the appropriate host and parasite genomes for each experiment. More information about the experiments, including the clinical information and summary diagrams, may be found at `http://plasmodb.org/plasmo/mahpic.jsp` [72].

In addition to the blood and bone marrow transcriptomic data, targeted proteomics analysis using plasma samples was also conducted. This was done using the SomaLogic platform. The test uses SOMAmer (Slow Off-rate Modified Aptamer) reagents, which are single-stranded DNA sequences that bind specifically to certain target proteins [73, 74].

2.6 Differential Expression

The differential expression analysis was conducted using the SKED object-oriented schema with the Bioinformatics Toolbox in MATLAB 2018a [70]. The raw gene counts from whole blood RNA-Seq analysis for each subject (*M. mulatta* or *M. fascicularis*) were first library-size normalized and a negative binomial distribution was used to infer differential expression [18] [75]. The Benjamini-Hochberg adjustment was used to correct for multiple testing problems with a false discovery rate of 10% [76]. Genes were considered differentially expressed if the adjusted p-value was less than 0.05 and the fold change was more than two-fold. The differentially up and down regulated genes were then compared across host species and parasite species to determine unique and common genes to each group.

The targeted proteomic analysis resulted in measures of median hybridized samples across plates which were downloaded from SKEDDB. A two-sample *t*-test was used to compare protein abundances between baseline and infected samples [77].

2.7 Reference Databases

To take advantage of previously accumulated knowledge, reference databases were used to classify targets and to identify already FDA-approved modulators (activators and inhibitors) of these targets. Publicly available databases like GenBank, for nucleotide sequences, and UniProt, for protein sequences, make information more accessible and search-able as well as easier to update so that the most current information is more available [78, 79]. The two databases references in this analysis were the Pharos database [80–82] and the Drug-Genome Interaction Database(DGIdb 3.0) [83–85]. The Pharos database categorizes targets into four categories based on types of knowledge available about the target. These are called Illuminating the Druggable Genome (IDG) targets and they are described in detail here [82]. The databases were accessed through application program interfaces (API) to retrieve information about the targets. The resulting data was visualized as graphs to summarize the retrieved information.

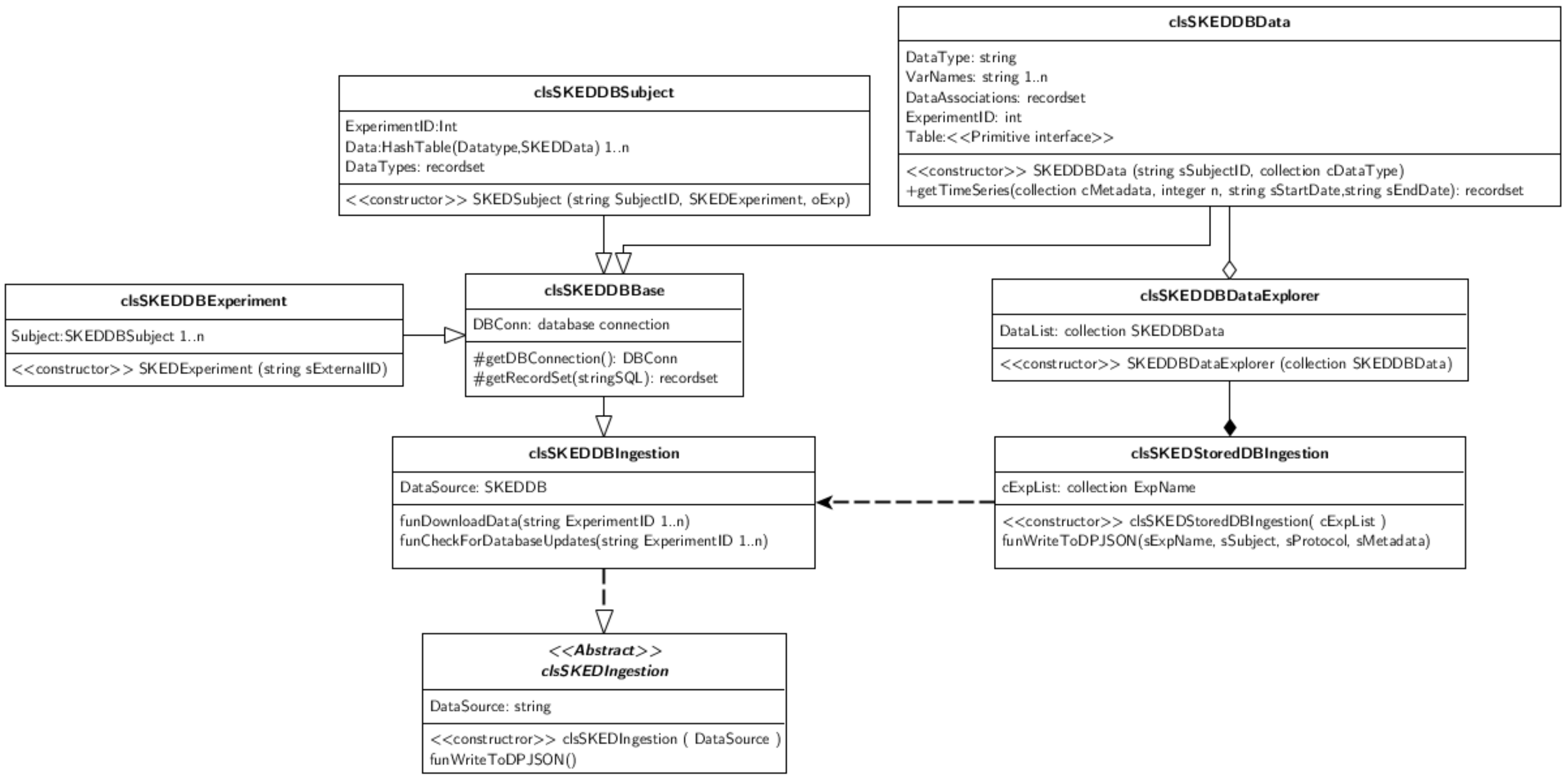

clsSKEDDBSubject
ExperimentID:Int
Data:HashTable(Datatype,SKEDData) 1..n
DataTypes: recordset
<<constructor>> SKEDSubject (string SubjectID, SKEDExperiment, oExp)
clsSKEDDBData
DataType: string
VarNames: string 1..n
DataAssociations: recordset
ExperimentID: int
Table:<<Primitive interface>>
<<constructor>> SKEDDBData (string sSubjectID, collection cDataType)
+getTimeSeries(collection cMetadata, integer n, string sStartDate,string sEndDate): recordset
clsSKEDDBExperiment
Subject:SKEDDBSubject 1..n
<<constructor>> SKEDExperiment (string sExternalID)
clsSKEDDBBase
DBConn: database connection
#getDBConnection(): DBConn
#getRecordSet(stringSQL): recordset
clsSKEDDBDataExplorer
DataList: collection SKEDDBData
<<constructor>> SKEDDBDataExplorer (collection SKEDDBData)
clsSKEDDBIngestion
DataSource: SKEDDB
funDownloadData(string ExperimentID 1..n)
funCheckForDatabaseUpdates(string ExperimentID 1..n)
clsSKEDStoredDBIngestion
cExpList: collection ExpName
<<constructor>> clsSKEDStoredDBIngestion(cExpList)
funWriteToDPJSON(sExpName, sSubject, sProtocol, sMetadata)
<<Abstract>>
clsSKEDIngestion
DataSource: string
<<constructror>> clsSKEDIngestion (DataSource)
funWriteToDPJSON()

Figure 2.2: SKED Ingestion Class diagram

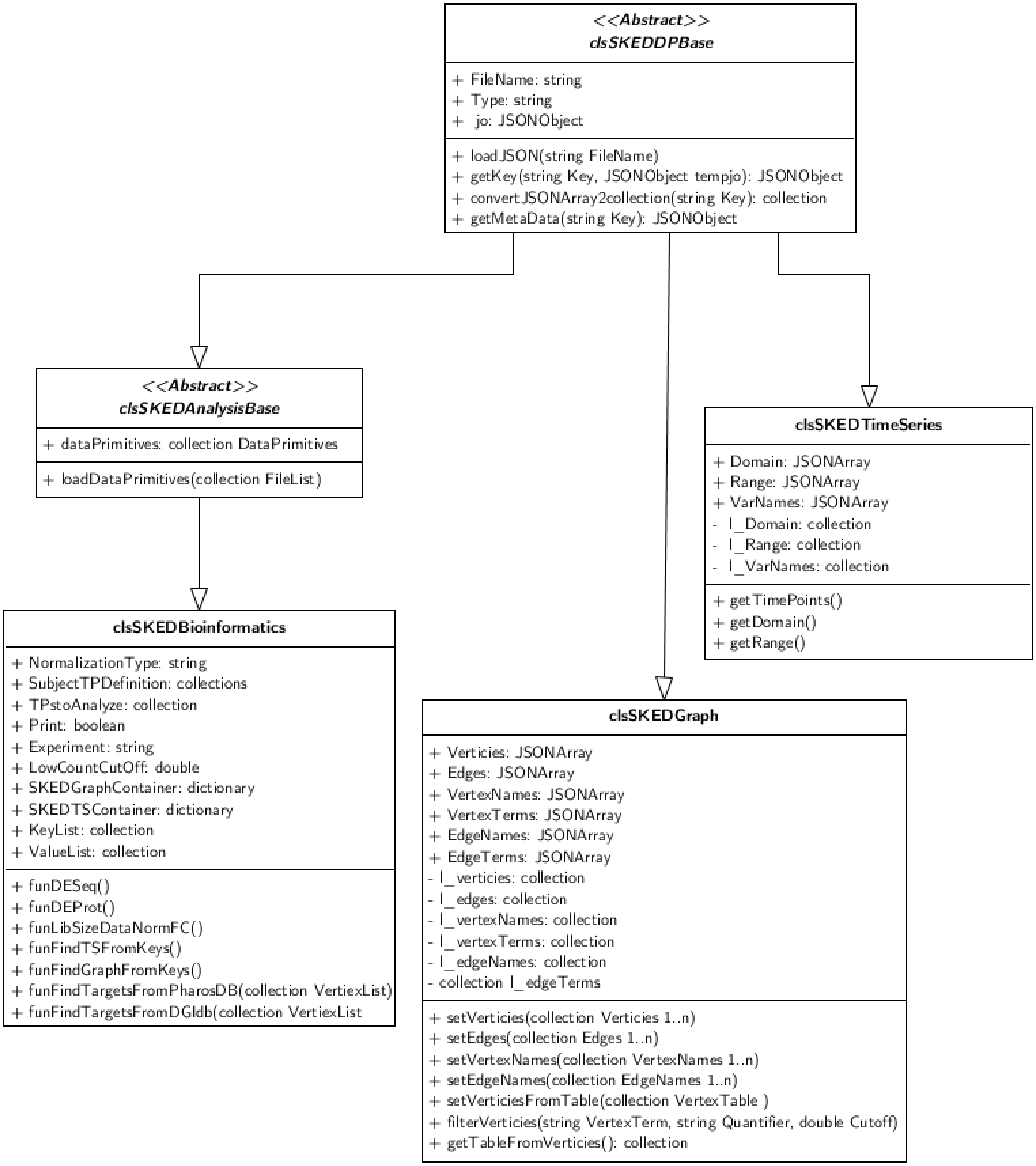

<<Abstract>>
clsSKEDDPBase
+ FileName: string
+ Type: string
+ jo: JSONObject
+ loadJSON(string FileName)
+ getKey(string Key, JSONObject tempjo): JSONObject
+ convertJSONArray2collection(string Key): collection
+ getMetaData(string Key): JSONObject
<<Abstract>>
clsSKEDAnalysisBase
+ dataPrimitives: collection DataPrimitives
+ loadDataPrimitives(collection FileList)
clsSKEDTimeSeries
+ Domain: JSONArray
+ Range: JSONArray
+ VarNames: JSONArray
- l_Domain: collection
- l_Range: collection
- l_VarNames: collection
+ getTimePoints()
+ getDomain()
+ getRange()
clsSKEDBioinformatics
+ NormalizationType: string
+ SubjectTPDefinition: collections
+ TPstoAnalyze: collection
+ Print: boolean
+ Experiment: string
+ LowCountCutOff: double
+ SKEDGraphContainer: dictionary
+ SKEDTSContainer: dictionary
+ KeyList: collection
+ ValueList: collection
+ funDESeq()
+ funDEProt()
+ funLibSizeDataNormFC()
+ funFindTSFromKeys()
+ funFindGraphFromKeys()
+ funFindTargetsFromPharosDB(collection VertiexList)
+ funFindTargetsFromDGIdb(collection VertiexList
clsSKEDGraph
+ Verticies: JSONArray
+ Edges: JSONArray
+ VertexNames: JSONArray
+ VertexTerms: JSONArray
+ EdgeNames: JSONArray
+ EdgeTerms: JSONArray
- l_verticies: collection
- l_edges: collection
- l_vertexNames: collection
- l_vertexTerms: collection
- l_edgeNames: collection
- collection l_edgeTerms
+ setVerticies(collection Verticies 1..n)
+ setEdges(collection Edges 1..n)
+ setVertexNames(collection VertexNames 1..n)
+ setEdgeNames(collection EdgeNames 1..n)
+ setVerticiesFromTable(collection VertexTable)
+ filterVerticies(string VertexTerm, string Quantifier, double Cutoff)
+ getTableFromVerticies(): collection

Figure 2.3: SKED Analysis Class diagram.

Table 2.2: Summary of MaHPIC-HAMMER experiments

E03	E04	E23	E24	E25	E30	E06	E07	E15
MaHPIC					**MaHPIC-HAMMER**			**MaHPIC**
P. coatneyi	*P. cynomolgi*				*P. knowelsi*			*P. vivax*
Primary infection			Secondary infection		Primary infection			
Macaca mulatta							*Macaca fascicularis*	*Aotus nancymaae*
Non-resilient							Resilient	
Hackeri strain	B/M strain			Ceylonensis strain	Malayan strain			Brazil VII strain
5 subjects	4 subjects	5 subjects			2 subjects	4 subjects	7 subjects	7 subjects

This information is summarized from the clinical documentation and from `http://plasmodb.org/plasmo/mahpic.jsp` [72].

Chapter 3

Results

3.1 Introduction

The SKED framework is flexible enough to enable both hypothesis-generating analysis and hypothesis-driven research. First, differential expression of genes between bone marrow and whole blood shows different cellular processes associated with the *Plasmodium* infection in resilient and non-resilient hosts. These changes are also reflected in changes to the plasma proteome.

With a goal of discovering novel drug targets and repurposing existing FDA approved drugs, the stages of the *Plasmodium* infection were divided into three target product profiles: early infection (liver stage), rising/peak parasitemia and chronic phase. Uniquely differentially expressed genes during these stages represent targets for these product profiles. Combining this information with target and drug-information from the Pharos database and DGidb identifies imatinib mesylate (trade name Gleevec) as a strong candidate for further testing to promote a resilient host response. For an example of hypothesis driven research, the expression of REVERBα, a key controller of circadian rhythm pathways will be examined.

3.2 Analysis Across Hosts, Tissues, and Data Types

3.2.1 Differentially expressed genes

During the controlled infection experiments, whole blood samples were taken from each subject. Bone marrow samples were taken during the same time points as whole blood but only for E03, E04, E30, E06, and E07. The numbers of differentially expressed genes for each time point category are found in Table 3.1 for whole blood, and in Table 3.2 for bone marrow samples.

Of the samples taken from whole blood, there can be seen to be a great deal of variation in the number of differentially expressed genes at each time point. The small sample sizes (n = 2 for E30, for example) are most likely a major contributor to this. The only time point that is directly comparable across all infections is acute parasitemia.

3.2.2 Plasma Proteomics analysis

Using the SKED classes, the analysis of plasma proteomics was conducted using the same Bioinformatics analysis class. The results of the top five up and down regulated proteins are summarized in Tables 3.3 and 3.4, respectively. The cellular processes that are associated with blood stage malaria infections (e.g. RBC lysis, anemia) are reflected in the proteins whose quantities changed the most throughout the infection. Noteably, this includes hemoglobin for both the resilient and non-resilient species and haptoglobin, a marker for cell lysis.

Also of special interest, is Platelet-derived growth factor (PDGF) BB (PDGF-BB), which is found as one of the most down regulated proteins in the non-resilient host. This family of growth factors and their receptors are seen through out this analysis including in blood and bone marrow transcriptomics.

3.3 Target and Therapeutic Intervention Identification

Target Product Profiles (TPPs) are used to plan drug and target intervention development through out the experimental and drug approval process [86]. They have been written for many infectious diseases including malaria [87]. Here the various *Plasmodium* infection stages across the experiments have been combined to investigate relevant molecular interventions. The target profile infection segments are: early infection, rising and peak parasitemia, and chronic infection.

