Harvest: an open platform for developing web-based biomedical data discovery and reporting applications

Jeffrey W Pennington,[1] Byron Ruth,[1] Michael J Italia,[1] Jeffrey Miller,[1] Stacey Wrazien,[1] Jennifer G Loutrel,[1] E Bryan Crenshaw,[2,3] Peter S White[1,4,5]

▶ Additional material is published online only. To view please visit the journal online (http://dx.doi.org/10.1136/amiajnl-2013-001825).

[1]Center for Biomedical Informatics, The Children's Hospital of Philadelphia, Philadelphia, Pennsylvania, USA
[2]Center for Childhood Communication, The Children's Hospital of Philadelphia, Philadelphia, Pennsylvania, USA
[3]Department of Otorhinolaryngology: Head and Neck Surgery, Perelman School of Medicine at the University of Pennsylvania, Philadelphia, Pennsylvania, USA
[4]Division of Oncology, The Children's Hospital of Philadelphia, Philadelphia, Pennsylvania, USA
[5]Department of Pediatrics, Perelman School of Medicine at the University of Pennsylvania, Philadelphia, Pennsylvania, USA

Correspondence to
Peter S White, Children's Hospital of Philadelphia, 34th St and Civic Center Blvd, Rm 1024 CHOP North, Philadelphia, PA 19104-4318, USA; whitep@email.chop.edu

JWP, BR and MJI contributed equally to this work.

Received 20 March 2013
Revised 4 September 2013
Accepted 28 September 2013
Published Online First 16 October 2013

To cite: Pennington JW, Ruth B, Italia MJ, *et al*. *J Am Med Inform Assoc* 2014;**21**:379–383.

ABSTRACT

Biomedical researchers share a common challenge of making complex data understandable and accessible as they seek inherent relationships between attributes in disparate data types. Data discovery in this context is limited by a lack of query systems that efficiently show relationships between individual variables, but without the need to navigate underlying data models. We have addressed this need by developing *Harvest*, an open-source framework of modular components, and using it for the rapid development and deployment of custom data discovery software applications. *Harvest* incorporates visualizations of highly dimensional data in a web-based interface that promotes rapid exploration and export of any type of biomedical information, without exposing researchers to underlying data models. We evaluated *Harvest* with two cases: clinical data from pediatric cardiology and demonstration data from the OpenMRS project. *Harvest's* architecture and public open-source code offer a set of rapid application development tools to build data discovery applications for domain-specific biomedical data repositories. All resources, including the OpenMRS demonstration, can be found at http://harvest.research.chop.edu

INTRODUCTION

Biomedical researchers are often challenged with navigating the large volumes of data available from medical and research information systems.[1] Datasets useful to biomedical research are typically complex, highly dimensional, and temporal, often with significant variation in granularity, sparsity, and representation across data dimensions.[2] Research data complexity is amplified by the high volume of data points generated by modern molecular and imaging platforms.[3 4] Unlike purely transactional data, such as those typically derived from business operations, these data are not readily summed or averaged, limiting the utility of traditional business intelligence tools in this context. Moreover, existing query and reporting tools tend to be general-purpose instruments with user interfaces designed to support expert analysts working in a variety of situations.[5] As a result, researchers without access to sophisticated informatics expertise are increasingly challenged with efficiently managing, exploring, and understanding the information at their disposal.

Accordingly, we developed *Harvest*, a new biomedical data application framework. Our primary development objectives were to (1) provide for researchers with limited informatics ability a toolkit to generate meaningful views of raw data according to their domain expertise and their specific interests; (2) dynamically query key aspects of a dataset based on the inherent characteristics of individual data attributes; (3) combine single attribute queries into multiattribute set operation queries; and (4) provide an actionable endpoint by exporting immediately available raw data in an analysis-ready format. To demonstrate its effectiveness, *Harvest* was used to develop and deploy intuitive data discovery applications for two distinctive biomedical domains: pediatric cardiology diagnostic modality and procedure data generated at The Children's Hospital of Philadelphia (CHOP), and infectious disease data published by the OpenMRS open-source electronic health record (EHR) project.

