Master's Degree Candidate Project

Data Integration Project With Upstate Medical University

Joseph Miles, PharmD

SUNY Oswego

Biomedical and Health Informatics 698

Table of Contents

Please address all comments or questions to: Joseph Miles, jmiles3@oswego.edu

List of Abbreviations and Key Terms:

biological = genetically related
CITI = Collaborative Institutional Training and Initiative
csv = comma separated values
ETL = Extraction, Transformation, and Loading
FamID = Unique identifier for biological families
FSD = Family Studies Demographics
GUID = Global Unique Identifier (NIMH)
Ibid = In the same source as previous
Ind. ID = individual identifier for Upstate
index = sequentially numbered location of a value in an array of values
NIH = National Institute of Health
NIMH = National Institute of Mental Health
RDF = Resource Description Framework
RDoC = Research Domain Criteria
Res_Subj= Research Subjects (file)
UML = Unified Modeling Language
Upstate = Upstate Medical University
vars = abbreviation for variables

Abstract

Proper data stewardship is essential for sharing biomedical data. Beyond accounting for the time to clean a dataset of missing and inaccurate values, considerable care is necessary for accurate data extraction, transformation, loading (ETL) and integration into the exact format specified for a shared database. In addition to the resource of time, manually manipulating data brings the potential for human copy and paste errors and typographical errors that could add additional noise to a dataset. A custom-built application will manipulate data accurately and consistently while also saving time. For this internship experience, I built an application for Upstate Medical University to integrate research data into a pre-specified format for loading study data into a data warehouse operated by the National Institute of Mental Health.

Effective sharing of biomedical data leads to better research, improved knowledge, and expands practitioners' abilities to individually treat each patient.

Introduction

"*The decentralized nature of our scientific communities and healthcare systems has created a sea of valuable but incompatible electronic databases.*" (Sujansky, 2001)

Heterogeneous methods of labelling and storing data across various institutions confounds the efforts to integrate data (Sujansky, 2001). Institutions may store data in identically named variables which have different semantic meanings or they may label identical data categories with different semantics. A data warehouse is ideal for establishing a universal semantic standard and for the curation and quality control of data (Louie et al, 2007). Data warehouses sometimes have a delay in the availability of data, but a well-planned data warehouse will help in the consistency, accuracy, completeness, reliability, availability, accessibility, and interpretability of the data (Schmidt and Prado, 2014). To maintain the integrity of the data in the data warehouse, the process of extraction, transformation and loading (ETL) of the data is of highest importance (Schmidt and Prado, 2014).

An exciting goal of modern medicine is the practice of precision medicine. Precision medicine is objectively treating each individual with therapies that are known to be highly effective based on the patient's genetics, living environment, and life style. To practice precision medicine with any degree of certainty, a great deal of data would need to be compiled and searchable from a universally accessible database. In the realm of psychiatric medicine, the National Institute of Mental Health (NIMH) is compiling such a data warehouse for the Research Domain Criteria (RDoC) initiative (NIMH, "RDoC Matrix").

Background

Under the supervision of Dr. Stephen Glatt, our team at Upstate Medical University (Upstate) is conducting a family RDoC study which is gathering data for multiple constructs in the psychiatric Positive Valence System (see NIMH, "RDoC Matrix"). A goal of RDoC is to accumulate vast amounts of data and to integrate "*many levels of information (from genomics to self-report) to better understand basic dimensions of functioning underlying the full range of human behavior from normal to abnormal*" (NIMH, "Research Domain Criteria"). As discussed in an article coauthored by Dr. Glatt, currently known candidate genes associated with the psychiatric Positive Valence System are only loosely correlated with changes in subjects' observed symptoms (r^2 values range from 0.181 to 0.233 meaning that only approximately 20% of the change in a symptom can be attributed to the candidate gene) and frequently lack data for genetic inheritance (Hess et al, 2016). By compiling a large body of genomic data coupled with reports of symptomology and other patient-specific traits, new candidate genes may be identified and correlated with inheritance and expected patient outcomes (Hess et al, 2016).

Each individual that participates in the Upstate study is currently represented by a record within a spreadsheet. Each record in the spreadsheet contains 96 variables for each individual. As of October 28, 2016, the spreadsheet contained 1362 individuals.

Every six months, NIMH requires Upstate to submit updated information for all study subjects. NIMH requests the uploaded data to be in a comma separated values

(csv) format with pre-specified variable titles. In line with ideal data warehousing procedure, uploads to the NIMH data warehouse need to be arranged with a specific format using pre-determined value ranges to ensure uniformity of data across all RDoC research locations. Since the study involves genetically related (biological) families, NIMH requests two different csv files: one file with each family unit represented as a unique record containing data for each family member ("Family Studies Demographics," FSD file) and a second csv file with each research subject represented as their own record ("Research Subjects," Res_Subj file). The study has been ongoing for approximately two years and the ETL process has been performed manually up until January of 2017. From self-report of the Upstate RDoC research team, the manual process of ETL has consumed at least 80 work-hours every six months.

Project Objective

The primary objective of this project was to assist Upstate in the process of semi-automating the ETL process for their RDoC study data. It was determined that writing a new computer application for performing this specific ETL task would be ideal.

Project Timeline

My time spent on the project can be broken down into three phases. Since I had no previous experience with working with human subject data, I first needed to be trained on the appropriate use of human subject data. To that end, I acquired two certificates from the Collaborative Institutional Training and Initiative (CITI) Program for "Biomedical Data or Specimens Only Research" and "Conflicts of Interest" and a certificate from the National Institutes of Health (NIH) for "Protecting Human Research Participants" (Certificates of Completion, Appendix 3).

The Python programming language was ideal for this project because of Python's ability to easily and intuitively manipulate data. Before this project, I had limited exposure to computer programming and the Python language. My second phase in this project was to familiarize myself with Python and best practices in computer programming (such as

learning Unified Modeling Language (UML)). I completed three "Programming Foundations" courses through Lynda.com: programming "Fundamentals," "Object-Oriented Design" and "Real-World Examples," as well as two Python courses: "Learning Python" and "Python GUI Development with Tkinter" (Certificates of Completion, Appendix 3).

The third phase of the project involved writing the program to extract, transform, sort, integrate and write the output necessary for upload into the NIMH system. Initially, a plan was made for the structure of the program using UML (see Figure 1). Then, program pseudocode was sketched on paper (Appendix 4 has an abbreviated version of the pseudocode, original pseudocode is too unintelligible to include here). The pseudocode was translated to Python code using the Geany (version 1.28) code editor.

Methods

A data dictionary for each upload file was specified by the data definitions from NIMH (see Appendix 1). From the data dictionary, a simplified input dictionary of 14 variables was determined to be needed from each research subject as listed in Table 1.

Table 1

Table of input variables needed for the data dictionary	
Variable Name	Transformation necessary: Convert data to NIMH format (*Yes* or *No*)
Family ID *(Part of composite key)	No
Individual ID *(Part of composite key)	No
Study Visit Date	No
Mental Health	No
Date of Birth	Yes
Gender	Yes
Racial Category	Yes
Ethnic Category	Yes
Highest Level of Education	Yes
Employment Status	Yes
Marital Status	No
Household Annual Income	No
Blood Sample ID	Yes
Saliva Sample ID	Yes

Once the location of the necessary data was identified, it was then determined which data would need to be transformed to fit the value ranges specified by NIMH (as indicated with “Yes” in the second column of Table 1). A parallel integration task was to read a second input file containing a unique and deidentified NIMH ID (also called a Global Unique Identifier (GUID)) for each research subject that provided informed consent for sharing their data with NIMH. The NIMH ID was then used as an identifier for each research subject in the RDoC upload files. The extracted and transformed data then needed to be integrated into one or both NIMH csv files for upload. The integration process required grouping research subjects into biological families and assimilating each subject’s data into the NIMH specified variable locations. The unique Upstate ID and NIMH ID of genetically related family members are requested variables in the Res_Subj

