PROPOSAL

DataBrew

BOHEMIA data management plan

Proposal

Executive summary:

The stage-gate period of the Bohema project will need an integrated and coordinated approach to building a data management system with the ability to handle all aspects of the "data lifecycle", from collection to analysis. The values of simplicity, reproducibility, security, costs, and real-time analyzability across sites are all required. Tools must be engineered and deployed in a top-down manner, but at the same time need to be flexible and dynamic so as to collect and incorporate input from partners in a bottom-up fashion. We propose to achieve this balance through a hub-and-spoke organizational approach with iterative development practices. This document describes DataBrew's proposal for building a data management system for the data lifecycle.

For the Bohemia-project, data needs to be (i) collected, (ii) validated, (iii) stored, (iv) accessed, (v) visualized, (vi) analyzed, and (vii) disseminated. DataBrew intends to manage the entire data lifecycle using custom-built, open-source software tools so that the project managers and researchers can focus on the public health aspects of the project, and not on its technological aspects.

I. Data collection:

The Bohemia project requires tailored data collection (survey) applications to be run on tablets and phones.

The data collection application:

Data will be collected through a custom-made Android application, running the open source OpenDataKit format, designed to run on both tablets and mobile phones. The application, will also have a browser-based component for use through a standard desktop.

Data entry:

As per the requirements of the project, we propose 3 modes of data entry (listed in order of frequency):

1. A tablet or telephone is used to run the ODK android application to follow a questionnaire with explicit survey flow and validation protocol (see “Validation” section).
2. A local site manager (spoke) uses the browser-based application to enter individual data points or batch entries.
3. A centralized database manager (hub) manually makes ad-hoc (but auditable) modifications at the request of a site.

If required, translation/ingestion scripts will be written to port data from currently existing databases to the centralized database. However, this approach is not recommended, since it would mean not benefitting from the validation and control checks implemented by DataBrew (see “Validation” section).

Language issues:

We will engineer a "translations tables" approach so as to expose the application's front-end in the language of the user, while at the same time ensuring that the back-end is standardized in English. The translation tables can be used bi-directionally: that is, to ensure that the end user is able to operate in her language, but also to translate variable and category levels from the back-end language (English) back to a local language for analysis. Upon upload to the centralized server, a merge-and-replace with the respective language table will translate all questions and answers to English. Original-language responses will be stored separately. The centralized database will be in English, but can be translated to any local language (even when data collection was in English) through a replace-and-merge operation. All language issues will be managed by the centralized database manager (CDM). DataBrew will build the tools, but will require input from site partners for correct translations.

On-device storage and upload:

As per project requirements, data entry will not rely on internet connectivity. Rather, entries are stored locally on the data-entry device as "packets". When the device achieves a wireless connection with a pre-authenticated network, the packets are sent to the centralized server. Upon full data upload, a confirmation is sent from the server to the device, at which time the already uploaded data is removed from the device. For more details on storage, see "Storage and security".

II. Validation:

Control flow:

The "Bohemia" application will have explicitly engineered survey control flow, to be specified in collaboration with consortium partners. Questions contingent on previous responses will be shown only when applicable.

Individual validation checks:

The "Bohemia" application will employ three levels of validation: (i) confirmatory, (ii) prohibitory, and (iii) iterative.

- Confirmatory validation consists of requiring the person entering data to explicitly confirm an entry if deemed to be unlikely. For example, when a person under the age of 18 has a "number of children" variable of greater than 0 (unlikely, but certainly not impossible), an additional "confirmatory" check is performed to ensure that the entry was not erroneous.
- Prohibitory validation consists of explicitly forbidding certain entries, or requiring the approval (via a unique password) of a manager. For example, a person previously entered in the database as "male" cannot have greater than 0 pregnancies.
- Iterative validation consists of allowing for prohibited responses *only* if the prohibition condition is removed. For example, to enter a new malaria case for on June 1st for a person previously entered as having died on May 1st is allowed, but requires a modification of previously entered data (ie, the person either did not die or died later). Another example: if the tablet device is detected via GPS to be at a certain location, but data entry is being performed for someone who has previously been registered as residing elsewhere, a modification of the person's residence is required for further entry. This "iterative" data entry process is tracked at each iteration, so previous states can be restored by the CDM in case of error.

Aggregate validation checks:

Certain data entry events are expected at the individual level, but deemed improbable when in high frequency. For example, it may not be uncommon to have a female diagnosed with malaria. However, the consecutive diagnosis of 10 females and no males suggests potential data entry error. These checks will be scripted to be run daily at the aggregate (ie, site) level, rather than in the data collection device. When statistically improbable occurrences arise, the site manager will automatically be prompted to confirm or correct. These checks will be built a priori, but documentation will also be created so as to enable the creation of additional aggregate checks as project needs arise.

III. Storage and security:

Storage:

A centralized PostgreSQL database will be employed via a AWS RDS (Amazon Web Services Relational Database Service). Backups (via SQL "dumps") will be generated nightly via a cron script running on an EC2 (Elastic Compute Cloud) instance, and stored on a remote server via S3 (Simple Storage Service) as well as a local back-up server. Data tables will consist of one table per survey form, in addition to translation tables.

