Addressing Provenance issues in Big Data Genome Wide Association Studies (GWAS)

David Lauzon, Beatriz Kanzki, Victor Dupuy, Alain April*
École de Technologie Supérieure (ÉTS)
1100, rue Notre-Dame Ouest,
Montréal, QC, Canada
davidonlaptop@gmail.com
beatriz.kanzki@gmail.com
vdcd120491@gmail.com
alain.april@etsmtl.ca

Michael S. Phillips, Johanne Tremblay*, Pavel Hamet*
Centre de Recherche du Centre Hospitalier de l'Université de Montréal (CRCHUM)
900, rue St-Denis
Montréal, QC, Canada
pgxdoc@gmail.com
johanne.tremblay@umontreal.ca
pavel.hamet@umontreal.ca

* corresponding authors

***Abstract*—Effective genome wide association studies (GWAS) present new Big Data challenges for health researchers: data processing delays, data provenance and efficient real-time visualization. This paper presents two recent open source initiatives that, used together, aim at solving these issues. First, an introduction to GWAS is presented followed by a description of the issues faced by the bioinformatics staff at this small health research lab. We then introduce two open source project we initiated: a query engine (QnGene) and a genetic output analysis tool (GOAT) to address these issues and give an overview of their internal architecture and our current experimentation and validation plan.**

Keywords—GWAS health systems provenance; Big Data; dbSNP discrepancies; GWAS visualization; open-source.

I. Background and Problem

Genome wide association studies (GWAS) have recently evolved into a powerful tool for investigating the genetic association to human diseases and are very popular among health scientists. This type of study searches the whole genome for small variations, called single nucleotide polymorphisms (SNPs), that occur more frequently in patients with a particular disease than in the general population that does not have the disease [1]. Results generated from GWAS involve millions of SNPs per phenotype which the researchers usually store in a relational database (for example MySQL) for annotation, drill down analysis and results quality investigations. In Dr. Pavel Hamet's diabetes research lab, at the CHUM located in Montreal, limits of the use of relational database technology was starting to hinder research. For example, retrieving data for one GWAS associated with one phenotype (i.e. which is around 6 000 000 SNPs) can take up to 5 minutes for just 4 criteria (i.e. rsId, p-value, position, and chromosome), and is up to 50 minutes if the researcher would like to extract 10 GWAS in order to compare the different phenotypes concurrently. So it looks like there is a linear relationship between the size of the data set being used and the MySQL query response time [2]. In that context, investigating this data is becoming time consuming, limit certain analysis and is preventing the research team from finding potential hidden relationships between phenotypes as they have such slow turnaround time.

The provenance of the GWAS results is another issue reported. GWAS results originates from a complex sequence of data transformations that have generated many intermediate files during the discovery process. Hence we can compare this discovery process to a graph of trial and errors activities, where a fraction of its paths could lead to significant findings that can be further investigated. As the number of steps, in the discovery process increases, the number of branching possibilities increase as well exponentially. Some branches may be deleted or ingnored, but others may become significant and investigated at a later time. The consequence of using this type of discovery process is that the workflow, from raw data to GWAS results, generates many larges intermediate files, and the management of those files and results, to ensure the repeatability of the GWAS is quickly becoming a major challenge for the bioinformatics staff of Dr. Hamet's lab. If the intermediate files, of a published experiment are deleted, then the provenance of the data is lost and it may be difficult or even impossible to reproduce the published result when time comes.

A third issue, also related to provenance, is that our scientific comprehension of the human genome is constantly evolving and some facts we take for granted may prove to be wrong in the future. The dbSNP database [3], which is used as the genetic variant reference for most GWAS experiments, is reported to contain erroneous data that could introduce significant errors in GWAS interpretations. Some authors [4], [5] have reported that the dbSNP database may contain up to 15–17% of false positives. dbSNP contains a "unique" RS number for each variant in its database, but the variant(s) associated with the RS number may change over different dbSNP build number releases. A variant can be splitted in multiple variants, merged with another variant, and/or moved

to a different position on a genome as our scientific comprehension of the genome improves [6]. Therefore it is recommended that authors also include the dbSNP build number release along with the RS number they are publishing. However, most GWAS software, such as plink [7] and snptest [8], do not store this information, which make it difficult for researchers to identify the provenance of the RS number they are publishing in their research.