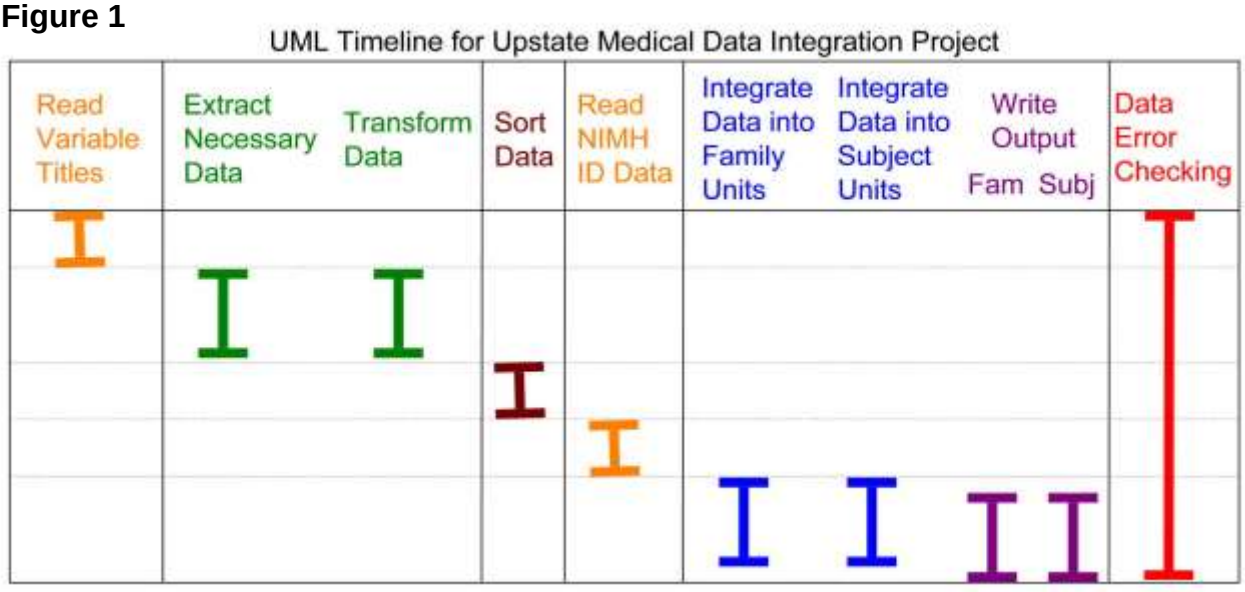
file, so sorting into family groups is essential for creating both the FSD file and the Res_Subj file.

The Python programming language, version 3.2, was used to create the RDoC-integration application.

As previously stated, the dataset contained 1362 research subjects. The data was read from a csv formatted spreadsheet. Each research subject record had 96 variables. From the 96 variables, the 14 key variables named in Table 1 were extracted. Additional data was derived from the extracted variables (such as deriving the name of the repository holding whole blood samples based on the biorepository ID) and data from genetically related research subjects was combined to construct the records of both RDoC upload files. The upload files contained 53 variables for each family in the FSD file and 51 variables for each research subject in the Res_Subj file (Appendix 1).

Before code was written, a UML timeline was constructed to outline the distinct functions of the application and the order of operations for each function (Figure 1).

Figure 1



Analysis of the UML Timeline in Figure 1 shows which functions are performed simultaneously. Due to the overlap of functionality, as can be seen in the program pseudocode in Appendix 4, integration of all data and writing of output should be

performed simultaneously for maximum efficiency. Loading of the data into the NIMH servers takes place through a utility provided by NIMH and is not a part of the timeline.

The source files used in this project are not constant. New variables have been added to the dataset over the course of the first two years of the study. Since the location of data within the dataset is not constant, the Python application needed to read the variable titles in the source documents (see Figure 2 for an example of the source titles). The location of the necessary titles was indexed by the Python application so that pertinent data could be read (as specified in Table 1) from each indexed location for each individual.

Figure 2: Sample titles from Upstate's spreadsheet of research subject data.

FamID	Ind. ID	Date of Birth	Gender 1=Male 2=Female	Racial Category 1=Am Indian or Alaska Native 2=Asian 3=Black/African American	Racial Category (Other specify)	Ethnic Category 1=Hispanic or Latino; 2=Not Hispanic or	Highest Level of Education 1=Less than HS 2=HS diploma or equivalent 3=Some college -	Employment Status 1=Unemployed 2=Employed part-time 3=Employed	Employment Status (Other Specify)	Marital Status 1=Married 2=Never married 3=Not	Household Annual Income 0=NA (child)	Parent agrees to repositories 1=yes both 2=Local Study only

Once the data was located, values were extracted and transformed (if necessary) for each individual and stored using a primary key of the composite of "FamID" and "Ind. ID" (these titles can be seen in Figure 2). Primary keys were then sorted, placing each family into a group and then sorted by "Ind. ID" within each family. The "Ind. ID" value signifies that individual's role in the family grouping: 01 is the primary research subject, in this case, a child with a mental health issue; 02 is the genetic father of 01; 03 is the genetic mother of 01; 04, 05, 06, and 07 are genetic siblings of 01 (NIMH is only requesting data for up to 4 siblings at this time).

Next, the NIMH ID information is read from a second csv file and stored under a matching primary key (FamID + Ind. ID).

The NIMH upload files are constant, so the Python application was programmed to know the exact index destination for each research subject's data. The application iterates through each "Ind. ID" and populates the indexed locations for each NIMH upload

file. Once the application reaches a new “FamID,” the program writes the accumulated records (which includes one family record and a separate research subject record for each family member) into the appropriate NIMH upload files. The process of populating record data and writing the output is repeated for each family.

Once the application finishes iterating through every record, all data will be integrated and processed for upload. The created csv files can then be manually reviewed and eventually uploaded to NIMH.

Results

Field testing of a beta version of the Python application was performed on April 10, 2017. In total, 1354 of Upstate’s 1362 research subject records processed correctly. Due to idiosyncratic and missing data, it was expected that some families would not be processed by the Python application. Since the unprocessed records have historically not been uploaded to the NIMH data warehouse, it is only an approximation to say the program was 99.4% successful. Both the FSD and Res_Subj files have been created by the application and have been confirmed to be correctly formatted and all data is presented in the acceptable ranges. During field testing, I was made aware of an additional variable in the source file that indicates whether each individual has given informed consent for sharing their data with NIMH. Program code was added to find this variable and skip any individual that has not given consent. Since the consent variable was unknown in the planning stage, there is no other mention of this variable in this current document.

The application processes all data in seconds, effectively saving at least 80 work-hours for the Upstate RDoC team for each six-month reporting cycle. All data will transform and integrate consistently every time the application is run. By using a custom-made application for the process of ETL and integration, human typographical errors and copy and paste errors will be eliminated. Since the process is semi-automated, there will still be the opportunity for human intervention to manage idiosyncratic data such as managing split families where genetic siblings may be represented under a separate

Upstate family group ("FamID"). Output data is readable by any software that can read csv files (such as Microsoft Excel) allowing human quality assurance to safeguard for appropriate transformation and integration of data.

Images of the application interface and user instructions can be seen in Appendix 2.

Future Plans

Actual loading of the data output from this project into the NIMH system will not take place until July 2017. Until that time, I will work on improvements in the application's usability by implementing a graphical interface. I will also refine the application's capability for managing idiosyncratic data.

"Data error checking," as indicated in the right-most column of Figure 1, has not been fully implemented. Code will be added to the program to look for missing data, including situations where a family may be missing a key individual such as the primary subject ("01" or proband). The goal of the next revision of the application will be to process all 1362 records, including those with idiosyncratic or missing data, and notify the user via a third output file (named "needs_attention.txt") of when and which records are not included in the output along with a reason why the record needed to be excluded.

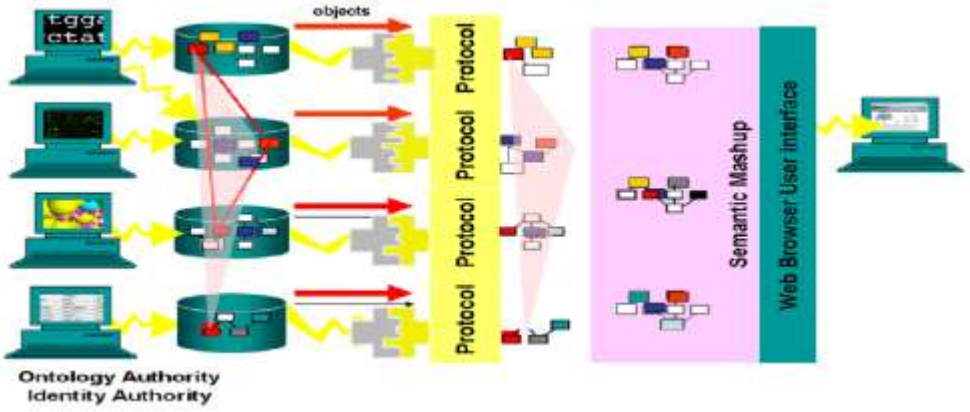
The source file used for field testing has an unstructured text field where the team can make notes about individuals and/or families. A future release of the RDoC-Integr8 application will read this free-text and include notes to the user in the needs_attention.txt output file. Additionally, the needs_attention.txt file will contain statistics for how many records were processed and an accounting of the location of the original source data for processing reproducibility.

Discussion

Creating a successful data warehouse requires planning and rules to maintain the integrity of data as well as usability / interpretability of the data. Goble and Stevens (2008) specify the following 6 needs for sharing complex biomedical data: 1) The need for common, shared identities and names; 2) The need for shared semantics; 3) The need for shared and stable access to the data; 4) The need to adhere to the established standards; 5) The need to explicitly state collection policies and governance; and 6) The need to balance data curation with data usability.