BACKGROUND AND SIGNIFICANCE

Adoption of EHR systems by academic medical centers has created significant potential for the re-use of clinical data for research.[6] Trends in translational research indicate a need for data discovery platforms that can stage and disseminate data in a readily accessible form to researchers focusing on disease. Such trends include the increasing adoption and diversity of EHRs to capture longitudinal patient information; rapid development and early adoption of genomics, imaging, and other complex data types; the progressive organization of multidisciplinary teams focusing on systems biology research; and the need for integration and exchange of many different types and complexities of data. We sought a solution that would enable rapid iteration of design ideas and provide the flexibility to adapt to the accelerated pace of innovation in diverse research settings.

Many available analytic applications operate on arbitrary datasets, including SAP Business Objects, IBM Cognos, QlikTech QlikView, and TIBCO SpotFire. Many of these tools are specific for business intelligence (BI) and perform well on quantitative transactional data aggregated across multiple dimensions. However, commercial BI tools have not yet been widely adopted in the research community, owing in part to the limitations of these tools for analyzing highly multidimensional data arising from discrete observations. For instance, aggregation of observations by counting is of little value to researchers when the data are categorical, and of marginal use for non-epidemiological patient-oriented research when the data are quantitative. Both QlikView and SpotFire allow for these features but are limited by their architectures, which rely on in-memory data stores that present performance barriers for *Big Data* applications. Moreover, the cost of licensing these tools for open-access internal and external academic use can be prohibitive.

The variability of research data from project to project requires either a monolithic application data model—which optimizes efficiency from application development and operational standpoints—or custom code developed for every application. One strategy for a monolithic database schema is to use an entity-attribute-value (EAV) model. An EAV model uses key-value fields in a general-purpose table to extensibly store arbitrary data without having to create dedicated tables, as exemplified in biomedicine by the i2b2 (informatics for integrating biology and the bedside) project.[7] An EAV approach provides the convenience of a fixed database model, but at the expense of the benefits provided by normalized relational models such as database-level referential integrity and performance optimization through indexing.[8] While successful in many domains, EAV models have difficulty supporting ad hoc, attribute-centric queries on highly dimensional data,[9] such as clinical and annotated genomic data.

Alternatively, custom code built on an application-specific database model makes sense when the application requirements are sufficiently unique for a generic solution to be impracticable. Our experience with numerous collaborators indicated commonality in many functional requirements among projects—namely, a capacity to iteratively browse, search, query, review, and export project-specific data, such that a certain level of generic functionality (and code) was clearly feasible. However, requirements differed in the data model, where we were faced with a wide variety of data types among projects. Based on the success of the BI tools discussed above, we reasoned that an application data access layer might solve the problem of generic access to highly variable database schemas. We hypothesized that a hybrid solution might handle an arbitrary schema with minimal to no custom code in the data access layer, while simultaneously supporting rapid development of data discovery applications customized to the domain of interest. Accordingly, we developed the *Harvest* framework both for our own use and dissemination to others.

MATERIALS, METHODS, AND TECHNOLOGIES

Harvest core components and implementation

Harvest comprises three main components: a data abstraction layer (*Avocado*), a web API (*Serrano*), and a web client (*Cilantro*). *Avocado* and *Serrano* are implemented using Python[10] and the Django web application framework.[11] *Cilantro* is implemented using JavaScript. Detailed documentation of each component is available online.[12]

Avocado

The core of *Harvest* is a data abstraction component termed '*Avocado*'. *Avocado* extends the Django object-relational mapper[13] to provide a stable and contextual application programming interface (API) for client data access. *Harvest* applications rely on *Avocado* to generate and manage application metadata, especially automatically generated data profiles that reveal and publish inherent characteristics of the raw data, such as determination of categorical versus continuous type, and indexing of text data for subsequent search. *Avocado* also supports additional metadata management, including providing an alias for data fields with human-readable concept names, a searchable thesaurus of synonym keywords describing concepts, and domain-specific categories for grouping concepts within a client user interface. *Avocado* currently authorizes access to both data rows and concepts through optional integration with Django Guardian.[14]

Serrano server

Harvest's Serrano server publishes a hypermedia API that enables web clients to consume data encapsulated by *Avocado* and saves user data for *Harvest* applications. This is accomplished by storing a representation of user actions into a history table in the application database. This supports several key features, including the ability for users to name and save queries, and provides an auditable record of user actions.