In this paper we present an overview of two open-source projects we initiated to meet the challenges of data provenance, horizontal scalability and visualization of GWAS in a Big Data context :

- QnGene: A scalable provenance-aware implementation of an initial GWAS algorithm;
- GOAT: A responsive user interface for mining GWAS results.

II. CURRENT SOLUTIONS AND SHORTFALLS

A. Genomics in Big Data

Most GWAS softwares, for example plink [7], use a single process or even a single thread of processing on computers to process the data. This leads to slow computing times. A typical solution, to improve the scalability of this type of software architecture, is to split the data into multiple files (e.g. 1 file per chromosome), invoke the software once for each file, and merge the results back at the end of all concurrent processing. This workaround solution is performed either manually, using a custom shell script, or using a workflow engine like Taverna [9]. While this approach works to an extent, it complexifies the user-software interactions and generates additional intermediate files that will need to be preserved to ensure the provenance and repeatability of experiments.

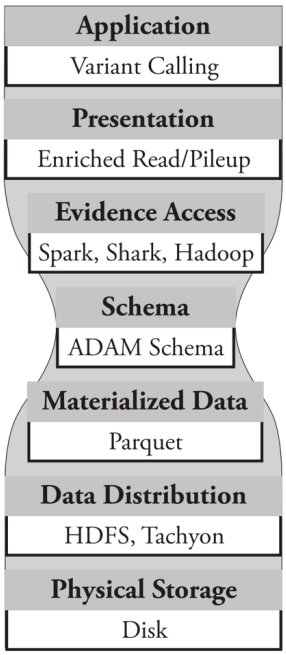

Fig. 1. Hourglass layered architecture of the ADAM variant calling framework developed at UC Berkeley[10].

A better approach would be to tackle the scalability problem at its core by creating a file schema that can be automatically be distributed on a cluster of computers. The ADAM open-source project, at U.C. Berkeley [10], has been able to achieve a 28x speedup over existing genetic variant calling solutions using this solution. To sustain the evolution of the project and provide a solid scalable core, they suggest using a 7-layer architecture (see Figure 1) that leverage existing Big Data technologies such as Hadoop[11] and Spark[12] to allow easy concurrent processing. Another advantage of building new algorithms based on ADAM is that it minimize the amount of code (and programming time which leads to lower efforts/costs) to implement new algorithms that can scale easily on such a distributed system. However, ADAM was designed for whole genome sequencing and therefore currently lacks the data structure required to perform GWAS at our lab. Section III of this paper will present how this is being adressed currently by our software engineering research project.

B. Provenance for Big Data health research applications

Some provenance solutions have been published but fall short of what is needed here. For example, the Provenance Aware Storage System (PASS) [13] is a file system approach to provenance. The principle of PASS is that instead of writing the data to the file system directly, the application writes to the PASS system which in turns records the provenance and stores the data in its underlying file system. PASS has been designed and optimized for storing data in the Amazon S3 file system which can be unpractical (because of storage cost) for long-term storing of terabytes and even petabytes of data required in the field of genomics research. Furthermore, PASS is currently not compatible with the Hadoop Distributed File System (HDFS) which is already an abstraction on top of a regular file systems. If PASS would be used to trace the file operations in HDFS, it would currently not be able to view the file names directly but only the file's blocks id (e.g. BP-1976835025-172.17.0.55-1440087701986).

RAMP [14] is one of the first attempt at bringing provenance natively in the Hadoop ecosystem. RAMP provides wrapper classes for the Map and Reduce functions. These wrapper classes record the provenance. The drawbacks are that 1) their benchmark results report a 16% to 76% range of performance overhead; 2) RAMP was designed to work only for older versions of Hadoop MapReduce (i.e. RAMP code has not been updated since nearly 5 years); and 3) Apache Spark is becoming more popular than Hadoop MapReduce which is beign phased out in recent months.

Recently, the Titian project [15] added provenance support natively in Apache Spark. Titian claims that its overhead "rarely exceeds 30% above the baseline job execution time". However, as Titian does not persist the provenance metadata after a job has completed, it seems to be useful mostly for debugging a job's code and not adressing repeatability of experiments in a real health lab experimental setting.