Table 3.1: Differentially regulated host genes in whole blood. Summary table of up and down regulated genes (p-value < 0.05, fold change > 2 or fold change < 0.5) from the different time point categories during each experiment relative to baseline. For E07, differential expression for infection time points was found compared to baseline 2. For E06 and E30, differential expression for infection time points was found compared to the post-telemetry time point.

	Post-Telemetry	**Baseline2**	**Early Infection**	**Log Phase**	**Acute Parasitemia**	**Post Peak**	**Post-treatment**	**Chronic**
Up								
E03					1116	1003	1157	959
E04					1248	1003		
E23				620	1806	952		
E30	3			508	970			
E06	62		78	939	758			
E07	350	67	200	1157	1133			1736
E15				335	662			778
Down								
E03					142		189	351
E04					132	154		
E23				371	465	225		
E30	27			1617	2238			
E06	33		32	381	481			
E07	120	157	67	883	637			1155
E15				373	639			912

Table 3.2: Differentially regulated host genes in bone marrow. Summary table of up and down regulated genes (p-value < 0.05, fold change > 2 or fold change < 0.5) from the different time point categories during each experiment relative to baseline. For E07, differential expression for infection time points was found compared to baseline 2. For E06 and E30, differential expression for infection time points was found compared to the post-telemetry time point.

	Post-Telemetry	Baseline2	Early Infection	Log Phase	Acute Parasitemia	Post Peak	Post-treatment	Chronic
Up								
E03					555		1301	871
E04					402	199		
E30	9			287	430			
E06	89		461	1677	959			
E07	82	70	188	527	363	662		1149
Down								
E03					555		772	676
E04					1248	1585		
E30	193			1110	1585			
E06	466		40	461	409			
E07	186	97	56	468	442	638		1750

Table 3.3: Up regulated differentially expressed proteins in plasma. Summary table of top five up regulated proteins (fold change > 2) from the different time point categories during E06 and E07 relative to baseline. For E07, differential expression for infection time points was found compared to baseline 2. For E06 and E30, differential expression for infection time points was found compared to the post-telemetry time point.

	Post-Telemetry	Baseline2	Early Infection	Log Phase	Acute Parasitemia	Post Peak	Chronic
Up							
E06	TroponinI		Haptoglobin MixedType	Haptoglobin MixedType	CATE		
	PKC_G		IgE	I_TAC	Haptoglobin MixedType		
	CyclinB1		TXD12	IL_1Ra	EPI		
	Hemoglobin		PTH	IP_10	I_TAC		
	PDGF_BB		IL_22	Transferrin	IL_1Ra		
E07		Hemoglobin	Phospho-glycerate Mutase1	I_TAC	ActivinA	Transferrin	Phospho-glycerate Mutase1
		C3a	CLC4K	ActivinA	LEAP_1	SNP25	IL_17D
		SNP25	Carbonic Anhydrase III	LEAP_1	IL_1R4	I_TAC	Epo
		Thrombin	CD38	IL_1R4	RAN	a1_Anti- chy-motrypsin	C2
		iC3b	phosphoglycera Kinase1	Transferrin	Carbonic Anhydrase III	LEAP_1	alpha_1_anti chy-motrypsin Complex

Table 3.4: Differentially expressed proteins in plasma. Summary table of top five down regulated proteins (fold change < 0.5) from the different time point categories during E06 and E07 relative to baseline. For E07, differential expression for infection time points was found compared to baseline 2. For E06, differential expression for infection time points was found compared to the post-telemetry time point.

	Post-Telemetry	**Baseline2**	**Early Infection**	**Log Phase**	**Acute Parasitemia**	**Post Peak**	**Chronic**
Down							
E06	CK_MM		Hemoglobin	Hemoglobin	Hemoglobin		
	Chk2		TroponinI	RAN	RANTES		
	CK_MM		PKC_G	SP_D	PDGF_BB		
	Transferrin		HistoneH1_2	PDGF_BB	IGFBP_1		
	TroponinI skeletal fastTwitch		SP_D	RANTES	CTAP_III		
E07		Phospho-glycerate Mutase1	Hemoglobin	Thrombin	Thrombin	PYY	Haptoglobin MixedType
		Transferrin	RAN	Hemoglobin	PDE4D	FTCD	C3a
		FTCD	PRKACA	PFD5	AN32B	M2_PK	IL_8
		ADAM9	C4b	PRKACA	CNTN2	PSA	PF_4
		GFRa_1	iC3b	CHIP	PSA	TLR4	SP_D

3.3.1 Early Infection Targets

In order to investigate infection targets specific to early infection or the liver stage, the differentially expressed genes specific to this time point were found as summarized in Figure 3.1. During this stage of the infection, the *Plasmodium* parasite has formed hypnozoites in liver tissue and there is not expected to be a significant transcriptomic response.

Among molecules classified in the clinical target category (T_{clin}), c-KIT, a tyrosine-protein kinase that is also a proto-oncogene is found (see Fig 3.1c). This target is inhibited by imatinib as highlighted in the insert in Fig 3.1d.

3.3.2 Targets from Rising and Peak Infection

To find targets and modulators of the different resilient and non-resilient responses, significantly differentially up regulated genes from log phase and peak parasitemia from the different hosts were combined before being compared, as summarized in Figure 3.2a. Only the targets in the clinical classification are shown since there were 1843 and 588 genes in each category being compared. The *Plasmodium* species infecting the different hosts are described in Table 2.2.

Of note in the clinical target category for targets from the non-resilient host is PDGFRA, Platelet-derived growth factor receptor alpha. PDGFRA is also inhibited by imatinib as was also found in the early infection target list.

3.3.3 Targets from Chronic Stage of Infection

The chronic stage of a *Plamodium* infection represents a time during the infection when the host is controlling the parasite burden but not has not eliminated the parasite. Chronic stage infections are characteristic of resilient but not necessarily resistant host responses. The chronic stage for the non-resilient host, was induced by the administration of sub-curative doses of Artemether, which is active against the blood-stage of the parasite. The resilient host species did not not require treatment to reach or to maintain the chronic phase. Unlike in the other stages, no targets for imatinib were found in any host during chronic phase.

3.3.4 Differentially Expressed Genes From Bone Marrow also Support Imatinib as an Intervention to Promote a Resilient Host Response

In addition to the targets, c-KIT and PDGFRA, found in whole blood and plasma, bone marrow also contains other targets inhibited by imatinib. Imatinib also inhibits PDGFRB, which was only found found to be significantly down regulated in the post-peak time point of E07 (*M. fascicularis* infected with *P. knowlesi*). Unlike in whole blood, the ligands PDGFA and PDGFB are found to be significantly upregulated at many of the same time points as the PDGF repectors. This is seen in E06 (*M. mulatta* infected with *P. knowlesi*) during log phase and peak parasitemia.

3.4 Targeted Analysis: Circadian Clock(REV-ERB pathway)

As an example of hypothesis driven investigation, the SIRT family of proteins was investigated. This family of proteins is known to control circadian rhythms and the REV-ERB pathway. The results from a Kruskal-Wallis test of REV-ERB (which is localized to the nucleus of cells) are shown for all experiments in Figure 3.4. The general trend for this transcription factor is that it's expression level is reduced during infection time points indicating a disruption in biological rhythms. This protein is also found in the plasma proteome with less of a relationship between infection stage and fold change from baseline. No relation was found with changes in REV-ERBα in the bone marrow transcriptomes across infections.

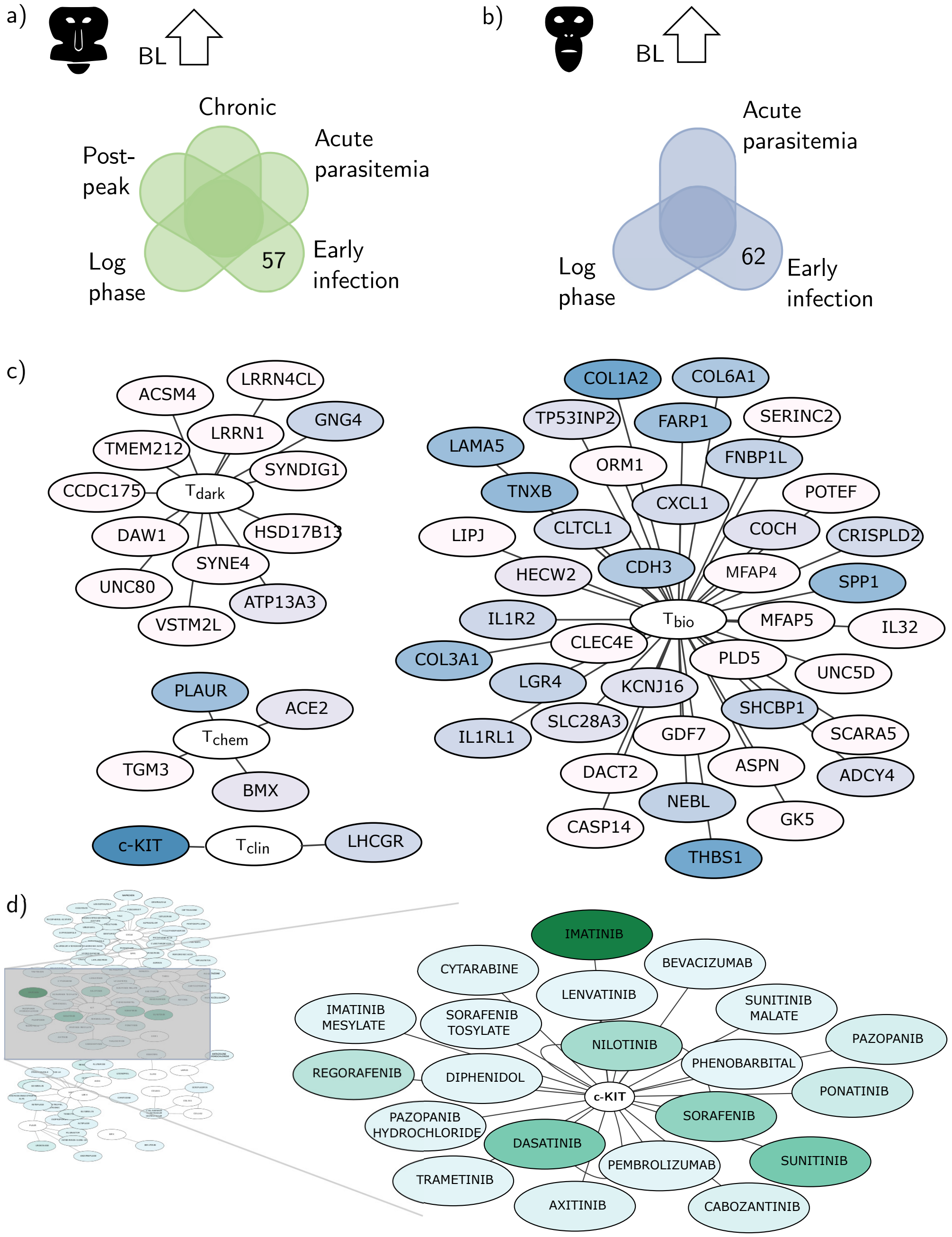


Figure 3.1: Early infection targets. a) Unique up regulated genes during early infection in the resilient host. b) Unique up regulated genes in the non-resilient host. c) IDG target classifications for the 62 genes unique to early infection in the non-resilient *M. mulatta* host. The shading represents the PDB DrugScore. d) FDA-approved drugs that are known to have interactions with the targets from Part c. The shading represents the Knowledge Availability Score, which is a combined literature reference score.

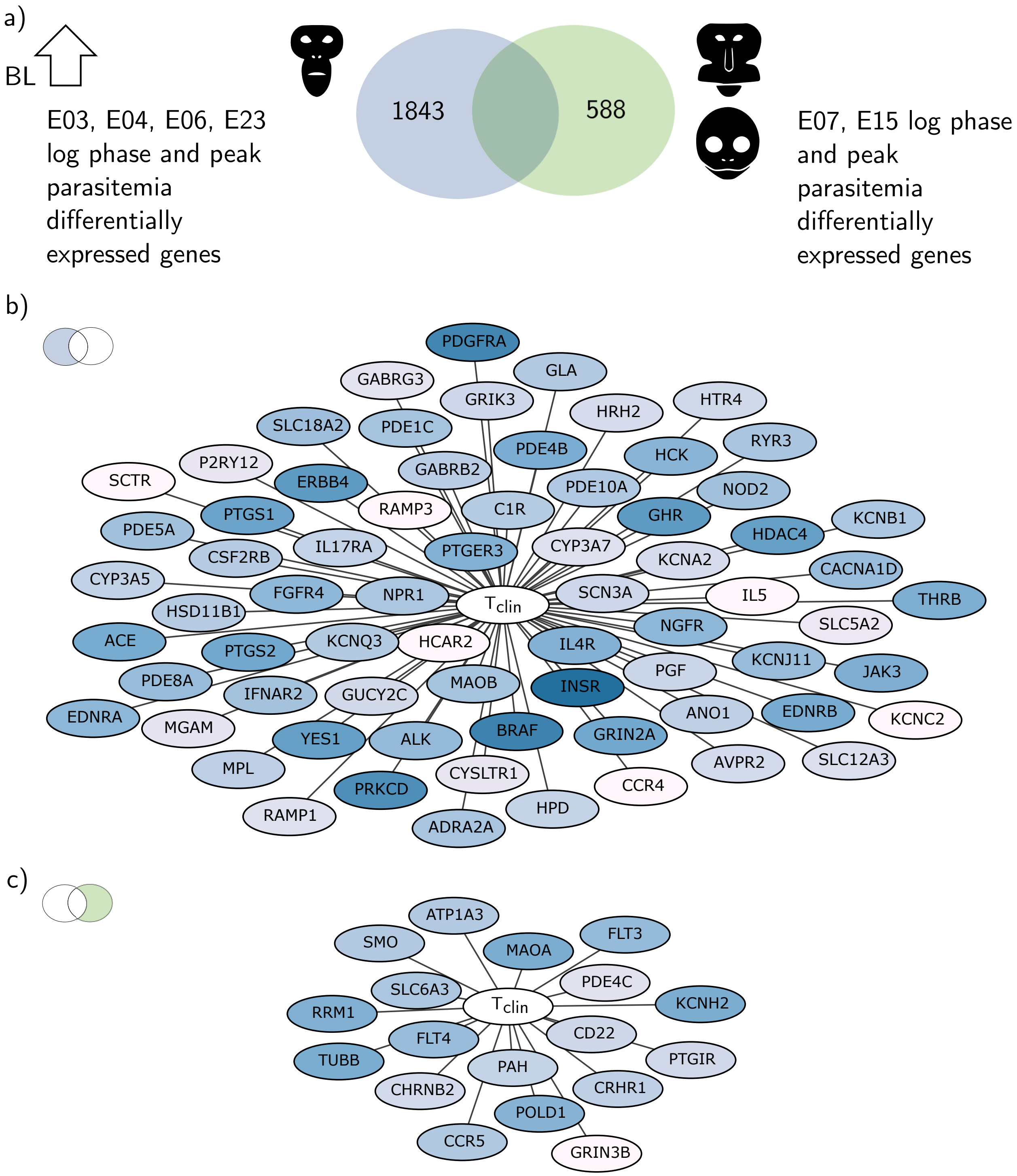


Figure 3.2: Rising and peak parasitemia targets. a) The diagram represents the comparison between time points of log phase and acute parasitemia for resilient and non-resilient hosts. b) Targets from the Pharos database for the genes unique to the non-resilient hosts were found. c) Targets from the Pharos database for the genes unique to the resilient hosts were found. The color of the target nodes represents the Knowledge Availability Score, a combined data availability measure.