Goble and Stevens (2008) predict future adaptations in Resource Description Framework (RDF) data warehouses that will more fully integrate the ideals of the Semantic Web "*to create a single Web of biological data and knowledge that can be crawled and queried*" using familiar web browsing user interfaces. Figure 3 is a graphical representation of semantic sharing of data.

Figure 3: Data sharing using semantic tag linkage and controlled vocabularies. Data is then accessed through a Web browsing interface (Goble and Stevens, 2008)



The ontology mentioned in Figure 3 refers to a common concept definition agreed upon by various researchers to facilitate mutual understanding for a set of concepts (Karasavvas, Baldock and Burger, 2004).

Full semantic linking of biomedical data is not yet a reality, so we still need to be good stewards of data integrity to ensure proper data classifications for accurate data analyses.

Despite clinical diagnoses having a heterogeneous mix of determinant factors, the trend is to treat patients based on a diagnosis rather than treating the problem(s) that initially brought the individual to the clinician (Cuthbert and Insel, 2010). Creating more specific classifications of disorders will lead to precise treatment of each patient. From what we can see in this current project, RDoC is an effort to supplement heterogeneous diagnosis classifications with homogenous groupings of characteristics within a matrix of psychiatric constructs. For example, under a grouping labelled "Positive Valence Systems," constructs are labeled as "Approach Motivation," "Initial Responsiveness to Reward Attainment," "Reward Learning," and "Habit" (NIMH, "RDoC Matrix"). The constructs exist as a way for researchers to pattern their investigations to accumulate research subject data including patient self-report of symptomology, environmental considerations, brain activity, genomics and inheritance.

Beyond accurate classification of symptomology for each individual patient, an additional goal of the NIMH is to use the accumulated data to allow clinicians to prevent the onset of mental disease symptoms. "*The best time to address a mental illness is before the appearance of symptoms that disrupt daily life*." (NIMH, "Strategic Objective 2")

Educational Statement

Utilizing a programming language like Python is effective for manipulating large amounts of data. Using a customized application for the ETL process eliminates the chance for variations in interpretation of the rules for extraction and transformation. Since the application is not copying and pasting data from a 2-dimentional spreadsheet, there is no chance that the program will accidentally copy from an incorrect record in the spreadsheet. Any errors performed by the application would be consistent and would be ubiquitous in the output. At this time, no errors in the currently discussed application's output have been found.

I have a heightened appreciation for data cleaning and the importance of accurately executing the ETL process. Without precision in transforming data into an

acceptable format, a data warehouse cannot be effectively used for data analysis and "as needed" querying of the data will not be possible.

Data is messy. Our habit is to accumulate as much data as possible. The end result is that we end up with too many variables and inconsistent recording of values. The specific structure of the RDoC data warehouse, as can be interpreted from Appendix 1, fills the need for 1) variable titles and type of data; 2) specific value ranges; 3) a common location to store RDoC data; 4) insistence that data adheres to the specified semantics; 5) specific cycles and rules for collecting data; and 6) a method for storing the data so all researchers will be better able to access the data and create more thorough conclusions based on the available data (adapted from the data integration "needs" in Goble and Stevens, 2008).

"*Integration is a complex task aiming to provide a unified view of the underlying resources, while eliminating potential technical and semantic heterogeneity.*" (Karasavvas, Baldock and Burger, 2004). The unification of biomedical data will result in better research, improved knowledge, and more personalized care for each person.

Acknowledgements

I would like to thank Dr. Isabelle Bichindaritz for advising me and leading me through the planning stages of this project. Thank you to SUNY Oswego for preparing me for the world of Biomedical Informatics and Health Information Technology. I would like to thank Dr. Stephen Glatt and the RDoC team at Upstate Medical University. Finally, a special thanks to Cheri Roe for instructing me in all things RDoC; I hope the new RDoC-Integr8 application continues to help with all the future NIMH data uploads.

Please address all comments or questions to: Joseph Miles, jmiles3@oswego.edu

References

Cuthbert, B; Insel, T. (2010). The data of diagnosis: new approaches to psychiatric classification. *Psychiatry: Interpersonal & Biological Processes*. *73*(4): 311-314. doi:10.1521/psyc.2010.73.4.31

Goble, C; Stevens, R. (2008) State of the nation in data integration for bioinformatics. *Journal of Biomedical Informatics*. 41: 687-693. doi:10.1016/j.jbi.2008.01.008

Hess, JL; Kawaguchi, DM; Wagner, KE; Faraone, SV; Glatt, SJ. (2016). The influence of genes on "positive valence systems" constructs: A systematic review. *American Journal of Medical Genetics Part B: Neuropsychiatric Genetics*, *171*(1), 92-110.

Karasavvas, K; Baldock, R; Burger, A. (2004). Bioinformatics integration and agent technology. *Journal Of Biomedical Informatics*. 37: 205-219. doi:10.1016/j.jbi.2004.04.003

Louie, B; Mork, P; Martin-Sanchez, F; Halevy, A; Tarczy-Hornoch, P. (2007). Data integration and genomic medicine. *Journal Of Biomedical Informatics*. 40: 5-16. Doi: 10.1016/j.jbi.2006.02.007

Nation Institute of Mental Health (NIMH). "RDoC Matrix" retrieved 4/30/2017 from https://www.nimh.nih.gov/research-priorities/rdoc/constructs/rdoc-matrix.shtml. "Research Domain Criteria (RDoC)" retrieved 4/28/2017 from https://www.nimh.nih.gov/research-priorities/rdoc/index.shtml. "Strategic Objective 2" retrieved 4/30/2017 from https://www.nimh.nih.gov/about/strategic-planning-reports/strategic-objective-2.shtml

Schmidt, SO; Prado, EPV. (2014) IT Architecture and Information Quality in Data Warehouse and Business Intelligence Environments. *IGI Global*. 6: 121-127. DOI: 10.4018/978-1-4666-4892-0.ch06

Sujansky, W. (2001). Heterogeneous database integration in biomedicine. *Journal Of Biomedical Informatics*. 34(4): 285-298. Doi:10.1006/jbin.2001.1024

Python 3.2 was downloaded from https://www.python.org/download/releases/3.2/

Geany 1.28 code editor was downloaded from https://www.geany.org/

Appendix 1: Full data dictionary as specified by NIMH

Selected value ranges explained	
Highest Level of Education Achieved	99=Don't know; 77=Refused; 1=8th grade or less; 2=9th grade or above, not a high school graduate; 3= high school graduate or GED; 4= post high school education, no college degree; 5=vocational-technical = degree or certificate; 6=2-year college degree/associate degree; 7= 4-year college degree/BA, BS degree; 8= some BA/BS work, no degree; 9=master degree, e.g. MSW, MA, MFA, MPH, MBA; 10=PHD, MD, JD, LLB, or other professional graduate degree; 11=other; 12=Less than high school diploma/GED
Current employment	1=Working full time(35hrs or more a week at one or more jobs); 2=Working part time; 3=Keeping house; 4=Unemployed, looking for work; 5=Unemployed, not looking for work; 6=Disabled; 7=Retired; 8=Student, full time; 9=Student, part time; 10=Other; 11 =Unemployed; 99=Not in household; 12 = Self-employed
Marital status	1=Married; 2=Never Married; 3=Not married (living together); 4=Divorced; 5=Separated; 6=Widowed
Household annual income	1=<3,001; 2=3,001-5,000; 3=5,001-10,000; 4=10,001-20,000; 5=20,001-30,000; 6=30,001-40,000; 7=40,001-50,000; 8=50,001-75,000; 9=75,001-100,000; 10=100,001 and above; 11=Would rather not say