Finally, we looked at Cloudera's Navigator [16] which is a commercial product that integrates well with the Hadoop

ecosystem. Navigator transparently records the provenance and has features for visual data lineage and audits. Overall great product if you can afford it. Navigator however is not optimized for doing the provenance of genome wide association studies.

Based on what is available, we decided to undertake the design of a solution that is divided in two open-source initiatives as some users may only want one of the two functions for their lab. The next section describes the proposed solution for adressing provenance issues in Big Data Genome Wide Association Studies (GWAS).

III. Architecture Overview

Our initial objective, with this first version of QnGene, is to solve the scalability issues of our GWAS open source project named: Genetic Output Analytic Tool (GOAT). The source code and vision document for the first version of GOAT is available on GitHub (https://github.com) by searching for: GOAT Genetic Ouput Analysis Tool.

GOAT's first release quickly suffered from performance issue where the main query to the database was not very responsive. We tried to use the Bokeh Server [22] to solve this problem, but this query was still taking several minutes to execute. We identified that most of the query process duration was spent in the relational database processing (in our case MySQL). After detailed analysis of the query, we found out that it only needed a small subset of the columns. However, traditional relational databases (RDBMS), like MySQL, are optimized for business transaction processing where the whole record is transferred. As a result of the RDBMS's internal design, the whole row (i.e. all the columns of a row) needed to be read from disk even when only a subset of columns was needed for the GWAS manhattan graph display.

Therefore, we proposed that this specific query would benefit from techniques used by analytical data stores (OLAP). OLAP systems store each column individually: a row is split across multiple files, or segment of a file. This means that if a query need only 1 column from a 10-columns table, it only needs to load 10% of the data from disk. The next version of GOAT, will use the open source Parquet columnar file format [23], which is one of the most popular format in use by Big Data systems today.

This open source project is based on Berkeley's variant calling architecture (refer to Fig. 1). QnGene will use a similar approach for GWAS (see Fig. 2). QnGene will use a REST web service interface (i.e. HTTP) so that clients, such as GOAT and other open-source visualization tools, can use it to query the GWAS results. This is an important layer because if a visualization tool, like GOAT, would have interacted directly with Spark, it would have been more to difficult to record its provenance efficiently. QnGene is using the Apache Parquet technology to distribute the data horizontally, on a cluster of nodes, and Apache Spark is used to scale the data processing.

Finally, we also extended the ADAM format to incorporate the data structures required to store GWAS results. We plan on contributing this extension back to the ADAM project once the QnGene code base is more stable and the project is released to the general public.

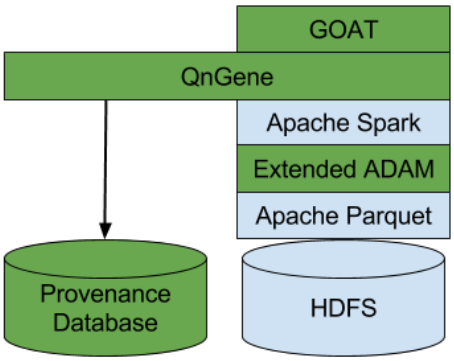

Fig. 2. Architecture of our GWAS components. Artefacts in green represents our contributions to the field.

IV. Recording the Provenance

While the focus of previous approaches to provenance in Big Data systems have been aiming at debugging the system itself [14], [13], [15], our objective is to record the provenance with the purpose of actual reproducibility and third party auditing for small health research labs. However, this approach can also be used along side other provenance software such as Titian [15] if needed.

From the perspective of GOAT, QnGene acts as presentation wrapper around Spark. We expect that recording the provenance this way would generate less than 10% overhead over the computation of GWAS studies since we only need to capture the provenance at the beginning and end of the transaction. This will be experimented later this year (see Figure 4).

Another advantage, over current proposals, is that now that the provenance metadata is available in a database, it becomes queryable by the users. While this is not a complete provenance solution yet, we believe it is a good first step towards meeting the regulations of the FDA CFR 21 Part 11 [17] imposed on all health research labs.

A. Improving the provenance of published variants

To get around dbSNP limitation described in section I, we've modified the ADAM data model to store the dbSNP build number along with every RS number in the database. When the genotypes are imported in QnGene, the user specify the dbSNP release that generated the genotypes. This information is then carried over automatically when performing a GWAS with QnGene.