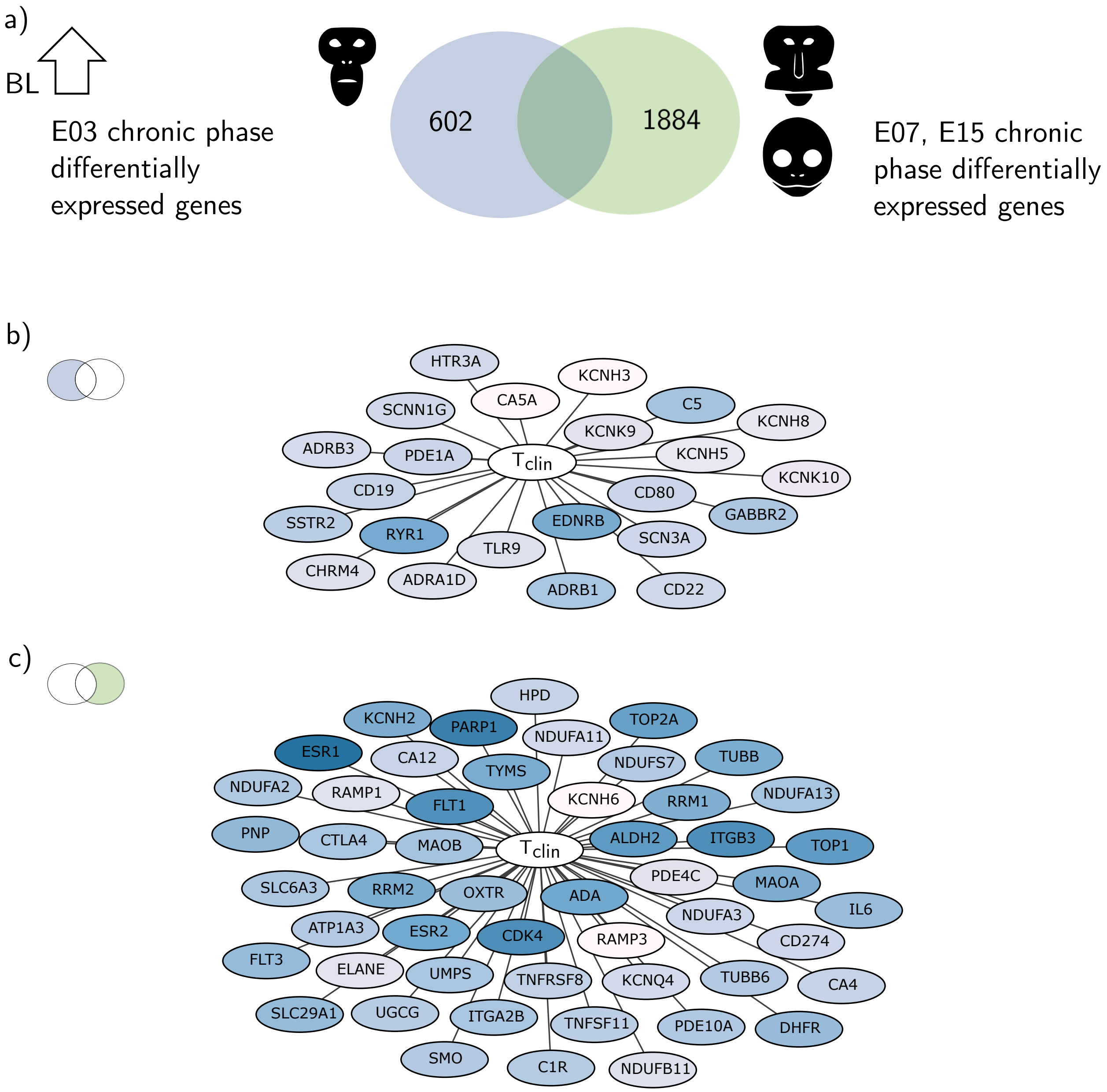
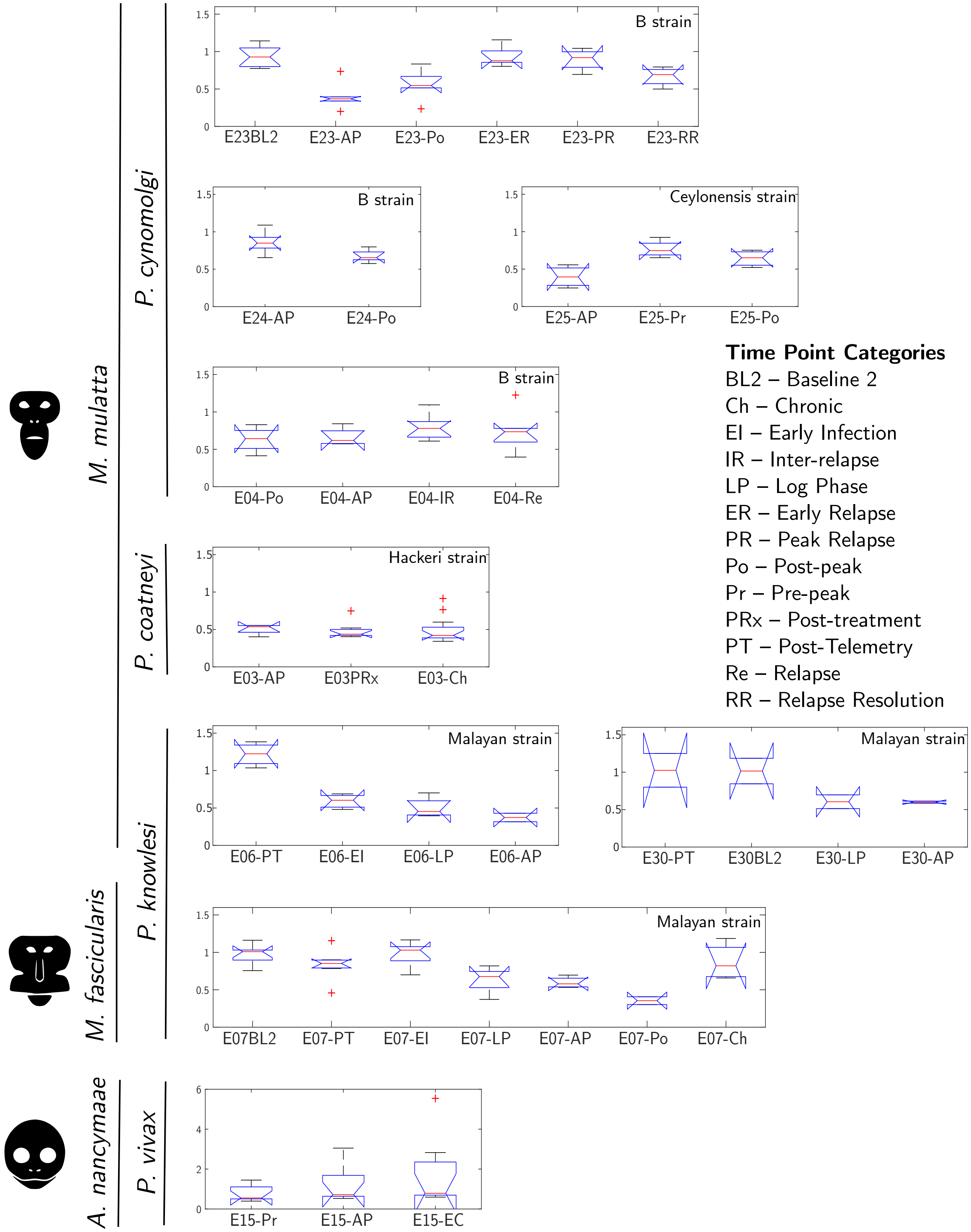


Figure 3.3: Chronic phase targets. a) The diagram represents the comparison between time points which were log phase and acute parasitemia for resilient and non-resilient hosts. b) Targets from the Pharos database for the genes unique to the non-resilient hosts were found. c) Targets from the Pharos database for the genes unique to the resilient hosts were found. The color of the target nodes represents the Knowledge Availability Score, a combined data availability measure.



Time Point Categories
BL2 – Baseline 2
Ch – Chronic
EI – Early Infection
IR – Inter-relapse
LP – Log Phase
ER – Early Relapse
PR – Peak Relapse
Po – Post-peak
Pr – Pre-peak
PRx – Post-treatment
PT – Post-Telemetry
Re – Relapse
RR – Relapse Resolution

Figure 3.4: Fold change in REV-ERBα expression across all time points and all experiments.

Chapter 4

Discussion

4.1 Overview

The SKED framework, which incorporates data primitives for data harmonization and uses object-oriented analysis and design (OOAD) to promote re-usable and reproducible quantitative analysis, was used to manage data analysis with a large systems biology project to produce meaningful, testable results. The project involved multiple -omic measurements over the course of controlled malaria infection experiments. The traditional differential expression analysis was extended to incorporate knowledge from publicly available databases to produce predictions concerning molecular targets for further investigation. A data mining approach was also used to discover if any FDA-approved drugs were known to modulate (activate or inhibit) these targets.

SKED provides a framework for data analysis management instead of just data management. Using data primitives enables a researcher to begin analysis without having to learn about a complicated, underlying data format. Some implications for data analysis management in the growing field of systems medicine and modern healthcare are discussed.

4.2 Resilience targets

The SKED framework was able to identify targets to promote the resilient host response from multiple -omic data sources. Supporting evidence from more than one tissue and more than one data type provides stronger support for a hypothesis than using one data type alone. In this study, supporting evidence was found for the use of the PDGFR inhibitor, imatinib

(Gleevec®) to promote a resilient response. Imatinib is taken orally and is well tolerated [88]. This drug was first widely used in chronic myeloid leukemia (CML) specifically for blocking the activity of BCR-ABL fusion protein [89]. The PDGF and its receptors, PDGFRA and PDGFRB, were found to be up regulated in multiple infection stages in the non-resilient host in multiple tissues in the experiments investigated here.

Because circadian rhythms are known to be influential on disease outcomes immune system function, an important circadian clock gene, REVERBα was also investigated in the different tissues and data types [90]. Unlike many protein targets, REVERBα is known to have small molecule activators, specifically GSK4112 and derivatives [91]. These are known to improve glucose homeostasis in obese mice and inhibit inflammatory response [91, 92]. Evidence is shown here for their continued investigate in clinical and experimental studies.

4.3 Analysis Management Using SKED

Data primitives enable the integration of heterogeneous data and the modular analysis and reuse of code. Current standards and ontologies function like puzzle pieces, which allow connections only to a reduced set of elements (as depicted in the left side of Figure 4.1). Using data primitives, however, enables us to use data like LEGO® building blocks, in which communication can occur between any two standards or combination of standards. The mathematical definitions of data primitives, their implementation in JSON formats and a relational database, as well as their utility in complex multi-omic systems biology studies of malaria infection have been described.

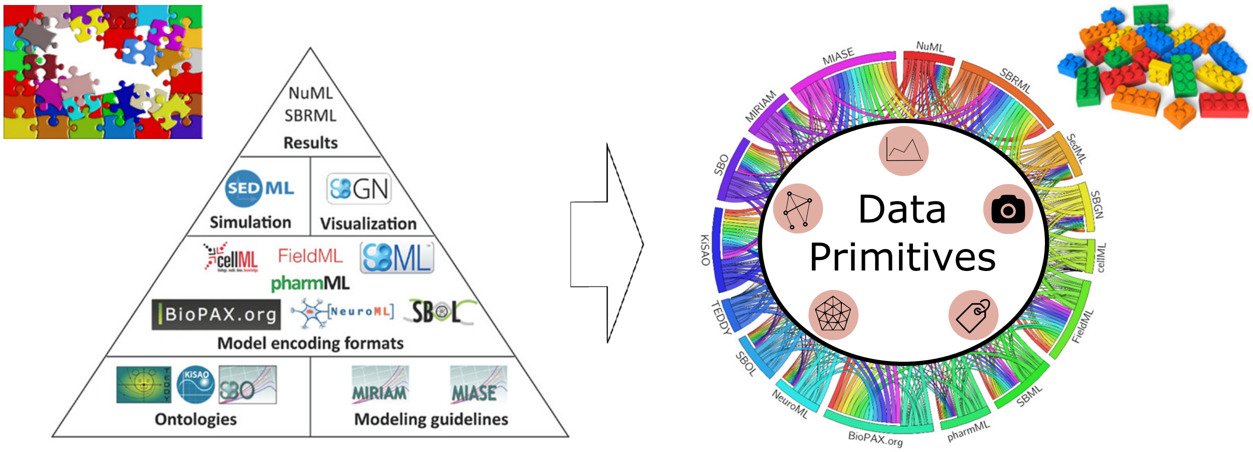


Figure 4.1: Current standards to data primitives is like moving from arranging puzzle pieces to building blocks. Pyramid modified from Palsson et al. [93].

Analysis beginning with data primitives is an extension of current analysis types as summarized in Table 4.1. The JSON data primitive storage format is an extension and addition to many common storage and data organization formats. Analysis of data using data primitives can be seen to be easily extensible to other experimental, analytical and modeling systems.

Table 4.1: Other storage formats and types of analysis for data primitives

Data Primitive	Types	Common Storage Formats	Types of Analysis
Time Series	-	Time Series Database (TSDB)	Frequency-domain: spectral and wavelet analysis Time-domain: correlation and cross-sectional analysis Dynamical systems analysis
Graph	Directed/undirected graph Hypergraph, bipartite graph Ranked list	CSV file	Optimization: linear/nonlinear programming Network: reconstruction (gene regulatory, metabolic)
Polygonal Mesh	Polygonal Volumetric	WRL, 3DMLW	Geometric analysis
Image	2D 3D	BMP, TIFF, JPEG, GIF 3DS	Object-based image analysis: medical diagnostic, geographic
Metadata	-	BioCompute Objects Common Working Language (CWL)	Meta-analysis any nested Data Primitive

4.4 Future Goals for Bioinformatics programming and analysis

Because using data primitives allows the focus to shift from data storage and sharing to model reuse and analysis interpretation, integrative systems studies in biology and medicine need an expanded set of guiding principles for data and model analysis. These goals are summarized by the acronym STRRAITE, and are explained in Table 4.2.

Table 4.2: Data analysis management guidelines

S	Scaleable	should work for large and small datasets
T	Transportable	should work with different species and different time scales as appropriate
R	Reliable	dependable, software reliability engineering and software quality definition (Consortium for IT Software Quality(CISQ) - reliability, efficiency, security, maintainability, size)
R	Reproducible	allows different researchers to achieve the same results
R	Replicable	same implications and meaning of program output
A	Actionable	leads to conclusions that promote further experimentation or treatments
I	Interpretable	data primitives could be easily repurposed to work with many different visualization tools
T	Transparent	traceable, known provenance of data exists
E	Extensible	could be used with new datasets and data types that have not been discovered or measured yet

The lack of these aspects without significant effort in many large data projects in the life sciences indicates a need for interoperable data types for both analysis input and output. There is a need for atomic units of data representation which have standard formats so that the underlying data structure can be easily understood by both humans and machines. This simplification will enable data analysis pipelines to be easily repurposed from one environment to another and to be more easily connected in more sophisticated computational pipelines.

4.5 Future Uses of Data Primitives

4.5.1 Introduction to P5 Medicine

Health informatics is a significant underlying component of the Triple Aim of health care which has goals of simultaneously improving the patient experience, improving the health of populations and reducing per capita costs [94]. As health care informatics begins to incorporate more P4 (predictive, preventative, personalized, participatory) systems medicine approaches, and to include patient measurements that have traditionally been used in research (genetic profiles, and other multi-omic technologies) [95], health informatics must integrate large heterogeneous datasets that cross temporal and spatial scales (see Figure 1.1), to accomplish the goals of the Triple Aim. Most efforts so far have focused on creating detailed, workable solutions to manage these datasets in isolation but few have focused on their reconciliation. The magnitude of the problem is described in Figure 1.1. Molecular, cellular, clinical, environmental and epidemiological data have all been gathered in vast quantities to describe both individual patients and to characterize diseases, but this data has not resulted in significant improvements to individual patient care or reduced care costs. Currently there is no robust, scalable method to incorporate clinical information and other multi-omic datasets for routine patient care. To address the informatics problems underlying P4 systems medicine and the Triple Aim of health care, we introduced the Scientific Knowledge Extraction from Data (SKED) architecture, a technology-agnostic framework to minimize the overhead of data integration, facilitate the reuse of analytical pipelines, and guarantee of reproducibility of quantitative results.

4.5.2 SKED in Research

We implemented the SKED framework to study the pathogenesis of malaria using multi-omic data (transcriptomics, proteomics, metabolomics, lipidomics), immunological data (flow cytometry, cytokine ELISAs), and clinical measurements (doctor assessments, and physiotelemetry), as part of the Malaria Host-Pathogen Interaction Center (MaHPIC). We investigated the host-pathogen interactions between non-human primates hosts and *Plasmodium* parasites as models for human malarial infections.

We were able to combine high frequency telemetry signals (ex. ECG) with other measurements taken over the course of an infection [39], for example metabolomics (daily), transcriptomics and immune response data (various times throughout the infection). Using data primitives allowed us to easily perform different types of meta-dimensional analysis as described by [32], including concatenation-based analysis, where multiple data types are combined before analysis.

One of the most powerful aspects of SKED was the ability to harmonize data over multiple time scales and multiple spatial scales and we envision this aspect to become even more important as additional real-time, continuous data measures (as could easily by gathered by e.g. a cell phone sensor) become available.

4.6 SKED Enables P4 Medicine

Combining data types before analysis (for e.g. network reconstruction) is time-consuming and difficult, but can result in unique insights, for example predicting HDL cholesterol levels from genotype and gene expression levels [96, 97].

Because the SKED framework provides a general, scalable solution to the problems of data integration and data harmonization across multiple time and spatial scales, patient treatments may be made more predictive as powerful algorithms that are able to identify the most important biomarkers for a disease are found. Algorithms designed for one type of data may be effortlessly repurposed for use on another data type. Concatenation-based analysis thus becomes more feasible and could allow for acceleration of biomarker discovery, since multiple-omic datasets may be combined in analysis [32].

Chronic diseases (diabetes, cardiovascular disease, etc.) are now a major cause of mortality in many countries; thus, biomarker discovery for early detection and intervention [98] is a pressing need. SKED provides a workable solution to combine the complicated multi-omic data sets that must be gathered from many people in order to determine the most significant molecular predictors for these diseases.