Cilantro client

The *Cilantro* JavaScript client generates and displays intuitive data visualizations such as histograms, bar charts, and pie charts for suitable database fields in real time using HighCharts JS.[15] Importantly, this feature allows users to see a summary profile of data even before constructing formal queries. In some instances, users interact directly with the graphical displays to construct their queries—for example, by clicking on a bar of interest in a graph that represents a subpopulation (figure 1A). The architecture of the client also allows for specialized query controls to be integrated when simple data-driven displays are not appropriate. For example, a diagnosis field might be presented as a hierarchical view that allows a user to browse, search, and select one or more diagnoses of interest while constructing complex set operation queries (figure 1B).

Technologies

Harvest implements a three-tiered application architecture using a relational database management system, web application server, and web browser client. All server components are built using Python and the Django web framework. *Harvest* configurations to date have used the open-source PostgreSQL[16] and SQLite[17] databases, and the Nginx[18] and Apache[19] web servers. However, *Harvest* is readily compatible with other database and web server technologies.

RESULTS

Harvest deployments

Harvest has been used to develop several biomedical data discovery applications spanning a variety of diseases and data types, two of which are described here. Typically, specific applications are built on patient-oriented databases encompassing unique, domain-specific relational schemas. Each application also employs a unique configuration of *Avocado* metadata that provides researchers with a user interface incorporating data heuristics, terminology, and an organization scheme specific to the area of study.

Harvest instance: CardioDB

CardioDB (figure 1A) is a quality improvement data resource for measuring outcomes in the Cardiac Center at CHOP, using clinical data derived primarily from cardiac catheterization and echocardiogram systems. The database and application includes 323 fields for 47 300 patients across 24 900 catheterization procedures and 54 000 echocardiogram procedures, refreshed nightly. Data types include problem list, diagnosis vocabulary, procedure vocabulary, radiation exposure, cardiac dimensions, and cardiac output (see online supplementary figure S1). Diagnosis and procedure vocabularies are implemented in a hierarchical data model populated with standard ICD9 and Current Procedural Terminology codes along with CHOP-customized codes. The query interface to these vocabularies is implemented in a generic *Cilantro* module called the *Harvest Vocab Browser*.[20] CardioDB enables discovery and reporting of clinical outcomes by providing clinicians and researchers with a longitudinal view of discrete, granular measures of patient cardiac function both before and after a catheterization procedure, together with relevant data describing both the patient and procedures. Sensitive measurement of outcomes is supported