Appendix 1a: Family Studies Demographics

ElementName	DataType	Size	Required	ElementDescription	ValueRange
subjectkey	GUID		Required	The NDAR Global Unique Identifier (GUID) for research subject	NDAR*
src_subject_id	String	20	Required	Subject ID how it's defined in lab/project	
interview_date	Date		Required	Date on which the interview/genetic test/sampling/imaging was completed. MM/DD/YYYY	
interview_age	Integer		Required	Age in months at the time of the interview/test/sampling/imaging.	0 :: 1260
gender	String	20	Required	Sex of the subject	M;F
race	String	30	Recommended	Race of study subject	American Indian/Alaska Native; Asian; Hawaiian or Pacific Islander; Black or African American; White; More than one race; Unknown or not reported
ethnicity	String	30	Recommended	Ethnicity of participant	Hispanic or Latino; Not Hispanic or Latino; Unknown
subjectkey_father	GUID		Recommended	The NDAR Global Unique Identifier (GUID) for subject's biological father	NDAR*
src_father_id	String	100	Recommended	site specific father ID	
ques_agebf	Integer		Recommended	Age -Biological Father(in months)	0 :: 1200; -999
ques_genderbf	String	50	Recommended	Gender -Biological Father	M;F
bio_father_race	String	150	Recommended	Biological Father Race	American Indian/Alaska Native; Asian; Hawaiian or Pacific Islander; Black or African American; White; More than one race; Unknown or not reported

bio_father_ethnicity	String	50	Recommended	"Biological Father ethnicity (Hispanic or Latino, Not Hispanic or Latino, No Answer)"	Hispanic or Latino; Not Hispanic or Latino; No Answer
et2b	Integer		Recommended	Highest Level of Education Achieved. Biological father	1::12; 99; 77
dem_08b	Integer		Recommended	Father current employment	1::12;99
fa_maritalstatus	Integer		Recommended	Marital status (biological father)	1::6
fa_householdincome	Integer		Recommended	Household annual income (biological father)	1::11
subjectkey_mother	GUID		Recommended	The NDAR Global Unique Identifier (GUID) for subject's biological mother	NDAR*
src_mother_id	String	100	Recommended	site specific mother ID	
ques_agebm	Integer		Recommended	Age -Biological Mother(in months)	0 :: 1200; -999
ques_genderbm	String	50	Recommended	Gender -Biological Mother	M;F
bio_mother_race	String	150	Recommended	Biological Mother Race	American Indian/Alaska Native; Asian; Hawaiian or Pacific Islander; Black or African American; White; More than one race; Unknown or not reported
bio_mother_ethnicity	String	50	Recommended	Biological Mother Ethnicity	Hispanic or Latino;Not Hispanic or Latino; No Answer
et2a	Integer		Recommended	Highest Level of Education Achieved. Biological mother	1::12; 99; 77
dem_08a	Integer		Recommended	Mother current employment	1::11;99
mo_maritalstatus	Integer		Recommended	Marital status (biological mother)	1::6
mo_householdincome	Integer		Recommended	Household annual income (biological mother)	1::11
subjectkey_sibling1	GUID		Recommended	The NDAR Global Unique Identifier (GUID) for subject's sibling	NDAR*
src_sibling1_id	String	100	Recommended	site specific sibling1 ID	
ques_age1	Integer		Recommended	Age -Sibling 1(in months)	0 :: 1200
ques_gender1	String	50	Recommended	Gender -Sibling 1	M;F;T
sib1_race	String	30	Recommended	Race- Sibling 1	American Indian/Alaska Native; Asian; Hawaiian or Pacific Islander; Black or African American; White; More than one race; Unknown or not reported
sib1_ethnicity	String	30	Recommended	Ethnicity- Sibling 1	Hispanic or Latino; Not Hispanic or Latino; No Answer
subjectkey_sibling2	GUID		Recommended	The NDAR Global Unique Identifier (GUID) for subject's sibling	NDAR*
src_sibling2_id	String	100	Recommended	site specific sibling2 ID	
ques_age2	Integer		Recommended	Age -Sibling 2(in months)	0 :: 1200

ques_gender2	String	50	Recommended	Gender -Sibling 2	M;F
sib2_race	String	30	Recommended	Race- Sibling 2	American Indian/Alaska Native; Asian; Hawaiian or Pacific Islander; Black or African American; White; More than one race; Unknown or not reported
sib2_ethnicity	String	30	Recommended	Ethnicity- Sibling 2	Hispanic or Latino; Not Hispanic or Latino; No Answer
subjectkey_sibling3	GUID		Recommended	The NDAR Global Unique Identifier (GUID) for subject's sibling	NDAR*
src_sibling3_id	String	100	Recommended	site specific sibling3 ID	
ques_age3	Integer		Recommended	Age -Sibling 3(in months)	0 :: 1200
ques_gender3	String	50	Recommended	Gender -Sibling 3	M;F
sib3_race	String	30	Recommended	Race- Sibling 3	American Indian/Alaska Native; Asian; Hawaiian or Pacific Islander; Black or African American; White; More than one race; Unknown or not reported
sib3_ethnicity	String	30	Recommended	Ethnicity- Sibling 3	Hispanic or Latino; Not Hispanic or Latino; No Answer
subjectkey_sibling4	GUID		Recommended	The NDAR Global Unique Identifier (GUID) for subject's sibling	NDAR*
src_sibling4_id	String	100	Recommended	site specific sibling4 ID	
ques_age4	Integer		Recommended	Age -Sibling 4(in months)	0 :: 1200
ques_gender4	String	50	Recommended	Gender -Sibling 4	M;F
sib4_race	String	30	Recommended	Race- Sibling 4	American Indian/Alaska Native; Asian; Hawaiian or Pacific Islander; Black or African American; White; More than one race; Unknown or not reported
sib4_ethnicity	String	30	Recommended	Ethnicity- Sibling 4	Hispanic or Latino; Not Hispanic or Latino; No Answer
mother_other_race	String	20	Recommended	Mother Other race (please specify):	
father_other_race	String	20	Recommended	Father Other race (please specify):	

Appendix 1b: Research Subjects File

ElementName	DataType	Size	Required	ElementDescription	ValueRange
Subjectkey	GUID		Required	The NDAR Global Unique Identifier (GUID) for research subject	NDAR*
src_subject_id	String	20	Required	Subject ID how it's defined in lab/project	
interview_date	Date		Required	Date on which the interview/genetic test/sampling/imaging was completed. MM/DD/YYYY	
interview_age	Integer		Required	Age in months at the time of the interview/test/sampling/imaging.	0 :: 1260
Gender	String	20	Required	Sex of the subject	M;F
Race	String	30	Required	Race of study subject	American Indian/Alaska Native; Asian; Hawaiian or Pacific Islander; Black or African American; White; More than one race; Unknown or not reported
ethnic_group	String	255	Recommended	Ethnic group	
Phenotype	String	200	Required	Phenotype/diagnosis for the subject	
phenotype_description	String	4000	Required	Description of the phenotype for the subject	
twins_study	String	100	Required	Is this study of twins?	Yes;No
sibling_study	String	50	Required	Was it sibling study? Study of sibling(s) of autistic child.	Yes;No
family_study	String	100	Required	Was it family study? Study of biological mother, biological father and/or sibling of proband.	Yes;No
family_user_def_id	String	20	Conditional	Family Pedigree User-Defined ID	
subjectkey_mother	GUID		Recommended	The NDAR Global Unique Identifier (GUID) for subject's biological mother	NDAR*
src_mother_id	String	100	Conditional	site specific mother ID	
subjectkey_father	GUID		Recommended	The NDAR Global Unique Identifier (GUID) for subject's biological father	NDAR*
src_father_id	String	100	Conditional	site specific father ID	
subjectkey_sibling1	GUID		Conditional	The NDAR Global Unique Identifier (GUID) for subject's sibling	NDAR*
src_sibling1_id	String	100	Conditional	site specific sibling1 ID	
sibling_type1	String	255	Conditional	Type of Sibling	Full Brother FB; Half Mother Brother HMB; Half Father Brother HFB; Full Sister FS; Half Mother Sister HMS; Half Father Sister HFS

subjectkey_sibling2	GUID		Recommended	The NDAR Global Unique Identifier (GUID) for subject's sibling	NDAR*
src_sibling2_id	String	100	Recommended	site specific sibling2 ID	
sibling_type2	String	255	Recommended	sibling type	Full Brother FB; Half Mother Brother HMB; Half Father Brother HFB; Full Sister FS; Half Mother Sister HMS; Half Father Sister HFS
subjectkey_sibling3	GUID		Recommended	The NDAR Global Unique Identifier (GUID) for subject's sibling	NDAR*
src_sibling3_id	String	100	Recommended	site specific sibling3 ID	
sibling_type3	String	255	Recommended	sibling type	Full Brother FB; Half Mother Brother HMB; Half Father Brother HFB; Full Sister FS; Half Mother Sister HMS; Half Father Sister HFS
subjectkey_sibling4	GUID		Recommended	The NDAR Global Unique Identifier (GUID) for subject's sibling	NDAR*
src_sibling4_id	String	100	Recommended	site specific sibling4 ID	
sibling_type4	String	255	Recommended	sibling type	Full Brother FB; Half Mother Brother HMB; Half Father Brother HFB; Full Sister FS; Half Mother Sister HMS; Half Father Sister HFS
zygosity	String	255	Conditional	Zygosity	monozygous; dizygous; trizygotic
sample_taken	String	50	Required	Was a sample taken at this interview/during this project time	Yes;No
sample_id_original	String	100	Conditional	Original, user-defined Sample ID	
sample_description	String	3500	Conditional	Sample description: tissue type, i.e. blood, saliva, brain etc.	whole blood; saliva; brain; urine; serum; plasma; CSF; IPS; Fibroblast; Neuronal Progenitor; skin biopsy; temporal cortex; lymphoblastoid cell line; unknown; semen; frontal cortex; parietal cortex; basal ganglia; Placenta
biorepository_name	String	100	Conditional	Biorepository where sample is stored	
sample_id_biorepository	String	100	Conditional	Biorepository Sample ID	
patient_id_biorepository	String	100	Conditional	Biorepository Patient ID	
cell_id_original	String	100	Recommended	Original, user-defined cell line ID	
cell_id_biorepository	String	100	Recommended	Biorepository cell line ID	
agre_subject_id	String	100	Recommended	AGRE subject ID	

sfari_subject_id	String	100	Recommended	SFARI subject ID	
cpea_site	String	100	Recommended	CPEA/STAART site name	
cpea_id	String	100	Recommended	CPEA/STAART subject ID	
blood_id	String	100	Recommended	blood ID	
adi_dx	String	15	Recommended	ADI: Diagnosis	
ados_dx	String	15	Recommended	ADOS Diagnosis	
agp_family_id	String	100	Recommended	AGP family ID	
agp_subject_id	String	100	Recommended	AGP subject ID	
site	String	100	Recommended	Site	
comments_misc	String	4000	Recommended	Miscellaneous comments on study, interview, methodology relevant to this form data	
week	Float		Recommended	Week in level/study	
study	String	100	Recommended	Study	