All sites will have access to their own data via AWS, and will be encouraged to regularly generate backup data-dumps for storage on site in whichever CRM is employed by each site.

Security:

Local site managers will be given read-only SSH access to the database, and will first undergo a training regarding data security and confidentiality. If additional levels of security are required, appropriate security groups will be created by the CDM. The CDM will connect to the database using private-public keypair authentication.

IV. Access:

The three AWS services employed by DataBrew (RDS, EC2, and S3) are industry standards and are used by some of the largest technology companies. They each come with "out-of-the-box" state of the art security measures, such as encryption via SSL, IP whitelisting, etc., while also allowing for custom security protocols. Access to any of the three services (the databases themselves, the virtual machine running the back-up scripts, or the back-up data dumps) will require multi-factor security authentication. Only authorized study managers will be given access, and data access and use will be monitored to ensure compliance with study protocol, ethical norms, and legal restrictions.

For authorized users, there will be two main points of access: (i) via CLI (command line interface) tools for tech-savvy users (such as the S3's RESTful API or the psql command utility for RDS) and (ii) a browser layer which allows for basic querying via a custom web application built by DataBrew. To avoid SQL injection or unauthorized access, this web application will be relatively straightforward, allowing the user to pick from a myriad of drop-down options so as to ensure successful and coherent data entry on-site, as well as to monitor data entry clerks.

V. Visualization (real-time dashboards):

Dynamic reporting:

By consolidating all data into a centralized location, dashboards and dynamic reports can be written prior to data actually being collected, and updated in real time. As per project requirements, two types of dynamic reporting are envisioned: (i) a process-oriented dashboard and (ii) a content-oriented dashboard.

A process-oriented dashboard is an up-to-date collection of charts, tables, and tests which detail the current status of data collection. For example, a process-oriented dashboard may show the number of certain data uploads by site location, number of manual corrections, time since last upload. It is most useful in ensuring that all data collection processes are going according to plan.

A content-oriented dashboard is an up-to-date reflection of the *content* of the data collected. For example, it might show a chart with real-time cumulative malaria incidence broken down by geography, etc.. It is most useful in monitoring results in real time so as to begin the processes of analysis and inference, *before* the final dataset is assembled.

VI. Study design and analysis:

During the study:

DataBrew will build several "scripts" in Rmarkdown, iPython Notebook, Shiny, or Flask which automatically run certain content-oriented analyses which go beyond simple counts. The intended use of these scripts is two-fold:

1. Study design. Scripts can be engineered for the estimation of certain pre-operational statistical procedures (ie, power calculations) which rely on potentially varying assumptions/parameters. By scripting these analyses, a change in assumptions can quickly be incorporated into the analysis and the relevant figures recalculated.
2. Analysis. These will be built a priori, so that as the databases are populated, real-time analyses can be automatically generated. For example, a researcher may need to know the most up-to-date estimated population percentage for a certain characteristic, or an odds ratio for a certain variable as a function of certain risk factors. This can be built into a dynamic report which both (a) runs with the most recent data and (b) allows for interactivity (such as the inclusion or exclusion of certain risk factors via a drop-down menu, etc.). The purpose of these analyses will be to inform the study managers, identify problems quickly, and allow for the planning of contingencies in the case of unexpected or adverse results. In the stage-gate phase of the project, several automated analyses will be built mainly for the purpose of providing examples for later (post stage-gate) analytical approaches.

VII. Dissemination:

Knowledge dissemination is not foreseen as a major component of the stage-gate phase of this project. DataBrew intends to propose the construction of a platform for the distribution of knowledge products such as interactive visualizations and maps; pre-print web hosting; online "toolkits" and information packets regarding research findings; etc. However, this platform does not figure into the stage-gate component of this project, nor this proposal.

Technical details/specifications

Software: For reasons of (a) reproducibility, (b) costs, and (c) transferability to partners, DataBrew will use open-source software and programming languages all work related to this project. These software are as follows:

1. Collection: OpenDataKit (Java-based) running on Android
2. Validation: PosgreSQL and Python (scripting)
3. Storage: PosgreSQL databases (running on AWS* platform)
4. Accessibility: Python flask (for browser layer) and Ruby Jekyll
5. Visualization: Shiny (R)
6. Analysis: R, Rmarkdown, Shiny, and Python
7. Dissemination: Ruby Jekyll

*AWS (Amazon Web Services) is a commonly used cloud service platform for data storage.

Code: All code for this project well be documented and publicly hosted (via a code-hosting service such as Gitlab or Github). By making code available, (a) its correctness can be verified independently, (b) their is less reliance on the external contractor (us), and (c) the data products can be re-used by partners in different scenarios.

Technical assistance: All data products will be accompanied by (a) documentation for both engineer and regular users and (b) instructional materials for training.

FDA 21 CFR Part 11 compliance

All work will be compliant with FDA 21 CFR Part 11. See checklist in appendix.

Protocol adherence

As per project requirements, DataBrew will work closely with project partners to ensure the adequacy of measures taken to for guaranteed double storage (see FDA checklist), local site data ownership, hardware compatibility, and appropriate survey control flow.