Currently, QnGene version 0.1 will provide built in support to perform GWAS with linear and logistic regression models only at first. We plan adding support for more algorithms in the future, as needed by our team and/or as the open source community will contribute to the project.

For GWAS results generated with other softwares, such as plink[7], the user will also be able to import the results in QnGene and then benefit from the dbSNP build number provenance fonctionality.

V. GENETIC OUTPUT ANALYSIS TOOL (GOAT)

GOAT is an other open source project we designed for GWAS real-time visualisation of manhattan graphs used in GWAS. It is a web based GWAS mining visualisation tool developed with Python programming language and using the Django framework [18]. We are working on the second release of GOAT presently. The following text explains the internals of this software. GOAT uses currently a MySQL relational database containing all GWAS data that is stored in Apache Parquet format in order to ensure compressed and efficient processing based on a columnar data storage technology developed for Big Data applications. In this second release, the database will interact with a Spark back-end where queries from the Django web application of GOAT will be processed so that response time for each query can be decreased, improving time response for queries.

Finally the Django web application will interact with GOAT's user interface by providing HTML, CSS, Javascript files, and data to user according to query and also process user's queries through the Django application. In this second release, the MySQL database will be only kept for defining a user table containing login info, and personal settings. In order to better allow open source contributions to GOAT's future development, the architecture was improved to be as modular as possible, from a high level perspective to a low level implementation.

Therefore, the second release of GOAT will be composed of four distinct layers. A summary of GOAT's new modular architecture can be seen in Figure 3.

First, at the top of Fig. 3, we find a columnar store database (i.e. in Parquet format), that will contains all the GWAS results. Second, a Spark back-end will query the database and send the data to the third layer, which is composed of a Django based Web application. This Web application will send the user HTML, CSS and JavaScript files, relay its queries to the Spark back-end and send back data, in JSON format, to the front-end application. The front-end, located at the bottom of Fig. 3, is GOAT's last layer, and consist of a Single Page Application (SPA) built with Facebook's framework React.js, on top of a Flux architecture.

React.js technology allows to create reusable components for the front-end. This is extremely useful in order to allow for quick design of new functionalities, and to avoid repeating a lot of logic, like user input validation. An SPA is like a state-machine: the state of the application determines what is displayed to the user. In order to create a new feature, a developer only has to add one additional state, which avoids to interfere and break the previous ones. The Django Web application main role is to pass queries from the front-end to the Spark back-end, as well as distributing the necessary files to the user in order to display using the SPA. Each feature is contained in its own module. To add a new feature, a developer only needs to create a new "Django App" (i.e. which is basically a folder), and import it where it is needed. A "Tools" module has also been dedicated to the re-usable tools: like passing the data to the JSON format, or passing a query to the Spark back-end.

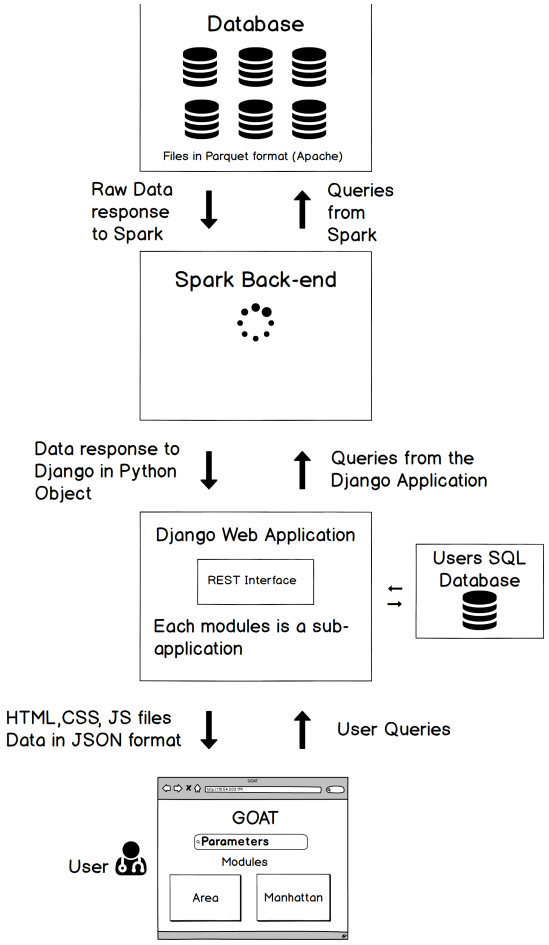

Fig. 3. Modular view of GOAT's new architecture.