Appendix 2: User instructions for the RDoC integration application (RDoC-Integr8).

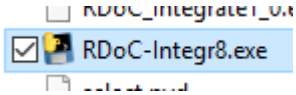
Note: "GUID" stands for "Global Unique Identifier." This is another term used to describe the NIMH assigned ID. To learn more, go to: https://data-archive.nimh.nih.gov/ndct/s/guid/nda-guid.html

To use RDoC-Integr8, you need to first download a zip file containing the application from:

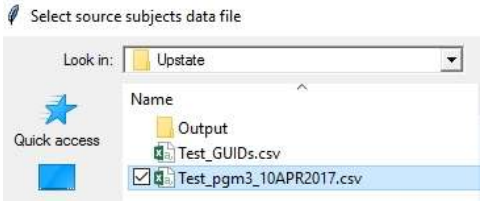
http://pi.cs.oswego.edu/~jmiles3/integr8/

After download, unzip the contents into a directory on your computer.

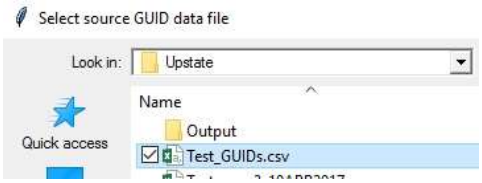
Open the directory (probably named RDoC-Integr8) and run the RDoC-Integr8.exe application



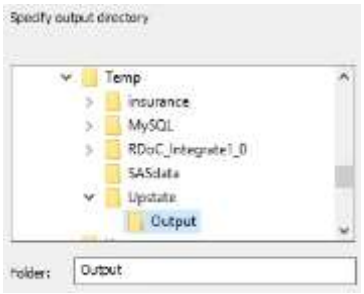
When prompted, choose the source comma separated variables (csv) file that contains the source data for your research subjects:



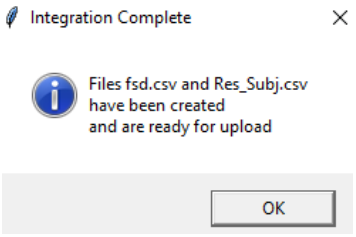
Then, you will be prompted for the csv file containing the research subject NIMH IDs (aka GUIDs):



You will then choose the location of your output:



When the application completes, you will be notified and the output directory will open for your inspection.



Appendix 2: continued

Prerequisites for using RDoC-Integr8.exe

- If you are downloading the zip file from http://pi.cs.oswego.edu/~jmiles3/integr8/, your computer virus protection may be very suspicious of this randomly downloaded executable. Use your best judgement in choosing to override your virus protection.

- The Upstate RDoC research subjects spreadsheet needs to be saved as a 'csv' format.
    - There is no need to manipulate any of the rows or columns since the application will search the file for the necessary variables (columns) and will sort the records (rows). Manual manipulation of the source file is unnecessary and could result in errors.
    - If the wording of the variable (column) titles has changed, the application will not know where to find data. This applies to the variables in Table 1. At this time, changes in title names (for those named in Table 1) will require a simple change to RDoC-Integr8.
    - Remember the name and location of your source csv files so you will be able to find each file when prompted by RDoC-Integr8

- The NIMH ID (GUID) file also needs to be in 'csv' format and needs to have at least 2 columns:
    - One column titled "subjectkey" that contains NIMH IDs (GUIDs)
    - A second column titled "src_subject" that contains Upstate IDs
        - Upstate IDs are expected to end in XXXXYY where XXXX is family ID and YY is the individual ID. (501XXXXYY also works since the 501 is ignored)
        - Column titles are **not** case sensitive.
        - Any characters after the above listed titles are ignored,
            - For example: "src_subject_id" would be read as "src_subject"

- If there are any problems with the application, please address all concerns to: jmiles3@oswego.edu

Appendix 3: Resources

Appendix 3a: A list of useful resources utilized for this project.

Lynda.com: "Programming Foundations" courses.

- If you have little to no programming experience, start with "Real-World Examples."
  - Python is used, but topics are discussed generally
- "Object-Oriented Design" is great for learning how to plan a programming project.
  - Not language specific. Unified Modeling Language (UML) discussed at length.
- "Fundamentals" is interactive experience for programming basics applicable to any language.
  - Javascript is used, activities include sample code templates and various exercises.

Having a book specific to your programming language is useful. Python books I used:

- "Python for Informatics" by Charles Severance
- "Python Crash Course" by Eric Matthes
- "Python Pocket Reference" by Mark Lutz

There are many resources on the Internet. Python resources I used include:

- "Interactive Python." (Try code interactively, also sample algorithms) http://interactivepython.org/courselib/static/pythonds/Introduction/toctree.html
- "PythonProgramming.net." (Useful instructional videos) https://pythonprogramming.net/introduction-to-python-programming/

Code editor:

- Geany: Intuitive code editor. Code can be tested (executed) from within the editor.
  - Free download from: https://www.geany.org/

The most valuable resource you have is the shared experiences of other programmers:

- https://stackoverflow.com

Appendix 3b: List of Certificates of Completion Acquired During the Internship		
Course Name	**Certifier**	**Link**
Biomedical Data or Specimens Only Research	CITI	https://tinyurl.com/MilesBiomedResearchCert
Conflicts of Interest	CITI	https://tinyurl.com/MilesCOICert
Protecting Human Research Participants	NIH	https://tinyurl.com/MilesNIHPHRPCert
Programming Foundations: Real-World Examples	Lynda.com	http://tinyurl.com/MilesFoundationsRW
Programming Foundations: Object-Oriented Design	Lynda.com	http://tinyurl.com/MilesFoundationsOOD
Programming Foundations: Fundamentals	Lynda.com	http://tinyurl.com/MilesFoundationsFnd
Learning Python	Lynda.com	http://tinyurl.com/MilesLearnPython
Python GUI Development with Tkinter	Lynda.com	http://tinyurl.com/MilesPythonTk

Appendix 4: RDoC-Integr8.py program code (5/10/2017)

*FSD-5 variable list:
1. ID
2. Age
3. Gender
4. Race
5. Ethnicity

**FSD-4 "parental" variables:
1. Education
2. Employment
3. Marital status
4. House Income

Pseudocode for creating the fsd csv file is as follows:

For each subject is subjects_list:
- If FamID not equal to previous FamID:
  - Write previous family as csv output
  - Clear buffer
  - Initialize a new family unit (data list with 53 empty variables)
- If subject has Individual ID of 01:
  - Integrate interview date
  - Integrate data from 01 into specified NIMH variables indexes (5 vars*)
  - Integrate GUID value
- If subject has Individual ID of 02 or 03:
  - Integrate parent data into NIMH variable indexes (5 vars* plus 4 parental vars**)
  - Integrate GUID value
- If subject has Individual ID of 04, 05, 06, 0r 07:
  - Integrate sibling data into NIMH variable indexes (5 vars*)
  - Integrate GUID value
- If last family:
  - Write last family as csv output
  - Clear buffer

Creating the Res_Subj csv file is very similar to creating the fsd csv file:

For each subject is subjects_list:

ΔRes_Subj-6 common variables:
Indexes 8, 9, 10, 11, 47, 50
Values on lines 440-447 in code

ΔΔRes_Subj-11 variables list:
1. ID
2. Family ID
3. Interview Date
4. Month Age
5. Gender
6. Race
7. Ethnicity
8. Mental Health? (yes/no)
9. Sample Type
10. Biorepository
11. Biorepository ID