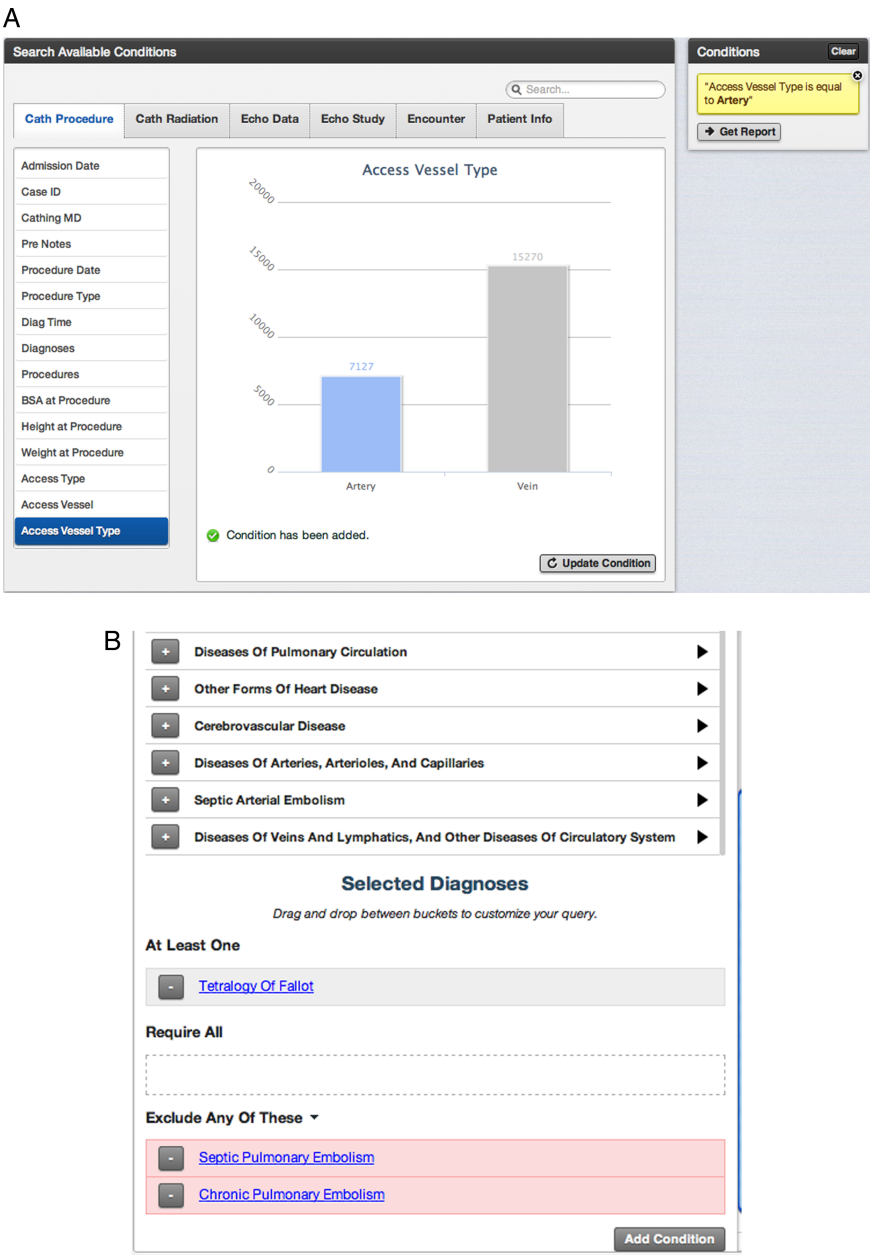

Figure 1 (A) Categorical data by default are displayed as bar or pie charts. Users click on chart elements of interest to select and add to the list of query conditions. (B) Custom controls may be developed to handle complex data types and query operations, such as this vocabulary browser that displays ICD9 diagnoses in a browseable and searchable hierarchy, together with input fields that enable element drag-and-drop supported construction of complex set-operation query conditions. Views displayed originate from the CardioDB *Harvest* application.

by use of *Harvest's* query interface to stratify CHOP's highly variable cardiac patient population. CardioDB is used by researchers and clinicians at CHOP with interest in abnormal cardiac anatomy and physiology.

Harvest instance: OpenMRS

We validated *Harvest* with a public dataset originating outside of CHOP, for which we have no specific interest or motivation, by developing a demonstration application (figure 2) using a deidentified clinical dataset[21] published by the OpenMRS open source EHR project.[22] OpenMRS data types include infection status, disease management, and clinical laboratory results (see online supplementary figure S2). The application includes 56 data fields on 5300 patients and is freely available online through the *Harvest* website[23] for direct exploration, and as an installable package. The extract, transform, and load (ETL) package for the *Harvest* OpenMRS application is also available at the *Harvest* website.

In the course of building these applications, we have found that *Harvest* separates design, coding, and implementation from the definition and organization of data elements by domain experts. We have also found that the framework enables rapid prototype development, as exemplified by the creation of the OpenMRS demonstration in just two days, excluding database creation and ETL. The *Cilantro* user interface and *Avocado* metadata promote intuitive and meaningful review that enables productive feedback and iterative development loops.

DISCUSSION

Our results indicate that *Harvest* facilitates construction of accessible biomedical data discovery applications by providing informatics researchers with open, standards-based components, an adaptable framework for defining domain-specific data concepts, and a user interface design that makes large and complex datasets accessible. We have used *Harvest* to support data discovery applications of highly multidimensional data across a variety of clinical and research domains. The two applications