As predictions about the onset of chronic disease improve, these accurate predictions could enable earlier, preventative treatments to be undertaken. The data integration capabilities of SKED establish a foundation for the use of more personalized medicine, as personalized medicine begins to make more use of genomic and other large-scale datasets to describe a patient. As the “individualome” of each patient is created and becomes more complicated, patients could be able to have a more active part in managing their own health and outcomes [2]. Patients will thus be better able to manage their own health and have a more active role in preventing the chronic disease they may be most susceptible to.

4.7 SKED enables the Triple Aim of Health Care

Because the SKED framework solves many problems associated with data integration and harmonization at multiple levels in health information analysis, it is aligned with achieving the goals of the Triple Aim in health informatics [94]. [99] identify three principles that successfully guided organizations working on the implementation of the Triple Aim.

The first guiding principle was establishing a foundation for population management to determine which populations (i.e. elderly, low-income, etc.) will be the focus of an intervention. A system integrator (e.g. a local or state health department) gathers resources and coordinates work in this step. The system integrator is also responsible for iterative improvements and testing to determine when and how the most short- and long-term progress has been made. Such analysis can be done easily and effectively on the kinds of heterogeneous data that describe health outcomes using SKED. SKED allows algorithms used in one context to be extended to others so that the most advanced up-to-date methods may be applied to any dataset to determine the effectiveness of an intervention.

The second guiding principle was to effectively manage services at scale. The SKED framework allows for the analysis of all types of data (epidemiological, clinical, etc.) at different scales. Automated analysis with SKED could allow the most important services and their beneficial effects to be identified and subsequently implemented. The results of implementing different health services at different scales may be studied and the most effective overall plans could be enabled through the use of SKED.

Last, Whittington et al. [99] identified the need for a learning system to determine which measures have had the most effect. The authors propose that cycles of iterative testing are needed to investigate the performance of different interventions and treatments in populations and individuals. Using data primitives in SKED can make such analyses more accurate and consistent. A Resource Allocation Service (RAS) could simplify finding analysis pipelines and data for comparison. For example, having data stored as data primitives could enable a public health official to easily integrate and compare data sets from different counties and states about the spread of an emerging infectious disease.

The power of SKED is not limited to multi-omic analysis and data integration, but also can be extended to enable the goals of the Triple Aim of health care.

4.7.1 Conclusion

Through more efficient management of patient clinical records and patient data at a systems medicine level, SKED could advance patient care towards more predictive and preventative measures that offer the ability to improve individual care, improve overall outcomes, and reduce overall costs associated with patient treatment. We have shown the usefulness of SKED in the interpretation of multi-omic data in clinical disease manifestations and our approach could be extended to general clinical and health management settings. Ultimately, the SKED framework has the ability to transform how complicated datasets for patients are managed and analyzed.

Chapter 5

Conclusions

Scientific and clinical studies often incorporate datasets that cross multiple spatial and temporal scales to describe a particular phenomenon. This is a particular challenge for modeling since an analytical method developed for one data type cannot be easily re-purposed for uses with an integrated dataset. In order to overcome these obstacles, SKED, including the use of data primitives is proposed as a common currency between analytical methods and modeling tools and an extensible object-oriented analysis and design . The data primitives identified are time series, annotated graph, image, and polygonal mesh, with associated metadata.

The induction of disease resilience to an infection could offer mechanisms to treat a host without exerting evolutionary pressure on the infecting agent. This also has implications for chronic diseases,like cancer, for which treatments could be designed that help the patient tolerate the disease giving treatments more time to be effective [57]. Disease resilience to malaria could be extremely important in high transmission settings where children are the first victims of malaria [100, 101]. This analysis has proposed several molecular targets and interventions for further validation including the use of imatinib to stimulate bone marrow responses.

The use of data primitives as inputs and outputs of algorithms promotes interoperability, scalability, and reproducibility in scientific studies. Data primitives were used in a multi-omic, multi-scale systems biology study of malaria infection to perform integrative analysis quickly and efficiently. Using data primitives for communication between analytical methods facilitates reproducible analyses of complex multi-scale datasets in a modular fashion.

Data primitives were designed to be minimalistic and modular. They provide unified structures for raw, processed, and computational data of different sizes and can act as reusable modular outputs and inputs of analytical pipelines. They allow data integration of multiple temporal and spatial scales and increase the reliability of a computational pipeline. They encompass and expand upon current standards for data and model sharing to increase the usability and reuseability of existing structures. Computational researchers and data analysts using data primitives are then able to focus on the investigation and interpretation of data as well as the design of new experiments and analysis pipelines. Their use has implications for the future of automated knowledge generation and the use of systems medicine approaches in health care.

Bibliography

[1] Scott D. Kahn. On the Future of Genomic Data. *Science*, 331(6018):728–729, February 2011. ISSN 0036-8075, 1095-9203. doi: 10.1126/science.1197891. URL `http://science.sciencemag.org.proxy-remote.galib.uga.edu/content/331/6018/728`.

[2] Khader Shameer, Marcus A Badgeley, Riccardo Miotto, Benjamin S Glicksberg, Joseph W Morgan, and Joel T Dudley. Translational bioinformatics in the era of real-time biomedical, health care and wellness data streams. *Briefings in bioinformatics*, page bbv118, 2016.

[3] Muin J. Khoury, Marta Gwinn, W. David Dotson, and Sheri D. Schully. Knowledge integration at the center of genomic medicine. *Genetics in Medicine*, 14(7): gim201243, May 2012. ISSN 1530-0366. doi: 10.1038/gim.2012.43. URL `https://www-nature-com.proxy-remote.galib.uga.edu/articles/gim201243`.

[4] Duncan Ayers and Philip J Day. Systems medicine: the application of systems biology approaches for modern medical research and drug development. *Molecular biology international*, 2015, 2015.

[5] Hiroaki Kitano and others. *Foundations of systems biology.* MIT press Cambridge, 2001.

[6] Gonen Ashkenasy, Thomas M Hermans, Sijbren Otto, and Annette F Taylor. Systems chemistry. *Chemical Society Reviews*, 2017.

[7] Lei Xie, Eli J Draizen, and Philip E Bourne. Harnessing big data for systems pharmacology. *Annual review of pharmacology and toxicology*, 57:245–262, 2017.

[8] Jonathan Friedman and Jeff Gore. Ecological systems biology: The dynamics of interacting populations. *Current Opinion in Systems Biology*, 1:114–121, 2017.

[9] Leroy Hood, James R Heath, Michael E Phelps, and Biaoyang Lin. Systems biology and new technologies enable predictive and preventative medicine. *Science*, 306(5696): 640–643, 2004.

[10] Leroy Hood and Mauricio Flores. A personal view on systems medicine and the emergence of proactive P4 medicine: predictive, preventive, personalized and participatory. *New biotechnology*, 29(6):613–624, 2012.

[11] Mark D. Wilkinson, Michel Dumontier, IJsbrand Jan Aalbersberg, Gabrielle Appleton, Myles Axton, Arie Baak, Niklas Blomberg, Jan-Willem Boiten, Luiz Bonino da Silva Santos, Philip E. Bourne, Jildau Bouwman, Anthony J. Brookes, Tim Clark, Merc Crosas, Ingrid Dillo, Olivier Dumon, Scott Edmunds, Chris T. Evelo, Richard Finkers, Alejandra Gonzalez-Beltran, Alasdair J.G. Gray, Paul Groth, Carole Goble, Jeffrey S. Grethe, Jaap Heringa, Peter A.C t Hoen, Rob Hooft, Tobias Kuhn, Ruben Kok, Joost Kok, Scott J. Lusher, Maryann E. Martone, Albert Mons, Abel L. Packer, Bengt Persson, Philippe Rocca-Serra, Marco Roos, Rene van Schaik, Susanna-Assunta Sansone, Erik Schultes, Thierry Sengstag, Ted Slater, George Strawn, Morris A. Swertz, Mark Thompson, Johan van der Lei, Erik van Mulligen, Jan Velterop, Andra Waagmeester, Peter Wittenburg, Katherine Wolstencroft, Jun Zhao, and Barend Mons. The FAIR Guiding Principles for scientific data management and stewardship. *Scientific Data*, 3:160018, March 2016. URL `http://dx.doi.org/10.1038/sdata.2016.18`.

[12] Michael Hucka, David P Nickerson, Gary D Bader, Frank T Bergmann, Jonathan Cooper, Emek Demir, Alan Garny, Martin Golebiewski, Chris J Myers, Falk Schreiber, and others. Promoting coordinated development of community-based information standards for modeling in biology: the COMBINE initiative. *Frontiers in bioengineering and biotechnology*, 3, 2015.

[13] Ronald Margolis, Leslie Derr, Michelle Dunn, Michael Huerta, Jennie Larkin, Jerry Sheehan, Mark Guyer, and Eric D. Green. The National Institutes of Health's Big Data to Knowledge (BD2k) initiative: capitalizing on biomedical big data. *Journal of the American Medical Informatics Association*, 21(6):957–958, November 2014. ISSN 1067-5027. doi: 10.1136/amiajnl-2014-002974. URL `https://academic.oup.com/jamia/article/21/6/957/2909314`.

[14] Immanuel Kant. *The Critique of Pure Reason*. 2 edition. URL `https://www.gutenberg.org/files/4280/4280-h/4280-h.htm`.

[15] Samuel Clarke, Sir Richard Bulkeley, and Gottfried Wilhelm Freiherr von Leibniz. *A Collection of Papers, which Passed Between the Late Learned Mr. Leibnitz, and Dr. Clarke, in the Years 1715 and 1716: Relating to the Principles of Natural Philosophy and Religion. With an Appendix. To which are Added, Letters to Dr. Clarke Concerning Liberty and Necessity; from a Gentleman of the University of Cambridge: with the Doctor's Answers to Them. Also Remarks Upon a Book, Entituled, A Philosophical Enquiry Concerning Human Liberty.* James Knapton, at the Crown in St. Paul's Church-Yard., 1717. Google-Books-ID: _RUHAAAAQAAJ.

[16] Robert Rosen. *Fundamentals of measurement and representation of natural systems*, volume 1. Elsevier Science Ltd, 1978.

[17] Robert Rosen. *Anticipatory Systems*. IFSR International Series on Systems Science and Engineering. Springer, New York, NY, 2012. ISBN 978-1-4614-1268-7 978-1-4614-1269-4. doi: 10.1007/978-1-4614-1269-4_6. URL `http://link.springer.com/chapter/10.1007/978-1-4614-1269-4_6`.

[18] Aloisius H Louie. *More than life itself: a synthetic continuation in relational biology*, volume 1. Walter de Gruyter, 2009.

[19] Dusty Phillips. *Python 3 Object-Oriented Programming*. Packt Publishing Ltd., Birmingham, UK, second edition, August 2015.

[20] Steven McConnell. *Code Complete*. Microsoft Press, Redmond, WA, twenty-fourth edition, February 2015.

[21] Robert Martin. *Clean code: a handbook of agile software craftsmanship*. Pearson Education, Boston, MA, 2009.

[22] Roger D Peng. Reproducible research in computational science. *Science*, 334(6060): 1226–1227, 2011.

[23] Geir Kjetil Sandve, Anton Nekrutenko, James Taylor, and Eivind Hovig. Ten Simple Rules for Reproducible Computational Research. *PLOS Computational Biology*, 9(10):e1003285, October 2013. ISSN 1553-7358. doi: 10.1371/journal.pcbi.1003285. URL `http://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1003285`.

[24] Keith A. Baggerly and Kevin R. Coombes. Deriving chemosensitivity from cell lines: Forensic bioinformatics and reproducible research in high-throughput biology. *The Annals of Applied Statistics*, 3(4):1309–1334, 2009. doi: 10.1214/09-AOAS291. URL `http://dx.doi.org/10.1214/09-AOAS291`.

[25] Daniel Garijo, Sarah Kinnings, Li Xie, Lei Xie, Yinliang Zhang, Philip E. Bourne, and Yolanda Gil. Quantifying Reproducibility in Computational Biology: The Case of the Tuberculosis Drugome. *PLoS ONE*, 8(11):e80278, November 2013. ISSN 1932-6203. doi: 10.1371/journal.pone.0080278. URL `http://dx.plos.org/10.1371/journal.pone.0080278`.

[26] Dagmar Waltemath, Ron Henkel, Felix Winter, and Olaf Wolkenhauer. Reproducibility of Model-Based Results in Systems Biology. In Ale Prokop and Bla Csuks, editors, *Systems Biology*, pages 301–320. Springer Netherlands, Dordrecht, 2013. ISBN

978-94-007-6802-4 978-94-007-6803-1. URL http://link.springer.com/10.1007/978-94-007-6803-1_10.

[27] Philip B. Stark. Before reproducibility must come preproducibility, May 2018. URL http://www.nature.com/articles/d41586-018-05256-0.

[28] Marcus R. Munaf, Brian A. Nosek, Dorothy V. M. Bishop, Katherine S. Button, Christopher D. Chambers, Nathalie Percie du Sert, Uri Simonsohn, Eric-Jan Wagenmakers, Jennifer J. Ware, and John P. A. Ioannidis. A manifesto for reproducible science. *Nature Human Behaviour*, 1(1):0021, January 2017. ISSN 2397-3374. doi: 10.1038/s41562-016-0021. URL http://www.nature.com/articles/s41562%20016%200021.

[29] Anne-Laure Boulesteix. Ten Simple Rules for Reducing Overoptimistic Reporting in Methodological Computational Research. *PLOS Computational Biology*, 11(4):e1004191, April 2015. ISSN 1553-7358. doi: 10.1371/journal.pcbi.1004191. URL http://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1004191.

[30] Nicolas P. Rougier, Konrad Hinsen, Frdric Alexandre, Thomas Arildsen, Lorena A. Barba, Fabien C. Y. Benureau, C. Titus Brown, Pierre de Buyl, Ozan Caglayan, Andrew P. Davison, Marc-Andr Delsuc, Georgios Detorakis, Alexandra K. Diem, Damien Drix, Pierre Enel, Benot Girard, Olivia Guest, Matt G. Hall, Rafael N. Henriques, Xavier Hinaut, Kamil S. Jaron, Mehdi Khamassi, Almar Klein, Tiina Manninen, Pietro Marchesi, Daniel McGlinn, Christoph Metzner, Owen Petchey, Hans Ekkehard Plesser, Timothe Poisot, Karthik Ram, Yoav Ram, Etienne Roesch, Cyrille Rossant, Vahid Rostami, Aaron Shifman, Joseph Stachelek, Marcel Stimberg, Frank Stollmeier, Federico Vaggi, Guillaume Viejo, Julien Vitay, Anya E. Vostinar, Roman Yurchak, and Tiziano Zito. Sustainable computational science: the ReScience initiative. *PeerJ*

Computer Science, 3:e142, December 2017. ISSN 2376-5992. doi: 10.7717/peerj-cs.142. URL `https://peerj.com/articles/cs-142`.

[31] Doug Laney. 3d data management: Controlling data volume, velocity and variety. *META group research note*, 6(70):1, 2001.

[32] Marylyn D. Ritchie, Emily R. Holzinger, Ruowang Li, Sarah A. Pendergrass, and Dokyoon Kim. Methods of integrating data to uncover genotypephenotype interactions. *Nature Reviews Genetics*, 16(2):85–97, February 2015. ISSN 1471-0064. doi: 10.1038/nrg3868. URL `https://www.nature.com/articles/nrg3868`.

[33] Yi H. Yan, Elizabeth D. Trippe, and Juan B. Gutierrez. A Method for Massively Parallel Analysis of Time Series. *arXiv:1612.08759 [q-bio]*, December 2016. URL `http://arxiv.org/abs/1612.08759`. arXiv: 1612.08759.

[34] *World malaria report 2017*. World Health Organization, Geneva, November 2017. ISBN 978-92-4-156552-3. URL `http://www.who.int/malaria/publications/world-malaria-report-2017/report/en/`.

[35] Margaret A. Phillips, Jeremy N. Burrows, Christine Manyando, Rob Hooft van Huijsduijnen, Wesley C. Van Voorhis, and Timothy N. C. Wells. Malaria. *Nature Reviews Disease Primers*, 3:17050, August 2017. ISSN 2056-676X. doi: 10.1038/nrdp.2017.50. URL `https://www.nature.com/articles/nrdp201750`.

[36] Alimuddin Zumla, Martin Rao, Robert S Wallis, Stefan H E Kaufmann, Roxana Rustomjee, Peter Mwaba, Cris Vilaplana, Dorothy Yeboah-Manu, Jeremiah Chakaya, Giuseppe Ippolito, Esam Azhar, Michael Hoelscher, and Markus Maeurer. Host-directed therapies for infectious diseases: current status, recent progress, and future prospects. *The Lancet Infectious Diseases*, 16(4):e47–e63, April 2016. ISSN 1473-3099. doi: 10.1016/S1473-3099(16)00078-5. URL `http://www.sciencedirect.com/science/article/pii/S1473309916000785`.