- If FamID not equal to previous FamID:
  - For each member of family in range x:
    - Write subject data to output (if applicable)
  - Clear buffer
  - x = 0 (counter for individuals in the family)
- If subject has Individual ID of 01: x += 1
  - Start a new01 Research Subject (data list with 45 empty vars + 6 common varsΔ)
  - Integrate data for 01 into specified NIMH variables indexes (11 varsΔΔ)
  - Integrate GUID value
- If subject has Individual ID of 02 or 03: x += 1
  - Start a new0x Research Subject (data list with 45 empty vars + 6 common varsΔ)
  - Integrate data for 02 (or 03) into specified indexes (11 varsΔΔ)
  - Integrate GUID value
  - Integrate data for 02 (or 03) into index for new01 # add parent info to "01"
- If subject has Individual ID of 04, 05, 06, 0r 07: x += 1
  - Add new sibling ID to previous sibling (if 04, previous sibling is 01)
  - Start a new0x Research Subject (data list with 45 empty vars + 6 common varsΔ)
  - Integrate data for 04 (or 5,6,7) into specified indexes (11 varsΔΔ)
  - Copy family data from previous sibling [01, 04, etc] into a new0x Research Subject
  - Copy previous sibling [01, 04, etc] as "sibling" for new0x Research Subject
- If last family:
  - For each member of family in range x:
    - Write subject data to output
  - Clear buffer

Appendix 4: RDoC-Integr8.py program code (5/10/2017)

C:\Users\Teva\program\RDoC-Integr8.py
Page 1 of 11 5/10/2017 9:39:43 PM

```
1   #  RDoC-Integr8.py
2   #  Copyright 2017 Joseph Miles <jmiles3@oswego.edu>
3   #
4   #  Version 1.0 of data integration software to be used by
5   #  Upstate Medical for converting study data to a suitable format
6   #  for submitting the data to NIMH.
7   #
8   #  Two files will be created for submission to NIMH:
9   #    "Family Studies Demographics" (fsd.csv)
10  #    "Resurch Subjects Demographics" (Res_Subj.csv)
11  #  In the next release:
12  #    A third file will be created for records that may
13  #    require attention: needs_attention.txt
14  #
15  #  This program is free software; you can redistribute it and/or modify
16  #  it under the terms of the GNU General Public License as published by
17  #  the Free Software Foundation; either version 2 of the License, or
18  #  (at your option) any later version.
19  #
20  #  This program is distributed in the hope that it will be useful,
21  #  but WITHOUT ANY WARRANTY; without even the implied warranty of
22  #  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
23  #  GNU General Public License for more details.
24
25  #  This program expects input of two different csv files
26  #  The program will prompt the user for file names
27  #  First file is the full research subjects spreadsheet in csv format
28  #    Program needs titles in order to find index location of key data
29  #  Second input file is a csv file each person's Upstate ID and their
30  #    unique NIMH ID (aka: GUID)
31  #  Upon completion, the program will open the output directory
32  #    and the two created files will be available for usage.
33  #  The names of the outputted files are: fsd.csv and Res_Subj.csv
34  #  fsd.csv is for "family studies demographics"
35  #  Res_Subj.csv contains every research subject
36  #    both files are created in the exact format specified by NIMH
37
38  # function to open and process research subjects source csv file
39  # input parameter is file name
40  # returns an array of research subjects sorted into family units
41  def process_csv(source):
42      # value transformation (Upstate value to NIMH) for gender
43      recodeGender = {    '1' : "M",
44                          '2' : "F" }
45
46      # value transformation for Race
47      recodeRace = {  '1' : "American Indian/Alaska Native",
48                      '2' : "Asian",
49                      '3' : "Black or African American",
50                      '4' : "Hawaiian or Pacific Islander",
51                      '5' : "White",
52                      '6' : "More than one race",
53                      '7' : "Unknown or not reported" }
54
55      # value transformation for ethnicity
56      recodeEthn= {   '1' : "Hispanic or Latino",
57                      '2' : "Not Hispanic or Latino",
```

- 1 -

Appendix 4: RDoC-Integr8.py program code (5/10/2017), continued

C:\Users\Teva\program\RDoC-Integr8.py
Page 2 of 11 5/10/2017 9:39:43 PM

```
58                      '3' : "No Answer" }
59
60      # define codes for education level, ie. Upstate Code : NIMH Code
61      recodeEduc = {  '1' : '12', '2' : '3',  '3' : '4',  '4' : '5',
62                      '5' : '6',  '6' : '7',  '7' : '9',  '8' : '10' }
63
64      # define codes for employment level, ie. Upstate Code : NIMH Code
65      recodeEmploy = {    '1' : '11',     '2' : '2',  '3' : '1',
66                          '4' : '3',      '5' : '7',  '6' : '6',
67                          '7' : '10',     '8' : '8',  '9' : '9'}
68      subjects = []
69
70      # open research subject source csv file
71      with open(source) as csvfile:
72          readCSV = csv.reader(csvfile, delimiter=',', quotechar='"')
73          for n, row in enumerate(readCSV):
74              # Read header row
75              if n == 0:
76                  for i, cell in enumerate(row):
77                      # remove excess whitespace
78                      cell = cell.strip()
79                      # remove carriage returns in header titles
80                      cell = cell.replace("\n", " ")
81                      # find header titles for data dictionary
82                      # keep track of important titles by storing indexes
83                      if   cell[:8].lower()  == "families": families = i
84                      elif cell[:9].lower()  == "family id": famID = i
85                      elif cell[:7].lower()  == "ind. id": indID = i
86                      elif cell[:11].lower() == "study visit": studyDate = i
87                      elif cell[:13].lower() == "mental health":  mentalHealth = i
88                      elif cell[:13].lower() == "date of birth": dob = i
89                      elif cell[:6].lower()  == "gender": gender = i
90                      elif cell[:17].lower() == "racial category 1": racialCat = i
91                      elif cell[:10].lower() == "ethnic cat": ethnicCat = i
92                      elif cell[:20].lower() == "highest level of edu": education = i
93                      elif cell[:20].lower() == "employment status  1": employment = i
94                      elif cell[:14].lower() == "marital status": marriage = i
95                      elif cell[:20].lower() == "household annual inc": income = i
96                      elif cell[:4].lower()  == "ruid": bloodSam = i
97                      elif cell[:17].lower() == "saliva kit serial": salivaSam = i
98                      elif cell[:21].lower() == "parent agrees to  rep": repo = i
99                      else: continue
100             # now, read all non-header rows and values
101             else:
102                 if row[famID] == '': continue #skip any empty records (blank family id)
103                 # if subject does not give us consent, skip that record
104                 if row[repo].strip() == '2': continue
105                 # composite key for data integration (famID + indID)
106                 compositeKey = row[famID].strip() + row[indID].strip()
107
108                 # Call function to Transform DOB to "month age" for NIMH
109                 monthAge = month_age(row[studyDate].strip(), row[dob].strip())
110
111                 # Transform various Upstate codes to NIMH standard
112                 # NIMH_translate is a function that translates the data
113                 gendr = NIMH_translate(row[gender].strip(), recodeGender)
114                 race = NIMH_translate(row[racialCat].strip(), recodeRace)
```

- 2 -

Appendix 4: RDoC-Integr8.py program code (5/10/2017), continued

C:\Users\Teva\program\RDoC-Integr8.py
Page 3 of 11 5/10/2017 9:39:43 PM

```
115                 ethnicity = NIMH_translate(row[ethnicCat].strip(), recodeEthn)
116                 educationLevel = NIMH_translate(row[education].strip(), recodeEduc)
117                 employmentCode = NIMH_translate(row[employment].strip(), recodeEmploy)
118
119                 # Find biorepository and biorepository ID
120                 biorepID = [] #biorepository data list (ID, location, description)
121                 biorepID = biorepID_find(row[bloodSam].strip(), row[salivaSam].strip())
122
123                 # Create input Data dictionary with extracted variables
124                 subject = [ compositeKey,               #index 0
125                             row[famID].strip(),         #index 1
126                             row[indID].strip(),         #index 2
127                             row[studyDate].strip(),     #index 3
128                             row[mentalHealth].strip(),  #index 4
129                             monthAge,                   #index 5
130                             gendr,                      #index 6
131                             race,                       #index 7
132                             ethnicity,                  #index 8
133                             educationLevel,             #index 9
134                             employmentCode,             #index 10
135                             row[marriage].strip(),      #index 11
136                             row[income].strip(),        #index 12
137                             row[bloodSam].strip(),      #index 13
138                             row[salivaSam].strip(),     #index 14
139                             row[families].strip(),      #index 15
140                             biorepID ]                  #index 16
141                 subjects.append(subject) # append next subject
142     subjects.sort() # sort subjects by family then ind id (compositeKey)
143     return subjects
144
145 # function to transform birthdate to "month age"
146 # input parameters are visit date and birth date
147 # returns "month age"
148 def month_age(visitDate, birthDate):
149 # for normalizing months to 30 days (interview age is "month age")
150     # Establish variables for month_age calculation
151     vList = [int(x) for x in visitDate.split("/")]
152     bList = [int(y) for y in birthDate.split("/")]
153
154     # Check for four digit year, convert if necessary
155     if len(str(vList[2])) <= 2:
156         if len(str(vList[2])) == 2:
157             vList[2] = int("20%s" % str(vList[2]))
158         if len(str(vList[2])) == 1:
159             vList[2] = int("200%s" % str(vList[2]))
160     if len(str(bList[2])) <= 2:
161         visitYear2 = str(vList[2])
162         if bList[2] < int(visitYear2[-2:]):
163             if len(str(bList[2])) == 2:
164                 bList[2] = int("20%s" % str(bList[2]))
165             if len(str(bList[2])) == 1:
166                 bList[2] = int("200%s" % str(bList[2]))
167         else: bList[2] = int("19%s" % str(bList[2]))
168
169     vDate = date(vList[2], vList[0], vList[1])
170     bDate = date(bList[2], bList[0], bList[1])
171
```