Technical assistance and ongoing support

As per work with previous clients, DataBrew will remain available to provide technical assistance, guidance, and basic modifications after the project period officially ends.

Detailed budget

Budget period: February 1st, 2019 - December 31st, 2019
Total: $181846.69
A full, itemized budget is viewable at: http://www.databrew.cc/bohemiabudget.

Examples of previous work

DataBrew has worked in successful collaborations with clients in both industry and academia. What follows are a few projects with relevance to this proposal:

- **VIDA (Vaccine Impact on Diarrhea in Africa) / University of Maryland (URL)**
    - DataBrew built a custom R package (here) and dataset-compatibility ingestion tool for the standardization and aggregation of public health datasets in multiple formats (project outline, online tool)
- **Santa Fe Institute (URL)**
    - DataBrew built custom visualizations, working with multiple academic contributors, for a standardized aesthetic in an upcoming publication on law and data.
- **York University / YouthRex (URL)**
    - DataBrew built a data management/munging pipeline and built a data exploration dashboard of youth social outcomes data in Ontario (online tool).
- **International Labour Organization (World Bank Group) / BetterWork (URL)**
    - DataBrew carried out data cleaning and survey aggregation work, and built an interactive web application / dashboard for exploration of survey data on work conditions in 5 countries
- **International Finance Corporation (World Bank Group) / Monitoring Evaluation and Learning (MEL)**
    - Built multiple dashboards using internal financial data (example 1, example 2)
- **Individual work**
    - DataBrew partners have extensive work and research experience (details)

Proof of previous international work

DataBrew has units in both the United States and Canada. Our clients have been from the US (ILO, SFI), Senegal (IFC/MEL), Canada (York), and multi-country (VIDA: Kenya, Gambia, Mali, USA).

DataBrew partners have work and research experience in North America (USA, Canada), Europe (Spain, France, Denmark, Netherlands), Africa (Mozambique, Togo, Kenya) and Asia (Nepal, China). CVs.

DataBrew often works as a “distributed” team, and has had many successful collaborations with foreign partners via both travel and exclusive remote work.

References are available upon request.

Appendix

CFR21 Part 11 Compliance checklist

Item	Response
Is all software validated	There are multiple software "layers" in this proposal. At the most basic level, some software services will be purchased (AWS, Shinyapps). These softwares are industry norms, and can be considered "validated". A second layer is the open-source programming software (Ruby, R, Python) which we will use for scripting and analysis. These softwares are commonly used industry and academia. They are open-source and collaborative by nature and therefore cannot clearly be classified as "validated" or not. A final layer is the custom software (applications, R packages, etc.) which we will build for this project. These, by definition, have not been validated. They *will* be validated for correct use in collaboration with the partner through iterative testing and tweaking.
Is software password protected	Those software components which require authorization restriction (database access, web application access, etc.) will be password protected.
Does every individual user have their own user name and password	For those application components where authorization restriction is applicable (see above) all users will their own unique user names and passwords.
Are there role-based access levels? E.g. entry/modification/deletion	For those application components where authorization restriction is applicable (see above), differential access groups will be created. For database access, there will be two groups (administrator and view-only). For web applications, there will be at least three groups (administrator, view-only, aggregated view only), to be determined in function of project needs.
Do you have a master list of users and roles that is updated as required?	This will be generated during software development.
Is there a clear audit trail for all entries/modifications/deletions – user, time and date?	All datasets will be built using a "state machine" framework, with a clear audit trail of all modifications and their source.
Are electronic signatures used and have staff signed a declaration that their electronic signature is binding?	Electronic signatures will not be used.
Does the format of the electronic signature show signature, name of signatory, company, designation, date of signature and reason for the signature?	NA
Is there a daily/weekly/monthly backup system?	Daily data dumps will be generated from the databases. Code will be version controlled via git.

Has the backup system been tested?	The backup system will be tested in the initial development phase.
Do you have a disaster recovery plan (DRP) in place?	A disaster recovery plan will be created in the initial development phase.
Do you have desktop/laptop(s) available for CRAs to access medical records for monitoring?	A laptop will be available.
Do you have a user level set for them?	User level datasets will be made available.
Do you have documentation in place to record this access and conditions of use?	Documentation will be generated in the initial development phase.
Do you have SOPs in place for the way you use the software e.g. training, granting of access, backup, change management (upgrades to software and how data integrity is maintained)	Software will be written in iterative collaboration with the project partners. SOPs will be devised concurrently. All software will be documented and version controlled. Upgrades and updates will be pre-announced, and pre-tested prior to deployment.
Can the system ever be offline? If so, what is the backup plan (part of DRP)?	All systems can suffer technical failures causing them to go offline. Our “offline plan” will be integrated into our DRP. Additionally, for data collection, offline status will be engineered as “default status”, since many areas of data collection will have no internet connectivity.
How will you store participant medical records once a study is completed? Can you guarantee a format that will not be legacy bound i.e. accessible and readable 15 years from date of archiving	Since we are using open-source software, our formats will not be legacy-bound. Therefore, future accessibility / readability is guaranteed.