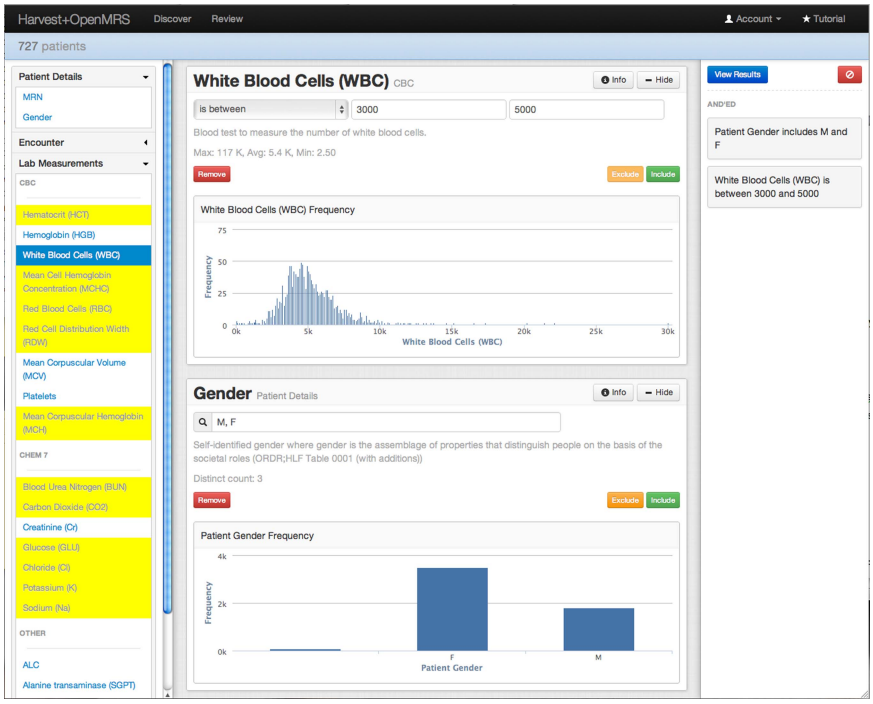

Figure 2 The query construction view is used to preview data, such as the distribution of white blood cell count, while building up query conditions that are displayed in a readable format. View originates from the OpenMRS *Harvest* application.

we describe signify generalizability as they share a common patient-encounter root data model but otherwise diverge in their detailed dimensions. Specifically, the CardioDB data model invokes procedural diagnostic, intervention, and cardiac function data, whereas the OpenMRS data model focuses on infection status, disease management, and clinical laboratory results. *Harvest* can support a variety of data discovery paradigms, including hypothesis generation and testing, clinical outcomes reporting, and multisite access to shared research data.

Frameworks such as *Harvest* provide researchers and data reporting groups with a toolkit suitable for addressing a primary need of biomedical research: the ability to be rapidly and effectively immersed in interoperable data relevant to a study of interest, and in a way in which the data are readily comprehensible. In our experience, researchers have gained trust with specific *Harvest* applications, in part because the immediate presentation of data visualizations provides transparency of data content. Data are presented as collected, with outliers and potential data inconsistencies clearly observable even before a researcher is required to execute a query. Often, these visualizations are the first time researchers have seen variability and potential quality problems in their data. This incentivizes users to participate in iterative improvement, thus generating process investment and subsequent increased scientific value from the data.

A decided strength of *Harvest* is the ease of reuse in a variety of biomedical domains. For example, we are currently developing *Harvest* instances to support medical imaging studies, multisite integration of clinical and molecular data, and genomic variant sets derived from next generation sequencing. These applications can completely reuse the *Avocado* and *Serrano* components, which greatly accelerates the development of a new client interface.

All *Harvest* deployments thus far have resulted in rapid adoption by biomedical researchers, requiring little user interface training. The domain-specific conceptual and visual representation of data enabled by *Harvest* seems to lower the cognitive burden on researchers to learn and understand an application's underlying data model before data discovery. This contrasts with our experience with commercial BI tools, which typically require an expert user to navigate both underlying data models and the extensive functionality implemented by these enterprise tools.

Harvest provides basic authorization capabilities, enabling controlled access both to data rows and concepts/fields at the user or user group level. Further development of this authorization capability to increase usability and transparency of the authorization scheme is warranted. A means to authorize access to application functionality, such as data export or drill-to-detail, is a promising area for development.

Importantly, *Harvest* itself does not deal with the need to transform and integrate biomedical data from original source systems and formats into well-structured data suitable for query and analysis. We address this need through various ETL methods and tools. We have found that the rapid application development capability of *Harvest* accelerates the ETL development process, especially when using the *Avocado* data API to evaluate staged and partially integrated data.

To date, *Harvest* has been primarily used in focused, project-specific applications, where the analysis and configuration required to take advantage of *Avocado*'s conceptual representation of data were feasible and justified by the expected end use of the application. We have yet to test the utility of *Harvest* in a more typical enterprise data warehouse scenario, where a potentially large number of data fields might impose challenges of scalability and required effort. Furthermore, while *Harvest* has been able to model and present highly complex data types, we have not yet fully dealt with the more difficult task of modeling longitudinal data in a manner that would enable construction of temporal query constraints.