- 3 -

Appendix 4: RDoC-Integr8.py program code (5/10/2017), continued

C:\Users\Teva\program\RDoC-Integr8.py
Page 4 of 11 5/10/2017 9:39:43 PM

```
172         # Corrective math for partial months
173         if vList[0] == 1:
174             testM = 12
175             testY = vList[2] - 1
176         else:
177             testM = vList[0] - 1
178             testY = vList[2]
179
180         # correction for birthdate days beyond 28 in a month
181         dCorrect = 0
182         if bList[1] > 28:
183             dCorrect = bList[1] - 28
184             testD = 28
185         else:
186             testD = bList[1]
187
188         # count days from month before visit date
189         dayRange = vDate - date(testY, testM, testD)
190         days = int(dayRange.days)
191
192         # correction for February and 31 day months
193         shortMonth = 2
194         longMonth = [1, 3, 5, 7, 8, 10, 12]
195         if testM == shortMonth: days = days + 2
196         if testM in longMonth: days = days - 1
197         # Subtract dCorrect for birthdays > 28th day (it just works!)
198         days = days - dCorrect
199         MonthAge = (12 * (vList[2] - bList[2])) + (vList[0] - bList[0])
200
201         # correction to add 1 month for > 15 days within month
202         if days > 45:
203             MonthAge = MonthAge + 1
204         # correction to subtract 1 month for < 15 days between months
205         elif days <= 15:
206             MonthAge = MonthAge - 1
207         return str(MonthAge)
208
209     # function to transform Upstate value to NIMH value
210     # input parameters are Upstate value and array for transformation
211     # returns NIMH value (or Upstate value if there is no equivalent
212     def NIMH_translate(code, dic): # Translate Upstate code for NIMH
213         if code in dic:     return dic[code]
214         else:               return code
215
216     # function to find biorepository data
217     # input parameters are whole blood id and saliva ID (one is usually '')
218     # returns bio specimen type and biorepository name
219     def biorepID_find(blood, saliva): # Find biorepository ID
220         if blood != '' and blood != '0':
221             return [blood, "RUCDR", "whole blood"]
222         elif saliva != '' and saliva != '0':
223             return [saliva,
224                     "SUNY Upstate Medical University PsychGENe Laboratory",
225                     "saliva"]
226         else:
227             return ["unidentified", "0", "0"]
228
```

- 4 -

Appendix 4: RDoC-Integr8.py program code (5/10/2017), continued

C:\Users\Teva\program\RDoC-Integr8.py
Page 5 of 11 5/10/2017 9:39:43 PM

```
229  # function to open GUID csv file and read contents
230  # creates dictionary of NIMH GUID with compositeKey as key
231  # input parameter is file name (GUID)
232  # returns GUID data for all research subjects
233  def process_GUID(source):
234      GUID = {}
235      with open(source) as csvfile:
236          readCSV = csv.reader(csvfile, delimiter=',', quotechar='"')
237          for n, row in enumerate(readCSV):
238              if n == 0: #Header row
239                  for i, cell in enumerate(row):
240                      cell = cell.strip() #remove excess whitespace
241                      #find header titles for GUID dictionary
242                      if   cell[:11].lower()  == "src_subject": famGUID = i
243  #                   elif cell[:7].lower()  == "ind. id": indGUID = i
244                      elif cell[:10].lower()  == "subjectkey": subkey = i
245                      else: continue
246              else: # All non-header rows
247                  # Define composite key for data integration
248                  # Skip any empty records (blank family id)
249                  if row[famGUID] == '': continue
250                  compositeKey = row[famGUID].strip()
251                  compositeKey = compositeKey[-6:] #only need last 6 digits
252
253                  # Add composite key and GUID to array
254                  GUID[compositeKey] = row[subkey].strip()
255      return GUID
256
257  # function to integrate data for both output files
258  # input parameters are output directory, list of subjects, and GUIDs
259  # no return, integrated data is written to output files
260  def output_create(dir, subjectlist, Guid):
261      fsdName = "fsd" # maybe add prompt for fsd file name?
262      resSubName = "Res_Subj" # maybe add prompt for Res_Subj file also?
263      # create a new file, archive any older file
264      create_file(dir, fsdName)
265      create_file(dir, resSubName)
266      with open("%s%s.csv" % (dir, fsdName), 'a') as fsdFile:
267          # write the header of the fsd file as defined by NIMH
268          fsdFile.write("fsd,1\n")
269          # write variable titles of fsd file as defined by NIMH
270          fsdFile.write("subjectkey,src_subject_id,interview_date,"
271                  "interview_age,gender,race,ethnicity,subjectkey_father,"
272                  "src_father_id,ques_agebf,ques_genderbf,"
273                  "bio_father_race,bio_father_ethnicity,et2b,"
274                  "dem_08b,fa_maritalstatus,fa_householdincome,"
275                  "subjectkey_mother,src_mother_id,ques_agebm,"
276                  "ques_genderbm,bio_mother_race,bio_mother_ethnicity,"
277                  "et2a,dem_08a,mo_maritalstatus,mo_householdincome,"
278                  "subjectkey_sibling1,src_sibling1_id,ques_age1,"
279                  "ques_gender1,sib1_race,sib1_ethnicity,"
280                  "subjectkey_sibling2,src_sibling2_id,ques_age2,"
281                  "ques_gender2,sib2_race,sib2_ethnicity,"
282                  "subjectkey_sibling3,src_sibling3_id,ques_age3,"
283                  "ques_gender3,sib3_race,sib3_ethnicity,"
284                  "subjectkey_sibling4,src_sibling4_id,ques_age4,"
285                  "ques_gender4,sib4_race,sib4_ethnicity,"
```

- 5 -

Appendix 4: RDoC-Integr8.py program code (5/10/2017), continued

C:\Users\Teva\program\RDoC-Integr8.py
Page 6 of 11 5/10/2017 9:39:43 PM

```
286                 "mother_other_race,father_other_race\n")
287
288     with open("%s%s.csv" % (dir, resSubName), 'a') as resSubFile:
289         # write the header and titles of the Res_Subj file
290         resSubFile.write("ndar_subject,1\n")
291         resSubFile.write("ndar_subjectkey,src_subject_id,"
292             "interview_date,interview_age,gender,race,ethnic_group,"
293             "phenotype,phenotype_description,twins_study,sibling_study,"
294             "family_study,family_user_def_id,subjectkey_mother,"
295             "src_mother_id,subjectkey_father,src_father_id,"
296             "subjectkey_sibling1,src_sibling1_id,sibling_type1,"
297             "subjectkey_sibling2,src_sibling2_id,sibling_type2,"
298             "subjectkey_sibling3,src_sibling3_id,sibling_type3,"
299             "subjectkey_sibling4,src_sibling4_id,sibling_type4,"
300             "zygosity,sample_taken,sample_id_original,"
301             "sample_description,biorepository_name,"
302             "sample_id_biorepository,patient_id_biorepository,"
303             "cell_id_original,cell_id_biorepository,agre_subject_id,"
304             "sfari_subject_id,cpea_site,cpea_id,blood_id,adi_dx,"
305             "ados_dx,agp_family_id,agp_subject_id,site,comments_misc,"
306             "week,study\n")
307
308     familyID = "0"  # for first pass, indicates first family
309     parentList = ["02", "03"]
310     sibList = ["04", "05", "06", "07"]
311     for sub in subjectlist:  # subjects will be read in order
312         if sub[1] != familyID: # new family ID, then write current data
313             if familyID != "0":  # first time through, skip
314                 # write to both files when a new family is started
315                 write_output(fsdData, dir, fsdName)
316                 write_Res_Subj(resSubData, dir, resSubName, familyCount)
317             # clear previous data for next family
318             fsdData = [''] * 53  # fsd template has 53 attributes
319             resSubData = [''] * 7 # template for up to 7 subs per family
320             familyCount = 0 # count non-proband people in the family
321             sibCount = 0 # count number of non-proband siblings
322             familyID = sub[1][:] # set new family ID
323             if  sub[2] == "01":  # primary (proband) subject
324                 fsdIndex = [0, 1, 3, 4, 5, 6] # index list for fsd
325                 # update fsd record with proband info
326                 fsdData = demog_update(fsdData,sub,Guid,fsdIndex)
327                 fsdData[2] = sub[3][:]  # Record interview date
328                 # if index 6 is "No Answer", change to "Unknown" per NIMH
329                 if fsdData[6] == "No Answer":
330                     fsdData[6] = "Unknown"
331                 # create a research subj record for Res_Subj file
332                 resSubData[familyCount] = resSub_initialize()
333                 resSubData[familyCount] = resSub_update(
334                                 resSubData[familyCount],sub,Guid)
335             else: # This would be hit if there was no "01" subject
336                 print("%s Is missing data" % familyID)
337         else:
338             # Is the next subject a father (02) or mother (03)?
339             if sub[2] in parentList: # parentList is indID 02 or 03
340                 familyCount += 1 # add 1 to total family members
341                 if  sub[2] == "02": # This is the father
342                     fsdIndex = [7, 8, 9, 10, 11, 12]
```