[37] Elizabeth K. K. Glennon, Selasi Dankwa, Joseph D. Smith, and Alexis Kaushansky. Opportunities for Host-targeted Therapies for Malaria. *Trends in Parasitology*, August 2018. ISSN 1471-4922. doi: 10.1016/j.pt.2018.07.011. URL http://www.sciencedirect.com/science/article/pii/S1471492218301491.

[38] Brenda Y. Torres, Jose Henrique M. Oliveira, Ann Thomas Tate, Poonam Rath, Katherine Cumnock, and David S. Schneider. Tracking Resilience to Infections by Mapping Disease Space. *PLOS Biology*, 14(4):e1002436, April 2016. ISSN 1545-7885. doi: 10.1371/journal.pbio.1002436. URL https://journals.plos.org/plosbiology/article?id=10.1371/journal.pbio.1002436.

[39] Chester Joyner, Alberto Moreno, Esmeralda V. S. Meyer, Monica Cabrera-Mora, Jessica C. Kissinger, John W. Barnwell, and Mary R. Galinski. Plasmodium cynomolgi infections in rhesus macaques display clinical and parasitological features pertinent to modelling vivax malaria pathology and relapse infections. *Malaria Journal*, 15: 451, September 2016. ISSN 1475-2875. doi: 10.1186/s12936-016-1480-6. URL https://doi.org/10.1186/s12936-016-1480-6.

[40] Ivo Mueller, Mary R Galinski, J Kevin Baird, Jane M Carlton, Dhanpat K Kochar, Pedro L Alonso, and Hernando A del Portillo. Key gaps in the knowledge of Plasmodium vivax, a neglected human malaria parasite. *The Lancet Infectious Diseases*, 9 (9):555–566, September 2009. ISSN 1473-3099. doi: 10.1016/S1473-3099(09)70177-X. URL http://www.sciencedirect.com/science/article/pii/S147330990970177X.

[41] Lars Rberg, Derek Sim, and Andrew F. Read. Disentangling Genetic Variation for Resistance and Tolerance to Infectious Diseases in Animals. *Science*, 318 (5851):812–814, November 2007. ISSN 0036-8075, 1095-9203. doi: 10.1126/science.1148526. URL http://science.sciencemag.org.proxy-remote.galib.uga.edu/content/318/5851/812.

[42] Temitope W. Ademolue and Gordon A. Awandare. Evaluating antidisease immunity to malaria and implications for vaccine design. *Immunology*, 153(4):423–434, April 2018. ISSN 1365-2567. doi: 10.1111/imm.12877. URL http://onlinelibrary.wiley.com/doi/abs/10.1111/imm.12877.

[43] Miguel P. Soares, Luis Teixeira, and Luis F. Moita. Disease tolerance and immunity in host protection against infection. *Nature Reviews Immunology*, 17(2):83, February 2017. ISSN 1474-1741. doi: 10.1038/nri.2016.136. URL https://www-nature-com.proxy-remote.galib.uga.edu/articles/nri.2016.136.

[44] MiguelP. Soares. Nuts and Bolts of Disease Tolerance. *Immunity*, 41(2):176–178, August 2014. ISSN 1074-7613. doi: 10.1016/j.immuni.2014.07.011. URL http://www.sciencedirect.com/science/article/pii/S1074761314002696.

[45] Temitope W. Ademolue, Yaw Aniweh, Kwadwo A. Kusi, and Gordon A. Awandare. Patterns of inflammatory responses and parasite tolerance vary with malaria transmission intensity. *Malaria Journal*, 16(1):145, December 2017. ISSN 1475-2875. doi: 10.1186/s12936-017-1796-x. URL http://malariajournal.biomedcentral.com/articles/10.1186/s12936-017-1796-x.

[46] Louis H. Miller, Hans C. Ackerman, Xin-zhuan Su, and Thomas E. Wellems. Malaria biology and disease pathogenesis: insights for new treatments. *Nature Medicine*, 19 (2):156–167, February 2013. ISSN 1546-170X. doi: 10.1038/nm.3073. URL https://www.nature.com/articles/nm.3073.

[47] Ricardo T. Gazzinelli, Parisa Kalantari, Katherine A. Fitzgerald, and Douglas T. Golenbock. Innate sensing of malaria parasites. *Nature Reviews Immunology*, 14 (11):744, November 2014. ISSN 1474-1741. doi: 10.1038/nri3742. URL https://www-nature-com.proxy-remote.galib.uga.edu/articles/nri3742.

[48] H. A. Mulder and H. Rashidi. Selection on resilience improves disease resistance and tolerance to infections. *Journal of Animal Science*, 95(8):3346–3358, August 2017. ISSN 0021-8812. doi: 10.2527/jas.2017.1479. URL http://academic.oup.com/jas/article/95/8/3346/4702448.

[49] Steve Bishop. A consideration of resistance and tolerance for ruminant nematode infections. *Frontiers in Genetics*, 3, 2012. ISSN 1664-8021. doi: 10.3389/fgene.2012.00168. URL https://www.frontiersin.org/articles/10.3389/fgene.2012.00168/full#Box1.

[50] G. A. A. Albers, G. D. Gray, L. R. Piper, J. S. F. Barker, L. F. Le Jambre, and I. A. Barger. The genetics of resistance and resilience to Haemonchus contortus infection in young merino sheep. *International Journal for Parasitology*, 17(7):1355–1363, October 1987. ISSN 0020-7519. doi: 10.1016/0020-7519(87)90103-2. URL http://www.sciencedirect.com/science/article/pii/0020751987901032.

[51] S. A. Bisset and C. A. Morris. Feasibility and implications of breeding sheep for resilience to nematode challenge. *International Journal for Parasitology*, 26(8):857–868, August 1996. ISSN 0020-7519. doi: 10.1016/S0020-7519(96)80056-7. URL http://www.sciencedirect.com/science/article/pii/S0020751996800567.

[52] Kumudika de Silva, Karren Plain, Auriol Purdie, Douglas Begg, and Richard Whittington. Defining resilience to mycobacterial disease: Characteristics of survivors of ovine paratuberculosis. *Veterinary Immunology and Immunopathology*, 195:56–64, January 2018. ISSN 0165-2427. doi: 10.1016/j.vetimm.2017.11.008. URL http://www.sciencedirect.com/science/article/pii/S0165242717304828.

[53] Manuel Gesto, Lone Madsen, Nikolaj R. Andersen, and Alfred Jokumsen. Differences in stress and disease resilience related to emergence time for first feeding in farmed rainbow trout (Oncorhynchus mykiss). *Journal of Experimental Biology*, 221

(8):jeb174623, April 2018. ISSN 0022-0949, 1477-9145. doi: 10.1242/jeb.174623. URL http://jeb.biologists.org/content/221/8/jeb174623.

[54] Andrea B. Doeschl-Wilson, Beatriz Villanueva, and Ilias Kyriazakis. The first step toward genetic selection for host tolerance to infectious pathogens: obtaining the tolerance phenotype through group estimates. *Frontiers in Genetics*, 3, December 2012. ISSN 1664-8021. doi: 10.3389/fgene.2012.00265. URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3571525/.

[55] Janelle S. Ayres and David S. Schneider. Tolerance of infections. *Annual review of immunology*, 30:271–294, 2012.

[56] Ruslan Medzhitov, David S. Schneider, and Miguel P. Soares. Disease Tolerance as a Defense Strategy. *Science*, 335(6071):936–941, February 2012. ISSN 0036-8075, 1095-9203. doi: 10.1126/science.1214935. URL http://science.sciencemag.org.proxy-remote.galib.uga.edu/content/335/6071/936.

[57] Sheila Rao and Janelle S. Ayres. Resistance and tolerance defenses in cancer: Lessons from infectious diseases. *Seminars in Immunology*, 32(Supplement C):54–61, August 2017. ISSN 1044-5323. doi: 10.1016/j.smim.2017.08.004. URL http://www.sciencedirect.com/science/article/pii/S1044532317300349.

[58] Beatriz Galatas, Quique Bassat, and Alfredo Mayor. Malaria Parasites in the Asymptomatic: Looking for the Hay in the Haystack. *Trends in Parasitology*, 32 (4):296–308, April 2016. ISSN 1471-4922. doi: 10.1016/j.pt.2015.11.015. URL http://www.sciencedirect.com/science/article/pii/S1471492215002597.

[59] Elsa Seixas, Raffaella Gozzelino, ngelo Chora, Ana Ferreira, Gabriela Silva, Rasmus Larsen, Sofia Rebelo, Carmen Penido, Neal R. Smith, Antonio Coutinho, and Miguel P. Soares. Heme oxygenase-1 affords protection against noncerebral forms of severe malaria. *Proceedings of the National Academy of Sciences*, 106(37):15837–15842,

September 2009. ISSN 0027-8424, 1091-6490. doi: 10.1073/pnas.0903419106. URL http://www.pnas.org/content/106/37/15837.

[60] Viktria Jeney, Susana Ramos, Marie-Louise Bergman, Ingo Bechmann, Jasmin Tischer, Ana Ferreira, Virginia Oliveira-Marques, ChrisJ. Janse, Sofia Rebelo, Silvia Cardoso, and MiguelP. Soares. Control of Disease Tolerance to Malaria by Nitric Oxide and Carbon Monoxide. *Cell Reports*, 8(1):126–136, July 2014. ISSN 2211-1247. doi: 10.1016/j.celrep.2014.05.054. URL http://www.sciencedirect.com/science/article/pii/S2211124714004483.

[61] Raffaella Gozzelino, BrunoBezerril Andrade, Rasmus Larsen, NiveaF. Luz, Liviu Vanoaica, Elsa Seixas, Antonio Coutinho, Slvia Cardoso, Sofia Rebelo, Maura Poli, Manoel Barral-Netto, Deepak Darshan, LukasC. Khn, and MiguelP. Soares. Metabolic Adaptation to Tissue Iron Overload Confers Tolerance to Malaria. *Cell Host & Microbe*, 12(5):693–704, November 2012. ISSN 1931-3128. doi: 10.1016/j.chom.2012.10.011. URL http://www.sciencedirect.com/science/article/pii/S1931312812003575.

[62] Chester Joyner, John W. Barnwell, and Mary R. Galinski. No more monkeying around: primate malaria model systems are key to understanding Plasmodium vivax liver-stage biology, hypnozoites, and relapses. *Frontiers in Microbiology*, 6, 2015. ISSN 1664-302X. doi: 10.3389/fmicb.2015.00145. URL https://www.frontiersin.org/articles/10.3389/fmicb.2015.00145/full.

[63] Maren L. Smith and Mark P. Styczynski. Systems Biology-Based Investigation of Host-Plasmodium Interactions. *Trends in Parasitology*, 0(0), May 2018. ISSN 1471-4922, 1471-5007. doi: 10.1016/j.pt.2018.04.003. URL https://www.cell.com/trends/parasitology/abstract/S1471-4922(18)30081-3.

[64] Juan B. Gutierrez, Mary R. Galinski, Stephen Cantrell, and Eberhard O. Voit. From within host dynamics to the epidemiology of infectious disease: Scientific overview

and challenges. *Mathematical Biosciences*, 270:143–155, December 2015. ISSN 0025-5564. doi: 10.1016/j.mbs.2015.10.002. URL http://www.sciencedirect.com/science/article/pii/S0025556415002084.

[65] JSON, . URL https://www.json.org/.

[66] Standard ECMA-404, . URL http://www.ecma-international.org/publications/standards/Ecma-404.htm.

[67] Jenn Riley. UNDERSTANDING METADATA. page 49, January 2017. URL https://www.niso.org/publications/understanding-metadata-2017.

[68] Elizabeth D. Trippe, Jacob B. Aguilar, Yi H. Yan, Mustafa V. Nural, Jessica A. Brady, and Juan B. Gutierrez. Introducing Data Primitives: Data Formats for the SKED Framework. *arXiv:1706.08131 [q-bio]*, June 2017. URL http://arxiv.org/abs/1706.08131. arXiv: 1706.08131.

[69] NCI Dictionary of Cancer Terms, February 2011. URL https://www.cancer.gov/publications/dictionaries/cancer-terms.

[70] MATLAB and Bioinformatics Toolbox Release 2018a, .

[71] Sean Leary. JSON in Java. URL http://stleary.github.io/JSON-java/.

[72] PlasmoDB :: MaHPIC, . URL http://plasmodb.org/plasmo/mahpic.jsp.

[73] John C Rohloff, Amy D Gelinas, Thale C Jarvis, Urs A Ochsner, Daniel J Schneider, Larry Gold, and Nebojsa Janjic. Nucleic Acid Ligands With Protein-like Side Chains: Modified Aptamers and Their Use as Diagnostic and Therapeutic Agents. *Molecular Therapy - Nucleic Acids*, 3:e201, January 2014. ISSN 2162-2531. doi: 10.1038/mtna.2014.49. URL http://www.sciencedirect.com/science/article/pii/S2162253116303365.

[74] Julin Candia, Foo Cheung, Yuri Kotliarov, Giovanna Fantoni, Brian Sellers, Trevor Griesman, Jinghe Huang, Sarah Stuccio, Adriana Zingone, Brd M. Ryan, John S. Tsang, and Anglique Biancotto. Assessment of Variability in the SOMAscan Assay. *Scientific Reports*, 7(1):14248, October 2017. ISSN 2045-2322. doi: 10.1038/s41598-017-14755-5. URL `https://www.nature.com/articles/s41598-017-14755-5`.

[75] Simon Anders and Wolfgang Huber. Differential expression analysis for sequence count data. *Genome Biology*, 11(10):R106, October 2010. ISSN 1474-760X. doi: 10.1186/gb-2010-11-10-r106. URL `http://genomebiology.biomedcentral.com/articles/10.1186/gb-2010-11-10-r106`.

[76] Yoav Benjamini and Yosef Hochberg. Controlling the False Discovery Rate: A Practical and Powerful Approach to Multiple Testing. *Journal of the Royal Statistical Society. Series B (Methodological)*, 57(1):289–300, 1995. ISSN 0035-9246. URL `http://www.jstor.org/stable/2346101`.

[77] Foo Cheung, Giovanna Fantoni, Maria Conner, Brian A Sellers, Yuri Kotliarov, Julin Candia, Katherine Stagliano, and Anglique Biancotto. Web Tool for Navigating and Plotting SomaLogic ADAT Files. *Journal of open research software*, 5, 2017. ISSN 2049-9647. doi: 10.5334/jors.166. URL `https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6017986/`.

[78] Dennis A. Benson, Mark Cavanaugh, Karen Clark, Ilene Karsch-Mizrachi, David J. Lipman, James Ostell, and Eric W. Sayers. GenBank. *Nucleic Acids Research*, 41(D1):D36–D42, January 2013. ISSN 0305-1048. doi: 10.1093/nar/gks1195. URL `https://academic.oup.com/nar/article/41/D1/D36/1068219`.

[79] Alex Bateman, Maria Jesus Martin, Claire ODonovan, Michele Magrane, Emanuele Alpi, Ricardo Antunes, Benoit Bely, Mark Bingley, Carlos Bonilla, Ramona Britto,

Borisas Bursteinas, Hema Bye-A-Jee, Andrew Cowley, Alan Da Silva, Maurizio De Giorgi, Tunca Dogan, Francesco Fazzini, Leyla Garcia Castro, Luis Figueira, Penelope Garmiri, George Georghiou, Daniel Gonzalez, Emma Hatton-Ellis, Weizhong Li, Wudong Liu, Rodrigo Lopez, Jie Luo, Yvonne Lussi, Alistair MacDougall, Andrew Nightingale, Barbara Palka, Klemens Pichler, Diego Poggioli, Sangya Pundir, Luis Pureza, Guoying Qi, Alexandre Renaux, Steven Rosanoff, Rabie Saidi, Tony Sawford, Aleksandra Shypitsyna, Elena Speretta, Edward Turner, Nidhi Tyagi, Vladimir Volynkin, Tony Wardell, Kate Warner, Xavier Watkins, Rossana Zaru, Hermann Zellner, Ioannis Xenarios, Lydie Bougueleret, Alan Bridge, Sylvain Poux, Nicole Redaschi, Lucila Aimo, Ghislaine Argoud-Puy, Andrea Auchincloss, Kristian Axelsen, Parit Bansal, Delphine Baratin, Marie-Claude Blatter, Brigitte Boeckmann, Jerven Bolleman, Emmanuel Boutet, Lionel Breuza, Cristina Casal-Casas, Edouard de Castro, Elisabeth Coudert, Beatrice Cuche, Mikael Doche, Dolnide Dornevil, Severine Duvaud, Anne Estreicher, Livia Famiglietti, Marc Feuermann, Elisabeth Gasteiger, Sebastien Gehant, Vivienne Gerritsen, Arnaud Gos, Nadine Gruaz-Gumowski, Ursula Hinz, Chantal Hulo, Florence Jungo, Guillaume Keller, Vicente Lara, Philippe Lemercier, Damien Lieberherr, Thierry Lombardot, Xavier Martin, Patrick Masson, Anne Morgat, Teresa Neto, Nevila Nouspikel, Salvo Paesano, Ivo Pedruzzi, Sandrine Pilbout, Monica Pozzato, Manuela Pruess, Catherine Rivoire, Bernd Roechert, Michel Schneider, Christian Sigrist, Karin Sonesson, Sylvie Staehli, Andre Stutz, Shyamala Sundaram, Michael Tognolli, Laure Verbregue, Anne-Lise Veuthey, Cathy H. Wu, Cecilia N. Arighi, Leslie Arminski, Chuming Chen, Yongxing Chen, John S. Garavelli, Hongzhan Huang, Kati Laiho, Peter McGarvey, Darren A. Natale, Karen Ross, C. R. Vinayaka, Qinghua Wang, Yuqi Wang, Lai-Su Yeh, and Jian Zhang. UniProt: the universal protein knowledgebase. *Nucleic Acids Research*, 45(D1):D158–D169, January 2017. ISSN 0305-1048. doi: 10.1093/nar/gkw1099. URL `https://academic.oup.com/nar/article/45/D1/D158/2605721`.