Finally, the Spark back-end will contain pre-configured queries, which will be used to fetch data in the database when the Django Web application will specifically call them.

VI. EARLY EXPERIMENTATION OF QNGENE

We are at very early validation stage at the time of writing this paper. In Figure 4, we present the experimental methodology that we are following to experiment QnGene provenance solution for Big Data Genome Wide Association Studies. At the top of the figure we find the a priori model (Provenance model v1) that has been presented and discussed with U.C. Berkeley's ADAM main contributors for feedback during our last visit. Since this meeting, we are now working on a second iteration (Provenance model V2) as described in this paper. This improved model is tightly integrated with the visualization software named GOAT. We will validate this second provenance model using a case study at CHUM lab, where we will compare the benefits of our approach with the current situation (i.e. without a workflow system) and with two other solutions (i.e. Biogenix and Galaxy). This will help us better understand the strenght/weaknesses of our proposed solution.

For the third version, we aim to develop a provenance measurement model that can produce a global score of the provenance quality of a GWAS pipeline.

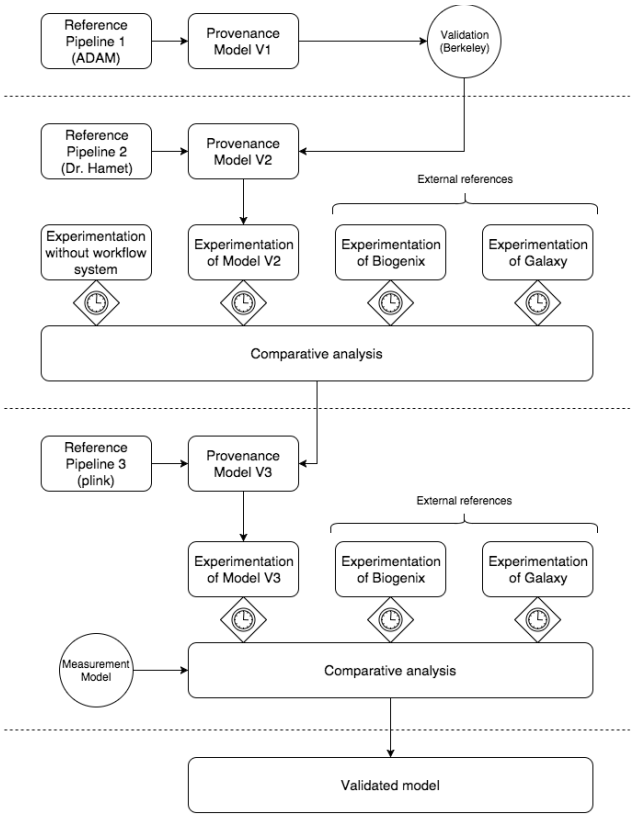

Fig. 4. Proposed experimental methodology for QnGene

Further experiments are planned during 2016-17 and we will report on our progress.

VII. CONCLUSION

In this paper, we have described the approach we are currently working on to improve the scalability and provenance of GWAS. We have also discussed the growing scalability challenge by showing how Apache Spark and Parquet can be used to improve GWAS provenance by showing how it is used in two GWAS open source mining tools: Genetic Output Analysis Tool (GOAT) and QnGene. While QnGene supports currently a limited subset of GWAS algorithms that tracks the dbSNP build number automatically, we have described how it was possible to import the GWAS results from other software. We hope that this novel approach to GWAS provenance and scalability will provide a reference model for all small health research centers and this is the reason to open source it.

ACKNOWLEDGMENT

We would like to thank Michael Phillips for his constant support and Frank Nothaft and Matt Massie of U.C Berkeley's AMPLAB for their comments and advice on provenance for ADAM.

REFERENCES

[1] W. S. Bush and J. H. Moore, "Genome-wide association studies," PLoS Comput Biol, vol. 8, no. 12, p. e1002822, 2012.

[2] R. Hofstede, A. Sperotto, T. Fioreze, and A. Pras, "The network data handling war: Mysql vs. nfdump," in Networked Services and Applications-Engineering, Control and Management. Springer, 2010, pp. 167–176.