CONCLUSION

Harvest promotes immediacy in data exploration and use in data-intensive clinical and translational science. For biomedical researchers, *Harvest*-based applications provide an accessible, easy-to-understand view of complex data, presented in a conceptual framework that they help develop, and with focus on data content rather than application development. The *Harvest* platform

and the OpenMRS demonstration application are available as open source under the unrestricted BSD license agreement. The demonstration application, system documentation tutorials, and ETL package may be found at http://harvest.research.chop.edu.

Acknowledgements We thank Mark Porter for critical reading of the manuscript.

Contributors JWP oversaw project operations, study design, and manuscript authoring; BR, MJI, JM, SW, and JGL contributed substantially to platform development and evaluation; EBC provided domain expertise and testing; PSW oversaw project development and provided overall project oversight. All authors read and approved the final manuscript.

Funding This work was supported in part by NIH grant DC012207 (EBC) and the David Lawrence Altschuler Endowed Chair Fund (PSW).

Competing interests None.

Provenance and peer review Not commissioned; externally peer reviewed.

Data sharing statement The *Harvest* platform and the OpenMRS demonstration application are available as open source under the unrestricted BSD license agreement. The demonstration application, system documentation tutorials, and extract, transform, and load (ETL) package may be found at http://harvest.research.chop.edu

REFERENCES

1 Fox P, Hendler J. Changing the equation on scientific data visualization. *Science* 2011;331:705–8.
2 D'Avolio LW, Farwell WR, Fiore LD. Comparative effectiveness research and medical informatics. *Am J Med* 2010;123(12 Suppl 1):e32–7.
3 Aldhous P. Managing the genome data deluge. *Science* 1993;262:502–3.
4 The Data Deluge. *An e-Science Perspective*. Wiley Online Library, 2003.
5 Thomsen C, Pedersen TB. A survey of open source tools for business intelligence. *Lect Notes Comput Sc* 2005;3589:74–84.
6 Safran C, Bloomrosen M, Hammond WE, *et al*. Toward a national framework for the secondary use of health data: an American Medical Informatics Association White Paper. *J Am Med Inform Assoc* 2006;14:1–9.
7 Murphy SN, Weber G, Mendis M, *et al*. Serving the enterprise and beyond with informatics for integrating biology and the bedside (i2b2). *J Am Med Inform Assoc* 2010;17:124–30.
8 Karwin B. *Entity-Attribute-Value*. *SQL Antipatterns: Avoiding the Pitfalls of Database Programming*. Raleigh, NC: Pragmatic Bookshelf, 2010.
9 Nadkarni PM, Brandt C. Data extraction and ad hoc query of an entity—Attribute—Value database. *J Am Med Inform Assoc* 1998;5:511–27.
10 Python Programming Language—Official Website. Secondary Python Programming Language—Official Website 2013. http://www.python.org
11 Django Secondary Django 2013. http://www.djangoproject.com/
12 Avocado—Metadata APIs for Django. Secondary Avocado—Metadata APIs for Django 2013. http://cbmi.github.com/avocado/
13 Django documentation—Models and databases. Secondary Django documentation—Models and databases 2013. http://docs.djangoproject.com/en/dev/topics/db
14 Django Guardian. Secondary Django Guardian 2013. http://pythonhosted.org/django-guardian/
15 Serrano—Hypermedia API for Avocado. 2013. http://github.com/cbmi/serrano
16 PostgreSQL. Secondary PostgreSQL 2013. http://www.postgresql.org/
17 SQLite. Secondary SQLite 2013. http://www.sqlite.org/
18 NGINX Web Server. Secondary NGINX Web Server 2013. http://wiki.nginx.org/Main
19 Apache HTTP Server Project. Secondary Apache HTTP Server Project 2013. http://httpd.apache.org/
20 Harvest Vocab Browser. Secondary Harvest Vocab Browser 2013. https://github.com/cbmi/harvest-vocab
21 OpenMRS Demo Data. Secondary OpenMRS Demo Data 2013. https://wiki.openmrs.org/display/RES/Demo+Data
22 Wolfe BA, Mamlin BW, Biondich PG, *et al*. The OpenMRS system: collaborating toward an open source EMR for developing countries. *AMIA Annu Symp Proc* 2006:1146. doi:86273 [pii][published Online First: Epub Date]
23 Harvest Website. Secondary Harvest Website 2013. http://harvest.research.chop.edu