[80] Dac-Trung Nguyen, Stephen Mathias, Cristian Bologa, Soren Brunak, Nicolas Fernandez, Anna Gaulton, Anne Hersey, Jayme Holmes, Lars Juhl Jensen, Anneli Karlsson, Guixia Liu, Avi Ma'ayan, Geetha Mandava, Subramani Mani, Saurabh Mehta, John Overington, Juhee Patel, Andrew D. Rouillard, Stephan Schrer, Timothy Sheils, Anton Simeonov, Larry A. Sklar, Noel Southall, Oleg Ursu, Dusica Vidovic, Anna Waller, Jeremy Yang, Ajit Jadhav, Tudor I. Oprea, and Rajarshi Guha. Pharos: Collating protein information to shed light on the druggable genome. *Nucleic Acids Research*, 45(D1):D995–D1002, January 2017. ISSN 0305-1048. doi: 10.1093/nar/gkw1072. URL `https://academic.oup.com/nar/article/45/D1/D995/2605932`.

[81] Chris Finan, Anna Gaulton, Felix Kruger, Tom Lumbers, Tina Shah, Jorgen Engmann, Luana Galver, Ryan Kelly, Anneli Karlsson, Rita Santos, John Overington, Aroon Hingorani, and Juan Pablo Casas. The druggable genome and support for target identification and validation in drug development. July 2016. doi: 10.1101/066027. URL `http://biorxiv.org/lookup/doi/10.1101/066027`.

[82] Tudor I. Oprea, Cristian G. Bologa, Sren Brunak, Allen Campbell, Gregory N. Gan, Anna Gaulton, Shawn M. Gomez, Rajarshi Guha, Anne Hersey, Jayme Holmes, Ajit Jadhav, Lars Juhl Jensen, Gary L. Johnson, Anneli Karlson, Andrew R. Leach, Avi Ma'ayan, Anna Malovannaya, Subramani Mani, Stephen L. Mathias, Michael T. McManus, Terrence F. Meehan, Christian von Mering, Daniel Muthas, Dac-Trung Nguyen, John P. Overington, George Papadatos, Jun Qin, Christian Reich, Bryan L. Roth, Stephan C. Schrer, Anton Simeonov, Larry A. Sklar, Noel Southall, Susumu Tomita, Ilinca Tudose, Oleg Ursu, Duica Vidovi, Anna Waller, David Westergaard, Jeremy J. Yang, and Gergely Zahornszky-Khalmi. Unexplored therapeutic opportunities in the human genome. *Nature Reviews Drug Discovery*, 17(5):317–332, May 2018. ISSN 1474-1784. doi: 10.1038/nrd.2018.14. URL `http://www.nature.com/articles/nrd.2018.14`.

[83] Malachi Griffith, Obi L. Griffith, Adam C. Coffman, James V. Weible, Josh F. McMichael, Nicholas C. Spies, James Koval, Indraniel Das, Matthew B. Callaway, James M. Eldred, Christopher A. Miller, Janakiraman Subramanian, Ramaswamy Govindan, Runjun D. Kumar, Ron Bose, Li Ding, Jason R. Walker, David E. Larson, David J. Dooling, Scott M. Smith, Timothy J. Ley, Elaine R. Mardis, and Richard K. Wilson. DGIdb: mining the druggable genome. *Nature Methods*, 10 (12):1209–1210, December 2013. ISSN 1548-7105. doi: 10.1038/nmeth.2689. URL http://www.nature.com/articles/nmeth.2689.

[84] Alex H. Wagner, Adam C. Coffman, Benjamin J. Ainscough, Nicholas C. Spies, Zachary L. Skidmore, Katie M. Campbell, Kilannin Krysiak, Deng Pan, Joshua F. McMichael, James M. Eldred, Jason R. Walker, Richard K. Wilson, Elaine R. Mardis, Malachi Griffith, and Obi L. Griffith. DGIdb 2.0: mining clinically relevant druggene interactions. *Nucleic Acids Research*, 44(D1):D1036–D1044, January 2016. ISSN 0305-1048. doi: 10.1093/nar/gkv1165. URL http://academic.oup.com/nar/article/44/D1/D1036/2502659.

[85] Kelsy C. Cotto, Alex H. Wagner, Yang-Yang Feng, Susanna Kiwala, Adam C. Coffman, Gregory Spies, Alex Wollam, Nicholas C. Spies, Obi L. Griffith, and Malachi Griffith. DGIdb 3.0: a redesign and expansion of the druggene interaction database. *Nucleic Acids Research*, 46(D1):D1068–D1073, January 2018. ISSN 0305-1048. doi: 10.1093/nar/gkx1143. URL http://academic.oup.com/nar/article/46/D1/D1068/4634012.

[86] Target Product Profiles (TPP) DNDi, . URL https://www.dndi.org/diseases-projects/target-product-profiles/.

[87] Jeremy N. Burrows, Stephan Duparc, Winston E. Gutteridge, Rob Hooft van Huijsduijnen, Wiweka Kaszubska, Fiona Macintyre, Sbastien Mazzuri, Jrg J. Mhrle, and Timothy N. C. Wells. New developments in anti-malarial target candidate and

product profiles. *Malaria Journal*, 16(1):26, January 2017. ISSN 1475-2875. doi: 10.1186/s12936-016-1675-x. URL https://doi.org/10.1186/s12936-016-1675-x.

[88] Carl-Henrik Heldin. Targeting the PDGF signaling pathway in tumor treatment. *Cell Communication and Signaling : CCS*, 11:97, December 2013. ISSN 1478-811X. doi: 10.1186/1478-811X-11-97. URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3878225/.

[89] Eric C. McGary, Amir Onn, Lisa Mills, Amy Heimberger, Omar Eton, Gary W. Thomas, Mikhail Shtivelband, and Menashe Bar-Eli. Imatinib mesylate inhibits platelet-derived growth factor receptor phosphorylation of melanoma cells but does not affect tumorigenicity in vivo. *The Journal of Investigative Dermatology*, 122(2): 400–405, February 2004. ISSN 0022-202X. doi: 10.1046/j.0022-202X.2004.22231.x.

[90] Helen McKenna, Gijsbertus T. J. van der Horst, Irwin Reiss, and Daniel Martin. Clinical chronobiology: a timely consideration in critical care medicine. *Critical Care*, 22, May 2018. ISSN 1364-8535. doi: 10.1186/s13054-018-2041-x. URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5946479/.

[91] Zheng Chen, Seung-Hee Yoo, and Joseph S. Takahashi. Development and Therapeutic Potential of Small-Molecule Modulators of Circadian Systems. *Annual Review of Pharmacology and Toxicology*, 58(1):231–252, 2018. doi: 10.1146/annurev-pharmtox-010617-052645. URL https://doi.org/10.1146/annurev-pharmtox-010617-052645.

[92] Ryan P. Trump, Stefano Bresciani, Anthony W. J. Cooper, James P. Tellam, Justyna Wojno, John Blaikley, Lisa A. Orband-Miller, Jennifer A. Kashatus, Helen C. Dawson, Andrew Loudon, David Ray, Daniel Grant, Stuart N. Farrow, Timothy M. Willson, and Nicholas C. O. Tomkinson. Optimized Chemical Probes for REV-ERB. *Journal of medicinal chemistry*, 56(11):4729–4737, June 2013. ISSN 0022-2623. doi: 10.1021/jm400458q. URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4347663/.

[93] Sirus Palsson, Timothy P. Hickling, Erica L. Bradshaw-Pierce, Michael Zager, Karin Jooss, Peter J. OBrien, Mary E. Spilker, Bernhard O. Palsson, and Paolo Vicini. The development of a fully-integrated immune response model (FIRM) simulator of the immune response through integration of multiple subset models. *BMC Systems Biology*, 7(1):95, December 2013. ISSN 1752-0509. doi: 10.1186/1752-0509-7-95. URL `http://bmcsystbiol.biomedcentral.com/articles/10.1186/1752-0509-7-95`.

[94] Donald M Berwick, Thomas W Nolan, and John Whittington. The triple aim: care, health, and cost. *Health affairs*, 27(3):759–769, 2008.

[95] Leroy Hood, Rudi Balling, and Charles Auffray. Revolutionizing medicine in the 21st century through systems approaches. *Biotechnology journal*, 7(8):992–1001, 2012.

[96] Emily R Holzinger, Scott M Dudek, Alex T Frase, Ronald M Krauss, Marisa W Medina, and Marylyn D Ritchie. ATHENA: a tool for meta-dimensional analysis applied to genotypes and gene expression data to predict HDL cholesterol levels. In *Pacific Symposium on Biocomputing. Pacific Symposium on Biocomputing*, page 385. NIH Public Access, 2013.

[97] Emily R Holzinger, Scott M Dudek, Alex T Frase, Sarah A Pendergrass, and Marylyn D Ritchie. ATHENA: the analysis tool for heritable and environmental network associations. *Bioinformatics*, 30(5):698–705, 2014.

[98] Michael Sagner, Amy McNeil, Pekka Puska, Charles Auffray, Nathan D Price, Leroy Hood, Carl J Lavie, Ze-Guang Han, Zhu Chen, Samir Kumar Brahmachari, and others. The P4 Health SpectrumA Predictive, Preventive, Personalized and Participatory Continuum for Promoting Healthspan. *Progress in Cardiovascular Diseases*, 2016.

[99] John W Whittington, Kevin Nolan, Ninon Lewis, and Trissa Torres. Pursuing the triple aim: the first 7 years. *The Milbank Quarterly*, 93(2):263–300, 2015.

[100] Health Insurance: Premiums and Increases, . URL http://www.ncsl.org/research/health/health-insurance-premiums.aspx.

[101] Katherine Cumnock, Avni S. Gupta, Michelle Lissner, Victoria Chevee, Nicole M. Davis, and David S. Schneider. Host Energy Source Is Important for Disease Tolerance to Malaria. *Current Biology*, 28(10):1635–1642.e3, May 2018. ISSN 0960-9822. doi: 10.1016/j.cub.2018.04.009. URL http://www.sciencedirect.com/science/article/pii/S0960982218304378.

Appendix

A JSON Data Primitive Formats

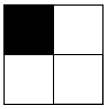


Figure A.1: JSON Image Example. This figure is coded by the JSON file in Listing A.1.

```
{
  "data_primitive": {
    "type": "image",
    "metadata": {
      "experiment": "example"
    },
    "data": {
      "header": {
        "transparency": true,
        "color_scheme": "RGBA",
        "default_color": [ 0, 0, 0, 0],
        "size": {
          "width": 2,
          "height": 2
          }
      },
      "pixels": [
        [ [ 255, 255, 255, 255], []
        ],
        [ [],[]
        ]
      ]
    }
  }
}
```

Listing A.1: Example of a small data primitive image file

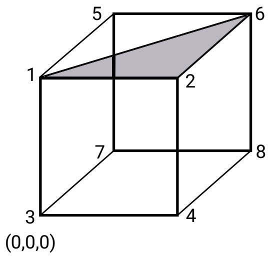


Figure A.2: Mesh Example. This is the figure coded by the JSON file in Listing A.2.

```
{
  "data_primitive": {
    "type": "mesh",
    "polygon_type": "triangle",
    "data":{
      "header": {
      "vertex_names": [ 1, 2, 3, 4, 5, 6, 7, 8 ]
      },
      "mesh": {
      "vertex": [
              [1,0,0,1], [2,1,0,1], [3,0,0,0], [4,1,0,0], [5,0,1,1], [6,1,1,1], [7,0,1,0], [8,1,1,0]
      ],
      "edges": [
              [1, 2], [2, 4], [4, 3], [3, 1], [5, 6], [6, 8], [8, 7], [7, 5], [1, 5], [2, 6], [3, 7], [4, 8]
      ],
      "polygons":[
              [1,2,6]
      ]
      }
    }
  }
}
```

Listing A.2: Example of a small data primitive mesh file

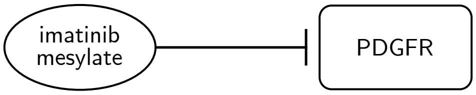


Figure A.3: Graph Example. This is the figure coded by the JSON file in Listing A.3.

```
{
  "data_primitive": {
    "type": "graph",
    "metadata": {
      "experiment": "example"
    },
    "data": {
      "header": {
        "terms": [ "id_vertex" ]
      },
      "vertex": [
        {
          "1": [ "imatinib mesylate" ]
        },
        {
          "2": [ "PDGFR" ]
        }
      ],
      "edges": [
        {
          "edge": [ 1 , 2 ],
          "source": "imatinib mesylate",
          "target": "PGDFR",
          "interaction_type": "inhibition"
        }
      ]
    }
  }
}
```

Listing A.3: Example of a small data primitive graph with two nodes and one edge

B Unified Modeling Language(UML) Reference Information

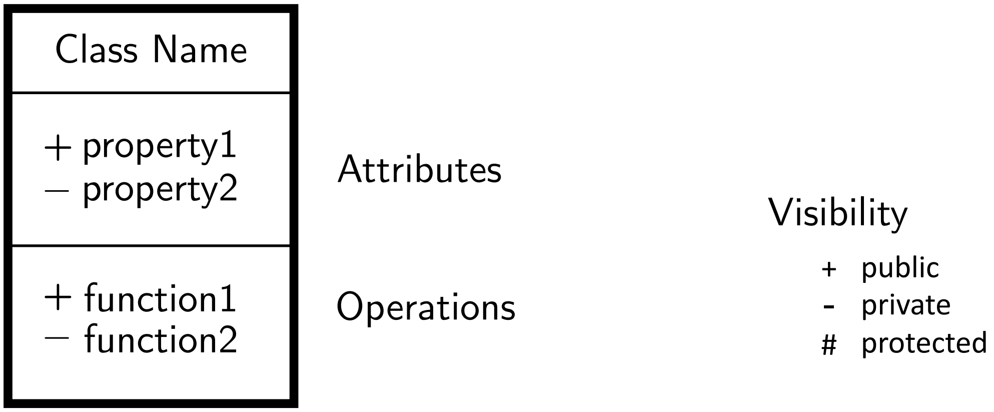
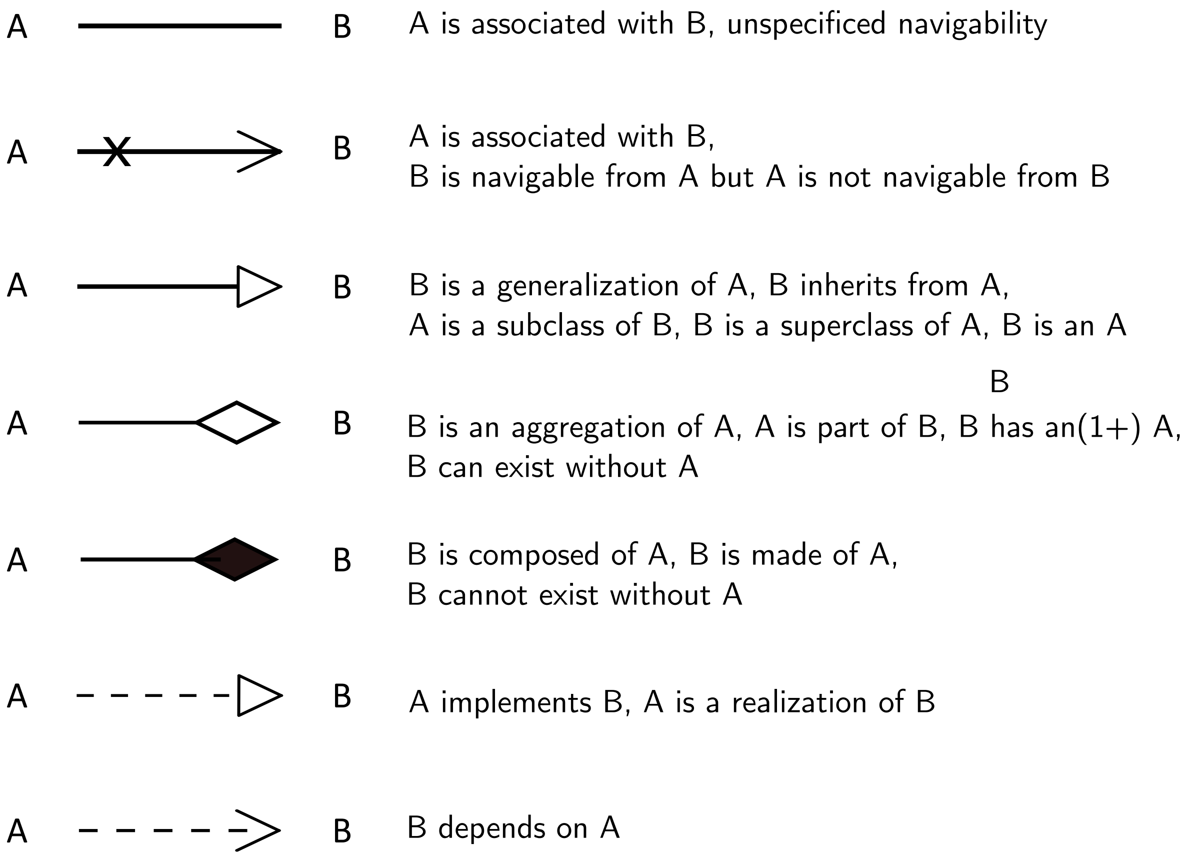


Figure B.1: Classes in UML are describes by attributes and methods, which are listed below the class in the diagram. The visibility is listed next to each property and operation.