[3] S. T. Sherry, M.-H. Ward, M. Kholodov, J. Baker, L. Phan, E. M. Smigielski, and K. Sirotkin, "dbsnp: the ncbi database of genetic variation," Nucleic acids research, vol. 29, no. 1, pp. 308–311, 2001.

[4] A. A. Mitchell, M. E. Zwick, A. Chakravarti, and D. J. Cutler, "Discrep-ancies in dbSNP confirmation rates and allele frequency distributions from varying genotyping error rates and patterns," Bioinformatics, vol. 20, no. 7, pp. 1022–1032, 2004.

[5] L. Musumeci, J. W. Arthur, F. S. Cheung, A. Hoque, S. Lippman, and J. K. Reichardt, "Single nucleotide differences (SNDs) in the dbSNP database may lead to errors in genotyping and haplotyping studies," Human mutation, vol. 31, no. 1, pp. 67–73, 2010.

[6] A. Kitts, L. Phan, M. Ward, and J. B. Holmes, "The Database of Short Genetic Variation (dbSNP) 2013 Jun 30 [Updated 2014 Apr 3]," 2014. [Online]. Available: http://www.ncbi.nlm.nih.gov/books/NBK174586/

[7] S. Purcell, B. Neale, K. Todd-Brown, L. Thomas, M. A. Ferreira, D. Bender, J. Maller, P. Sklar, P. I. De Bakker, M. J. Daly et al., "Plink: a tool set for whole-genome association and population-based linkage analyses," The American Journal of Human Genetics, vol. 81, no. 3, pp. 559–575, 2007.

[8] J. Marchini and B. Howie, "Genotype imputation for genome-wide association studies," Nature Reviews Genetics, vol. 11, no. 7, pp. 499– 511, 2010.

[9] K. Wolstencroft, R. Haines, D. Fellows, A. Williams, D. Withers, S. Owen, S. Soiland-Reyes, I. Dunlop, A. Nenadic, P. Fisher, J. Bhagat, K. Belhajjame, F. Bacall, A. Hardisty, A. Nieva de la Hidalga, M. P. Balcazar Vargas, S. Sufi, and C. Goble, "The Taverna workflow suite: designing and executing workflows of Web Services on the desktop, web or in the cloud." Nucleic acids research, vol. 41, no. Web Server issue, pp. 557–561, 2013.

[10] F. A. Nothaft, M. Massie, T. Danford, Z. Zhang, U. Laserson, C. Yeksigian, J. Kottalam, A. Ahuja, J. Hammerbacher, M. Linderman, M. J. Franklin, A. D. Joseph, and D. A. Patterson, "Rethinking Data-Intensive Science Using Scalable Analytics Systems Categories and Subject Descriptors," Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, 2015.

[11] D. Cutting and M. Cafarella. (2016) Apache hadoop: an open-source software for reliable, scalable, distributed computing. [Online]. Available: https://hadoop.apache.org

[12] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauley, M. J. Franklin, S. Shenker, and I. Stoica, "Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing," NSDI'12 Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation, pp. 2–2, 2012.

[13] K.-K. Muniswamy-Reddy, P. Macko, and M. Seltzer, "PASS: Provenance for the Cloud," Proceedings of the 8th USENIX Conference on File and Storage Technologies, pp. 14–15, 2010.

[14] R. Ikeda, H. Park, and J. Widom, "Provenance for generalized map and reduce workflows," Proceedings of the Fifth CIDR, pp. 273–283, 2011.

[15] M. Interlandi, K. Shah, S. D. Tetali, M. A. Gulzar, S. Yoo, M. Kim, T. Millstein, and T. Condie, "Titian: Data Provenance Support in Spark," Proceedings of the VLDB Endowment, vol. 9, no. 3, pp. 216–227, 2015.

[16] Cloudera. (2015) Cloudera navigator: Big data meets data governance. [Online]. Available: https://cloudera.com/products/cloudera-navigator.html

[17] FDA, "Title 21: Food and Drugs, Chapter I: Food and Drug Administra-tion, Department of Health and Human Services, Subchapter A:General, Part 11: Electronic Records; Electronic Signatures," Tech. Rep., 2016.

[18] D. Moore, R. Budd, and W. Wright, Professional Python Frameworks: Web 2.0 Programming with Django and Turbogears. John Wiley & Sons, 2008