A is associated with B, unspecificed navigability

A is associated with B,
B is navigable from A but A is not navigable from B

B is a generalization of A, B inherits from A,
A is a subclass of B, B is a superclass of A, B is an A

B

B is an aggregation of A, A is part of B, B has an(1+) A,
B can exist without A

B is composed of A, B is made of A,
B cannot exist without A

A implements B, A is a realization of B

B depends on A

Figure B.2: UML arrow Glossary. The different arrows and their various meanings are explained.

C SKED MATLAB Code

```
classdef clsSKEDTimeSeries < clsSKEDDPBase
    properties (SetAccess = public)
        Domain % JSONArray
        Range % JSONArray
        VarNames  % JSONArray
    end

    properties (SetAccess = private)
        l_Domain %cell array,Column Label, Each Column represent a distinct value of the independent variable
        l_Range %nxm Matrix, n Rows corresponds with Row Name(variable name), m Column Corresponds with m Time Points
        l_VarNames  %Row Label, Each Row Represent a Variable
    end

    methods
         %%% Purpose: SKEDTimeSeries object Constructor
         %%% Input: sFileName
         %%% Output: SKEDTimeSeries Object
         %%% Example: o = clsSKEDTimeSeries(sFileName);
         function obj = clsSKEDTimeSeries(sFileName)
             obj.Type = 'time_series';
             obj.Range = [];
             switch nargin
                case 0
                    error('A JSON file must be provided');
                case 1
                    obj.jo = obj.loadJSON(sFileName);
                    [obj.l_Domain, obj.Domain] = obj.getDomain();
                    [obj.l_Range, obj.Range ] = obj.getRange();
                    obj.l_VarNames = obj.convertJSONArray2collection('/data_primitive/data/header/term');
                    obj.VarNames = obj.getKey('/data_primitive/data/header/term');
                    obj.FileName = sFileName;
             end
         end
        %% ----------------------------------------
        function out = getTimePoints(obj)
            out = obj.getKey('/data_primitive/data/time_series');
            if strcmp(class(out), 'org.json.JSONObject')
               for i = 1:out.length
                   tempjo = LoadJSON(cFile{i});
                   out = getKey(sKey,tempjo);
                   if ~strcmp(out,'')
                       return
                   end
               end
            end
        end
        %% ----------------------------------------
        function [ out, l_out ] = getDomain(obj)
            import org.json.*
            out = {};
            data = obj.getKey('/data_primitive/data/time_series');
            for i = 1:data.length
                o = data.get(i-1);
                out{i} = o.get('time_stamp');
            end
            l_out = JSONArray(string(out));
            % set the obj.Domain = obj.getKey.('/data_primitive/data/time_series')
        end
        %% ----------------------------------------
        function [ out, l_out ] = getRange(obj)
            import org.json.*
            out = {};
            tempOut = {};
            data = obj.getKey('/data_primitive/data/time_series');
            for i = 1:data.length
                o = data.get(i-1);
                o = o.get('value');
                for j = 1:o.length
                    out{i,j} = str2num(o.get(j-1).get(0));
                end
                tempOut{i} = JSONArray(out(i,:));
            end
            l_out =  JSONArray(string(tempOut));
        end
    end
end
```

Listing C.1: clsSKEDTimeSeries

```
classdef clsSKEDGraph < clsSKEDDPBase
    properties (SetAccess = public)
        Verticies % JSONArray
        Edges % JSONArray
        VertexNames  % JSONArray
        VertexTerms  % JSONArray
        EdgeNames % JSONArray
        EdgeTerms % JSONArray
    end

    properties (SetAccess = private)
        l_verticies % cell array of vertex values(weights)
        l_edges % cell array describing edges; each edge connects two verticies
        l_vertexNames  % cell array of vertex names (row variables, ex gene names)
        l_vertexTerms  % cell array of vertex terms (column variables, ex mean)
        l_edgeNames % cell array of edge names
        l_edgeTerms % cell array of edge terms
    end

    methods
         %%% Purpose: SKEDGraph object Constructor
         %%% Input: none
         %%% Output: SKEDGraph Object (empty)
         %%% Example: o = clsSKEDGraph();
         function obj = clsSKEDGraph()
             import org.json.*
             obj.Type = 'graph';
             obj.Verticies = JSONArray();
             obj.Edges = JSONArray();
             obj.VertexNames = JSONArray();
             obj.VertexTerms = JSONArray();
             obj.EdgeNames = JSONArray();
             obj.EdgeTerms = JSONArray();
             obj.l_verticies = {};
             obj.l_edges = {};
             obj.l_vertexNames = {};
             obj.l_vertexTerms = {};
             obj.l_edgeNames = {};
         end

         function obj = setVerticies(obj, cVerticies)
             import org.json.*
             obj.Verticies = JSONArray(string(char(cVerticies)));
             obj.l_verticies = cVerticies;
         end

        function obj = setEdges(obj, cEdges)
             import org.json.*
             obj.Verticies = JSONArray(string(char(cEdges)));
             obj.l_verticies = cEdges;
        end

         function obj = setVertexNames(obj, cVertexNames)
             import org.json.*
             obj.VertexNames = JSONArray(string(char(cVertexNames)));
             obj.l_vertexNames = cVertexNames;
         end

         function obj = setEdgeNames(obj, cEdgeNames)
             import org.json.*
             obj.EdgeNames = JSONArray(string(char(cEdgeNames)));
             obj.l_EdgeNames = cEdgeNames;
         end

         function obj = setVerticiesFromTable(obj, tTable)
            import org.json.*
            cVertexTerms = tTable.Properties.VariableNames ;
            cVertexNames = tTable.Properties.RowNames;
            tTableValues = tTable{:,:} ;
            obj.Verticies = JSONArray(string(tTableValues));
            obj.VertexNames = JSONArray(string(cVertexNames));
            obj.VertexTerms = JSONArray(string(cVertexTerms));
            obj.l_verticies = tTableValues;
            obj.l_vertexNames = cVertexNames;
            obj.l_vertexTerms = cVertexTerms;
         end

         function obj = filterVerticies(obj, sVertexTerm, sQuantifier, nCutoff)
            import org.json.*

            % find idx of sVertexTerm
            idx = 0;
            for iVtxCtr = 1:size(obj.l_vertexTerms,2)
                if strcmp(obj.l_vertexTerms{iVtxCtr},sVertexTerm)
                    idx = iVtxCtr;
                end
            end

            cValues = obj.l_verticies(:,idx);
            switch sQuantifier
                case '<='
                    idx = cValues <= nCutoff;
                case '<'
```

```
                    idx = cValues < nCutoff;
                case '>='
                    idx = cValues >= nCutoff;
                case '>'
                    idx = cValues > nCutoff;
                case '=='
                    idx = cValues == nCutoff;
                case '~='
                    idx = cValues ~= nCutoff;
                otherwise
                    error("Please enter a valid MATLAB quantifier (ex. '<')")
            end

            obj.l_verticies = obj.l_verticies(idx,:);
            obj.l_vertexNames = obj.l_vertexNames(idx,:);
            obj.Verticies = JSONArray(string(obj.l_verticies));
            obj.VertexNames = JSONArray(string(obj.l_vertexNames));

         end

         function tTable = getTableFromVerticies(obj)
            tTable = cell2table([num2cell(obj.l_verticies)]);
            tTable.Properties.VariableNames = obj.l_vertexTerms;
            tTable.Properties.RowNames = obj.VertexNames;
         end

         function obj = setVerticiesFromList(obj, cArray)
            import org.json.*
            tTable =  cell2table(num2cell(zeros(length(cArray),2) )) ;
            tTable.Properties.RowNames = cArray;
            %tTable.Properties.VariableNames = { 'Varr'};
            obj = setVerticiesFromTable(obj, tTable);
         end


%          function tTable = getTableFromVerticies(obj)
%             tTable = cell2table([num2cell(obj.l_verticies)]);
%             tTable.Properties.VariableNames = obj.l_vertexTerms;
%             tTable.Properties.RowNames = obj.VertexNames;
%          end

    end

end

```

Listing C.2: clsSKEDGraph

```
classdef (Abstract) clsSKEDAnalysisBase < clsSKEDJSONBase
    %clsSKEDAnalysisBase Base class for analysis
    properties
        dataPrimitives % cell array of JSON data primitives
    end

    methods
        function loadDataPrimitives(obj,cFiles)
            import org.json.*
            obj.dataPrimitives = {};
            for i=1:length(cFiles)
                sFile = cFiles{i};
                tic
                disp(['Loading ' sFile])
                charData = fileread(sFile);
                strData = convertCharsToStrings(charData);
                jo = JSONObject(strData);
                try
                    obj.jo = jo;
                catch err
                    toc
                end
                Type = obj.getKey('/data_primitive/type');
                switch Type
                    case 'time_series'
                        o = clsSKEDTimeSeries(sFile);
                    case 'graph'
                        o = clsSKEDGraph(sFile);
                    case 'polygonal_mesh'
                        o = clsSKEDMesh(sFile);
                    case 'image'
                        o = clsSKEDImage(sFile);
                end
                obj.dataPrimitives{i} = o;
                toc
            end
        end
    end
end
```

Listing C.3: clsSKEDAnalysisBase

```
classdef clsSKEDBioinformatics < clsSKEDAnalysisBase
    %clsSKEDBioinformatics Common methods for bioinformatics

    properties
        % Generic properties
        Print % Boolean variable indicating whether results are visualized
        Experiment % Descriptor for set of values to be analyzed
        LowCountCutOff % Cut off point for RNAseq data
        SubjectTPDefinitions % Subject time point definitions
        TPstoAnalyze % List of time points to be compared ex. {'A','B' }
        NormalizationType % 'libSize' is only option now
        SKEDGraphContainer % Container object for graphs
        SKEDTSContainer % container object for time series
        KeyList % list of keys in container objects
        ValueList % list of values corresponding to keys
    end

    methods
        function obj = clsSKEDBioinformatics(cFiles)
            obj.loadDataPrimitives(cFiles);
            %Initialize properties
            obj.Print = false;
            obj.Experiment = '';
            obj.LowCountCutOff = 10;
            obj.SubjectTPDefinitions = {};
            obj.TPstoAnalyze = {};
            obj.NormalizationType = 'libSize';
            obj.KeyList = {};
            obj.ValueList = {};
            obj.SKEDGraphContainer = containers.Map;
            obj.SKEDTSContainer = containers.Map;

            %load DP into Container Objects
            for iDPCtr = 1: length(obj.dataPrimitives)
                sDPName = obj.dataPrimitives{iDPCtr}.getKey(
                 ↪  '/data_primitive/metadata/id');
                oMap = containers.Map({sDPName}, {obj.dataPrimitives{iDPCtr}} );
                switch obj.dataPrimitives{iDPCtr}.Type
                    case 'time_series'
                        obj.SKEDTSContainer = [obj.SKEDTSContainer; oMap];
                    case 'graph'
                        obj.SKEDGraphContainer = [obj.SKEDGraphContainer; oMap];
                    otherwise
                        error('Data primitive not implemented');
                end
            end
        end

        function findTSFromKeys(obj)
            obj.ValueList = values(obj.SKEDTSContainer,obj.KeyList);
        end

        function findGraphFromKeys(obj)
            obj.ValueList = values(obj.SKEDTSContainer,obj.KeyList);
        end

        function [oGraphReport, oGraphReportAllGenes, oGr_FC] = DESeq(obj)
            % This function finds DE genes and also records all the statistical testing for DE genes in two output tables.
            % Input: clsSKEDBioinformatics object
            % Output: MATLAB tables
            % Usage: [tReport_DE_E03_AP, tReport_E03_AP ] = DESeq(clsSKEDBioinformaticsE03);
            % Note: this script is based on the example given online at https://www.mathworks.com/help/bioinfo/examples/identifying-differentially-expressed-genes-from-rna-seq-data.html

            % NOTE: This function uses Group A as baseline and compares to
            % Group B. Ex. fold-change is B/A    (B over A)
            if strcmp(obj.NormalizationType,'libSize')
                [~, oGr_FC, tDataNorm, ~ ,cLabelsA, cLabelsB] = funLibSizeDataNormFC(obj);
            else
                error(' Please assign a valid string to NormalizationType.');
            end
            % assumes that only two types of timepoints are xeing compared
            mGroupAnorm = cell2mat( table2cell(tDataNorm(:,1:(size(cLabelsA,2)) )));
            mGroupBnorm = cell2mat( table2cell(tDataNorm(:,(size(cLabelsA,2))+1:end)));
            cGeneNames = tDataNorm.Properties.RowNames;

            sGroupADesc = obj.TPstoAnalyze{1}{2}; sGroupBDesc = obj.TPstoAnalyze{2}{2};

            normCountsmG = [mGroupAnorm, mGroupBnorm ];
            meanGA = mean(mGroupAnorm,2);
            meanGB = mean(mGroupBnorm,2);

            meanBase = (meanGA + meanGB) / 2;
            foldChange = meanGB ./ meanGA;
            log2FC = log2(foldChange);

            lowCountThreshold = obj.LowCountCutOff;
            lowCountGenesmG = all(normCountsmG < lowCountThreshold, 2);
            % The online MATLAB demo uses the unnormalized values
            %  tLocal = nbintest(mGroupB, mGroupA, 'VarianceLink',
             ↪  'LocalRegression');
            % The normalized values will be compared here.
            tLocal = nbintest( mGroupBnorm,mGroupAnorm,
             ↪  'VarianceLink','LocalRegression');
            if obj.Print == 1
                h = plotVarianceLink(tLocal,'compare',true);
                h(1).Title.String = ['Variance Link on ' sGroupBDesc ];
                h(2).Title.String = ['Variance Link on ' sGroupADesc ];
            end

            pValue = tLocal.pValue;
            [mFDR qValue] = mafdr(pValue);
            cFoldChange = mean(mGroupBnorm,2)./mean(mGroupAnorm,2);
            clog2FC = log2(cFoldChange);
            % create table with statistics about each gene
            geneTable = table(meanBase,meanGB,meanGA,cFoldChange,clog2FC);
            geneTable.Properties.RowNames = cGeneNames;
            geneTable.Properties.VariableNames = {'meanBase',
             ↪  'meanGroupB','meanGroupA','FoldChange','log2FC'};


            % create table with statistics about each gene
            geneTableWithoutLowCounts =
                ↪  table(meanBase(~lowCountGenesmG),meanGB(~lowCountGenesmG),
                ↪  meanGA(~lowCountGenesmG),cFoldChange(~lowCountGenesmG),
                ↪  clog2FC(~lowCountGenesmG));
            geneTableWithoutLowCounts.Properties.RowNames =
             ↪  cGeneNames(~lowCountGenesmG);
            geneTableWithoutLowCounts.Properties.VariableNames = { 'meanBase',
             ↪  'meanGroupB','meanGroupA','FoldChange','log2FC'};

            if obj.Print == 1
                summary(geneTable)

                mairplot(meanGB,meanGA,'Labels',cGeneNames,'Type','MA');
                set(get(gca,'Xlabel'),'String','mean of normalized counts')
                set(get(gca,'Ylabel'),'String','log2(fold change)')

                mairplot( meanGB(~lowCountGenesmG), meanGA(~lowCountGenesmG),
                 ↪  'Labels',cGeneNames(~lowCountGenesmG), 'Type','MA');
                set(get(gca,'Xlabel'),'String','mean of normalized counts')
                set(get(gca,'Ylabel'),'String','log2(fold change)')

                figure('units','normalized','outerposition',[0 0 1 1]);
                histogram(tLocal.pValue,100);
                title(['Histogram of P-Values' ]);
                xlabel('P-value')
                ylabel('Frequency')

                figure('units','normalized','outerposition',[0 0 1 1]);
                histogram(tLocal.pValue(~lowCountGenesmG),100)
                title(['Histogram of P Values for without low count genes']);
                xlabel('P-value')
                ylabel('Frequency')

                figure('units','normalized','outerposition',[0 0 1 1]);
                histogram(qValue(~lowCountGenesmG),100);
                title(['Histogram of Q Values for without low count genes']);
                xlabel('Q-value')
                ylabel('Frequency')

                figure('units','normalized','outerposition',[0 0 1 1]);
                nlog2MeanGA = log2(mean(mGroupAnorm,2));
                nlog2FoldChange = log2(cFoldChange);
                scatter(nlog2MeanGA(~lowCountGenesmG),
                 ↪  nlog2FoldChange(~lowCountGenesmG),
                 ↪  3,qValue(~lowCountGenesmG),'o')
                colormap(flipud(cool(256)))
                colorbar;
                ylabel('log2(Fold Change)')
                xlabel('log2(Mean of normalized counts)')
                title(['Fold change by FDR without low count genes between ',
                 ↪  sGroupADesc,' and ', sGroupBDesc ])
            end

            % Multiple Testing and Adjusted P-values
            % compute the adjusted P-values (BH correction)
            padj = mafdr(tLocal.pValue,'BHFDR',true);
            % add to the existing table
            geneTableWithoutLowCounts.pvalue = tLocal.pValue(~lowCountGenesmG);
            geneTableWithoutLowCounts.padj = padj(~lowCountGenesmG);

            % create a table with significant genes using low count cutoff
            sig = geneTableWithoutLowCounts.padj < 0.1;
            geneTableSigWithoutLowCounts = geneTableWithoutLowCounts(sig,:);
            geneTableSigWithoutLowCounts =
             ↪  sortrows(geneTableSigWithoutLowCounts,'padj');
            tReport = geneTableSigWithoutLowCounts;
            oGraphReport = clsSKEDGraph();
            oGraphReport = setVerticiesFromTable(oGraphReport, tReport);
        
            % create table with all genes
            geneTable.pvalue = tLocal.pValue;
            geneTable.padj = padj;
            tReport_AllGenes = geneTable;
            oGraphReportAllGenes = clsSKEDGraph();
            oGraphReportAllGenes = setVerticiesFromTable( oGraphReportAllGenes, tReport_AllGenes);
            % % create a table with significant genes to compare to MATLAB online
            % sig = geneTable.padj < 0.1;
            % geneTableSig = geneTable(sig,:);
            % geneTableSig = sortrows(geneTableSig,'padj');
            % tReport = geneTableSig;
            % numberSigGenes = size(geneTableSig,1);

            numberSigGenes = size(geneTableSigWithoutLowCounts,1);

            % find up-regulated genes
            up = geneTableSigWithoutLowCounts.log2FC > 1;
            upGenes = sortrows(geneTableSigWithoutLowCounts(up,:),'log2FC','descend');
            numberSigGenesUp = sum(up);
            % create table with significant genes to compare to MATLAB online
            % up = geneTableSig.log2FC > 1;
            % upGenes = sortrows(geneTableSig(up,:),'log2FC','descend');
            % numberSigGenesUp = sum(up);

            % find down-regulated genes
            down = geneTableSigWithoutLowCounts.log2FC < -1;
            downGenes = sortrows(geneTableSigWithoutLowCounts(down,:), 'log2FC','ascend');
            numberSigGenesDown = sum(down);
            % create table with significant genes to compare to MATLAB online
            % down = geneTableSig.log2FC < -1;
            % downGenes = sortrows(geneTableSig(down,:),'log2FC','ascend');
            % numberSigGenesDown = sum(down);

            % show table summary and figures
            if obj.Print == 1
                disp(['The number of significantly up regulated genes: ' num2str(numberSigGenesUp)])
                disp(['The number of significantly down regulated genes: ' num2str(numberSigGenesDown)])
                % display the top 10 up-regulated genes
                if size(upGenes,1) < 10
                    top10GenesUp = upGenes(:,:)
                else
```

```
                   top10GenesUp = upGenes(1:10,:)
               end

               % find top 10 down-regulated genes
               if size(downGenes,1) < 10
                   top10GenesDown = downGenes(:,:)
               else
                   top10GenesDown = downGenes(1:10,:)
               end

               figure
               scatter(log2(geneTableSigWithoutLowCounts.meanBase),
                 ↪  geneTableSigWithoutLowCounts.log2FC, 3,
                 ↪  geneTableSigWithoutLowCounts.padj,'o');
               colormap(flipud(cool(256)));
               colorbar;
               ylabel('log2(Fold Change)');
               xlabel('log2(Mean of normalized counts)');
               title(['Fold change by FDR between ', sGroupADesc,' and ',
                 ↪  sGroupBDesc,' in without low count genes' ]);
           end

       end % end function DE_Seq

       function [oGr_DataNorm, oGr_FC, tDataNorm, tFC, cLabelsA,cLabelsB] =
↪  funLibSizeDataNormFC(obj)
           % This function performs library size normalization and finds fold
↪  change.
           % Input: clsSKEDBioinformatics object
           % Output: MATLAB tables
           % Usage: [tDataNorm, tFC ]= DESeq(clsSKEDBioinformaticsE03);
           % Note: this script is based on the example given online at
             ↪  https://www.mathworks.com/help/bioinfo/examples/
             ↪  identifying-differentially-expressed-genes-from-rna-seq-data.html

           % NOTE: This function uses Group A as baseline and compares to
           % Group B. Ex. fold-change is B/A    (B over A)

           % define matricies mGroupA and mGroupB to hold values for the
             ↪  different
           % time point groups to be compared
           mGroupA = []; mGroupB = [];
           cLabelsA = {}; cLabelsB = {};
           sGroupADesc = obj.TPstoAnalyze{1}{2}; sGroupBDesc =
             ↪  obj.TPstoAnalyze{2}{2};
           % assign group values and labels using subject time point
           % definitions
```

```
for iTPDefCtr = 1: size(obj.SubjectTPDefinitions,1)
    sTPDefSubjectName = obj.SubjectTPDefinitions{iTPDefCtr}{1};
    %fprintf('TPDefSubjectName: %s\n', sTPDefSubjectName )
     ↪
    for iDPCtr = 1:size(obj.ValueList,2)
        sSubject = obj.ValueList{1,iDPCtr}.getMetaData('subject');
        if contains(sSubject,'pasilla')
            sExp = '';
        else
            sExp = obj.ValueList{1,iDPCtr}.getMetaData('experiment');
        end
        if strcmp(sTPDefSubjectName, sSubject)
            %fprintf('SubjectName: %s\n', sSubject)
            for iSubTPCtr =
             ↪  2:size(obj.SubjectTPDefinitions{iTPDefCtr},2)
                if strcmp( obj.SubjectTPDefinitions
                 ↪  {iTPDefCtr}{iSubTPCtr}{2}, obj.TPstoAnalyze{1}{1}
                 ↪  )
                    % assign values and labels for TPA
                    if strcmp(sExp,'E07B')
                        cTPAName = strcat( {'Mf'}, sSubject, {'_'},
                         ↪  sExp, {'_TP'},
                         ↪  num2str(obj.SubjectTPDefinitions
                         ↪  {iTPDefCtr}{iSubTPCtr}{1}), '_',
                         ↪  obj.SubjectTPDefinitions
                         ↪  {iTPDefCtr}{iSubTPCtr}{2});
                    else
                        cTPAName = strcat( sSubject, {'_'}, sExp,
                         ↪  {'_TP'}, num2str(
                         ↪  obj.SubjectTPDefinitions
                         ↪  {iTPDefCtr}{iSubTPCtr}{1}), '_',
                         ↪  obj.SubjectTPDefinitions
                         ↪  {iTPDefCtr}{iSubTPCtr}{2});
                    end
                    %cTPAName = strcat( sSubject, {'_'}, sExp,
                     ↪  {'_TP'}, num2str(obj.SubjectTPDefinitions
                     ↪  {iTPDefCtr}{iSubTPCtr}{1}), '_',
                     ↪  obj.SubjectTPDefinitions
                     ↪  {iTPDefCtr}{iSubTPCtr}{2});
                    cLabelsA = [cLabelsA cTPAName ];
                    %fprintf('\tsTPAName: %s\n', string(cTPAName))
                    cTPARange = obj.ValueList{ 1,
                     ↪  iDPCtr}.l_Range(obj.SubjectTPDefinitions
                     ↪  {iTPDefCtr}{iSubTPCtr}{1}, :);
                    mGroupA = [mGroupA cell2mat(cTPARange')];
                elseif strcmp(obj.SubjectTPDefinitions
                 ↪  {iTPDefCtr}{iSubTPCtr}{2}, obj.TPstoAnalyze{2}{1}
                 ↪  )
```

```
                            % assign values and labels for TPB
                            if strcmp(sExp,'E07B')
                                cTPBName = strcat( {'Mf'}, sSubject, {'_'},
                                 ↪  sExp, {'_TP'}, num2str(
                                 ↪  obj.SubjectTPDefinitions
                                 ↪  {iTPDefCtr}{iSubTPCtr}{1}), '_',
                                 ↪  obj.SubjectTPDefinitions
                                 ↪  {iTPDefCtr}{iSubTPCtr}{2} );
                            else
                                cTPBName = strcat( sSubject, {'_'}, sExp,
                                 ↪  {'_TP'} ,num2str(
                                 ↪  obj.SubjectTPDefinitions
                                 ↪  {iTPDefCtr}{iSubTPCtr}{1}),'_',
                                 ↪  obj.SubjectTPDefinitions
                                 ↪  {iTPDefCtr}{iSubTPCtr}{2} );
                            end
                            %cTPBName = strcat( sSubject, {'_'}, sExp,
                             ↪  {'_TP'}, num2str(obj.SubjectTPDefinitions
                             ↪  {iTPDefCtr}{iSubTPCtr}{1}),'_',
                             ↪  obj.SubjectTPDefinitions
                             ↪  {iTPDefCtr}{iSubTPCtr}{2});
                            cLabelsB = [cLabelsB cTPBName ];
                            %fprintf('\tsTPBName: %s\n', string(cTPBName))
                            cTPBRange = obj.ValueList {1,iDPCtr}.l_Range(
                             ↪  obj.SubjectTPDefinitions
                             ↪  {iTPDefCtr}{iSubTPCtr}{1}, :);
                            mGroupB = [mGroupB cell2mat(cTPBRange')];
                        end
                    end % for loop over
                end
            end
        end % end for lood over SubjectTPDefinitions
        %%%%%%%%%%%% library size normalization
        mG = [mGroupA mGroupB];
        %normalize using  data from both groups
        pseudoRefSample = geomean(mG,2);
        nz = pseudoRefSample > 0;
        ratios = bsxfun(@rdivide,mG(nz,:),pseudoRefSample(nz));
        sizeFactors = median(ratios,1);
        normCountsmG = bsxfun(@rdivide,mG,sizeFactors);

        mGroupAnorm = normCountsmG(:,1:( size(cLabelsA,2) ) );
        mGroupBnorm = normCountsmG(:,(size(cLabelsA,2))+1:end);

        meanGA = mean(mGroupAnorm,2);
        meanGB = mean(mGroupBnorm,2);

```

```
            meanBase = (meanGA + meanGB) / 2;
            foldChange = meanGB ./ meanGA;
            log2FC = log2(foldChange);

            cGeneNames = regexprep(obj.ValueList{1,1}.l_VarNames,'^x_', '');
            cGeneNames = regexprep(cGeneNames,'_$', '');

            % save normalized data in a table
            tDataNorm = cell2table([num2cell(mGroupAnorm),
             ↪  num2cell(mGroupBnorm)]);
            tDataNorm.Properties.VariableNames = [cLabelsA,cLabelsB];
            tDataNorm.Properties.RowNames = cGeneNames;

            oGr_DataNorm = clsSKEDGraph();
            oGr_DataNorm = setVerticiesFromTable(oGr_DataNorm, tDataNorm);

            % find fold change relative to TP A
            foldChange = [];

            for iGrpBCtr = 1:size(mGroupBnorm,2)
                cTPBValues = mGroupBnorm(:,iGrpBCtr);
                sTPBLabel = cLabelsB{iGrpBCtr};
                sTPBName  = regexp(sTPBLabel,
                 ↪  '^[a-zA-Z0-9]*','match','forceCellOutput' );
                idx = contains(cLabelsA,sTPBName{1}{1});
                foldChange = [ foldChange cTPBValues ./ mGroupAnorm(:,idx) ] ;
            end

            tFC = cell2table(num2cell(foldChange));
            tFC.Properties.VariableNames = cLabelsB;
            tFC.Properties.RowNames = cGeneNames;

            oGr_FC = clsSKEDGraph();
            oGr_FC = setVerticiesFromTable(oGr_FC, tFC);

            if obj.Print == 1
                figure('units','normalized','outerposition',[0 0 1 1]);
                subplot(2,1,1)
                maboxplot(log2(mG), 'orientation','horizontal','BoxPlot',
                 ↪  {'Labels',[ cLabelsA cLabelsB]  } )
                title( ['Raw read count for all subjects between ', sGroupADesc,'
                 ↪  and ', sGroupBDesc ])
                ylabel('Time Points')
                xlabel('Log2(counts)')

                subplot(2,1,2)
             maboxplot(log2(normCountsmG),'title',['Normalized read count for all subjects between ', sGroupADesc,' and ', sGroupBDesc ],'orientation','horizontal','BoxPlot', {'Labels',[ cLabelsA cLabelsB]  })
             ylabel('Time points')
             xlabel('Log2(counts)')
           end
        end % funlibSizeDataNormFC

        function oGr = funLibSizeNormAllTP(obj)
          cLabels = {}; mGroup = [];
          for iTPDefCtr = 1: length(obj.SubjectTPDefinitions)
             sTPDefSubjectName = obj.SubjectTPDefinitions{iTPDefCtr}{1};
             %fprintf('TPDefSubjectName: %s\n', sTPDefSubjectName )
             for iDPCtr = 1:length(obj.ValueList)
               sSubject = obj.ValueList{iDPCtr}.getMetaData('subject');
               try
                 sExp = obj.ValueList{iDPCtr}.getMetaData('experiment');
               catch
                 sExp = '';
               end
               if strcmp(sTPDefSubjectName, sSubject)
                 %fprintf('SubjectName: %s\n', sSubject)
                 for iSubTPCtr = 2:length(obj.SubjectTPDefinitions{iTPDefCtr})
                   % assign values and labels for TPA
                   cTPName = strcat( sSubject, {'_'}, sExp, {'_TP'}, num2str( obj.SubjectTPDefinitions{iTPDefCtr}{iSubTPCtr}{1}), '_', obj.SubjectTPDefinitions{iTPDefCtr}{iSubTPCtr}{2});
                   cLabels = [cLabels cTPName ];
                   cTPRange = obj.ValueList {1,iDPCtr}.l_Range( obj.SubjectTPDefinitions{iTPDefCtr}{iSubTPCtr}{1}, :);
                   mGroup = [mGroup cell2mat(cTPRange')];
                 end
               end
             end
          end

           pseudoRefSample = geomean(mGroup,2);
           nz = pseudoRefSample > 0;
           ratios = bsxfun(@rdivide,mGroup(nz,:),pseudoRefSample(nz));
           sizeFactors = median(ratios,1);
           normCountsmG = bsxfun(@rdivide,mGroup,sizeFactors);
```

```
            cGeneNames = obj.ValueList{1,1}.l_VarNames;
            % save normalized data in a table
            tDataNorm = cell2table(num2cell(normCountsmG));
            tDataNorm.Properties.VariableNames = cLabels;
            tDataNorm.Properties.RowNames = cGeneNames;
            % transform normalized data into graph data primitive
            oGr = clsSKEDGraph();
            oGr.setVerticiesFromTable(tDataNorm);
       end % end function
    end
end
```

Listing C.4: clsSKEDBioinformatics

```
classdef  (Abstract) clsSKEDDPBase  < clsSKEDJSONBase
    %clsSKEDDPBase Abstract class with function that returns JSON Object
    properties (Access = public)
        FileName % location of the JSON file
        Type % Type of data primitive (e.g. 'TimeSeries','Graph')
    end

    methods (Access = public)
        %% ----------------------------------------
        function obj = clsSKEDDPBase(obj)
            %
        end
    end
end
```

Listing C.5: clsSKEDDPBase