Appendix 4: RDoC-Integr8.py program code (5/10/2017), continued

C:\Users\Teva\program\RDoC-Integr8.py
Page 7 of 11 5/10/2017 9:39:43 PM

```
343                         fsdParent = [13, 14, 15, 16]
344                         resSubIndex = [15, 16]
345                     elif sub[2] == "03": # This is the mother
346                         fsdIndex = [17, 18, 19, 20, 21, 22]
347                         fsdParent = [23, 24, 25, 26]
348                         resSubIndex = [13, 14]
349                     # update info for fsd file
350                     fsdData = demog_update(fsdData,sub,Guid,fsdIndex)
351                     fsdData = parent_update(fsdData,sub,fsdParent)
352                     # update info for Res_Subj file
353                     resSubData[0] = resSub_add(resSubData[0],sub[0],Guid,
354                                                resSubIndex)
355                     # create a Res_Subj line for the parent
356                     resSubData[familyCount] = resSub_initialize()
357                     resSubData[familyCount] = resSub_update(
358                                     resSubData[familyCount], sub, Guid)
359
360                 # Is the next subject a sibling 04, 05, 06 or 07?
361                 elif sub[2] in sibList: # sibList is 04, 05, 06, or 07
362                     # note sibling 0 is the proband (primary subject)
363                     sibCount += 1 # add one to count each sibling
364                     familyCount += 1 # add one to family count
365                     if sibCount == 1: # This is sibling 1
366                         fsdIndex = [27, 28, 29, 30, 31, 32]
367                         resSubIndex = [17, 18]
368                     elif sibCount == 2: # This is sibling 2
369                         fsdIndex = [33, 34, 35, 36, 37, 38]
370                         resSubIndex = [20, 21]
371                     elif sibCount == 3: # This is sibling 3
372                         fsdIndex = [39, 40, 41, 42, 43, 44]
373                         resSubIndex = [23, 24]
374                     else:               # This is sibling 4
375                         fsdIndex = [45, 46, 47, 48, 49, 50]
376                         resSubIndex = [26, 27]
377                     # update the fsd file information
378                     fsdData = demog_update(fsdData,sub,Guid,fsdIndex)
379                     # create a new Res_Subj line for the sibling
380                     resSubData[familyCount] = resSub_initialize()
381                     resSubData[familyCount] = resSub_update(
382                                     resSubData[familyCount], sub, Guid )
383                     if sibCount == 1: # first sibling needs proband info
384                         sibIndx = 0   # 0 is proband
385                     else: sibIndx = familyCount - 1 # get from previous sib
386                     # assume sibling has same parent information
387                     # exceptions will need to be handled manually
388                     resSubData[familyCount] = resSub_copy(
389                             resSubData[sibIndx][:], resSubData[familyCount])
390                     # add sibling data to previous siblings
391                     for sib in range(sibCount):
392                         if sib == 0:
393                             sibX = 0
394                         else: sibX = familyCount - sib
395                         resSubData[sibX] = resSub_add(resSubData[sibX],
396                                                       sub[0],Guid,resSubIndex)
397                     # add previous sibling info as "sibling" of new sibling
398                     resSubData[familyCount] = resSub_sibadd(
399                             resSubData[familyCount],
```

Appendix 4: RDoC-Integr8.py program code (5/10/2017), continued


```
400                         resSubData[sibIndx][0],
401                         resSubData[sibIndx][1], resSubIndex)
402             else: continue # Only space for 4 siblings per NIMH
403         # Last time through, this will write the buffer data
404         write_output(fsdData, dir, fsdName)
405         write_Res_Subj(resSubData, dir, resSubName, familyCount)
406         # output files created, confirm with user and open output directory
407         complete = messagebox.showinfo(title="Integration Complete",
408                 message = 'Files fsd.csv and Res_Subj.csv\nhave been created\nand are ready for upload')
409         direct = dir.replace('/', '\\')
410         os.startfile("%s" % direct)
411
412     # function to input patient demographics into fsd.csv
413     # input parametes are current fsd record, subject data, GUID and index list
414     # return the updated fsd record
415     def demog_update(fsd, sub, GUid, indx):
416         subjectID = "501" + sub[0]
417         fsd[indx[0]] = GUid[sub[0]]      #GUID
418         fsd[indx[1]] = subjectID         #subject ID
419         fsd[indx[2]] = sub[5]            #interview age (in months)
420         fsd[indx[3]] = sub[6]            #gender
421         fsd[indx[4]] = sub[7]            #race
422         fsd[indx[5]] = sub[8]            #ethnicity
423         return fsd
424
425     # function to input extra parental data (fsd.csv)
426     # input parameters: fsd record, subject data and index list
427     # return the updated fsd record
428     def parent_update(fsd, sub, indx):
429         fsd[indx[0]] = sub[9]            #level of education
430         fsd[indx[1]] = sub[10]           #employment status
431         fsd[indx[2]] = sub[11]           #marital status
432         fsd[indx[3]] = sub[12]           #household annual income
433         return fsd
434
435     # function to initialize a new Res_Subj record
436     # no parameters, return the new Res_Subj record
437     def resSub_initialize():
438         tempList = [''] * 51
439         # these values are static for all research subjects at Upstate
440         pheno0 = '"Ever seen by a health professional for any mental '
441         pheno1 = 'health/behavioral issues? (1=Yes,2=No)"'
442         tempList[8]  = "%s%s" % (pheno0, pheno1)
443         tempList[9]  = "No"
444         tempList[10] = "No"
445         tempList[11] = "Yes"
446         tempList[47] = "501"
447         tempList[50] = "RDoC"
448         return tempList
449
450     # function to input patient demographics for Res_Subj.csv
451     # input parameters are research subject record, subject data, GUID data
452     # returns updated research subject record
453     def resSub_update(resSub, sub, GUid):
454         subjectID = "501" + sub[0]
455         subjectIDdash = "501-%s-%s" % (sub[1], sub[2])
```

Appendix 4: RDoC-Integr8.py program code (5/10/2017), continued


```
456         resSub[0]  = GUid[sub[0]]   #GUID
457         resSub[1]  = subjectID      #subject ID
458         resSub[2]  = sub[3]         #interview date
459         resSub[3]  = sub[5]         #interview age (in months)
460         resSub[4]  = sub[6]         #gender
461         resSub[5]  = sub[7]         #race
462         resSub[6]  = sub[8]         #ethnicity
463         resSub[7]  = sub[4]         #mental health issue, 1 = yes
464         resSub[12] = sub[1]         #family ID
465         if sub[16][0] != "unidentified":
466             resSub[30] = "Yes"
467             resSub[31] = subjectIDdash
468             resSub[32] = sub[16][2]
469             resSub[33] = sub[16][1]
470             resSub[34] = sub[16][0]
471             resSub[35] = subjectIDdash
472         else:
473             resSub[30] = "No"
474         return resSub
475
476     # function to copy subject data from a previous family member
477     # input parameters: subject record (previous) and subject record (next)
478     # ruturn updated subject record (next)
479     def resSub_copy(sub0, sub1):
480         for i in range(8,28):
481             sub1[i] = sub0[i][:]
482         return sub1
483
484     # function to add a family member to research subject record
485     # input parameters: subject record, key of family member, GUID list, index
486     # return subject record with family member data added
487     def resSub_add(resSub, key, GUid, indx):
488         resSub[indx[0]] = GUid[key]
489         resSub[indx[1]] = "501%s" % key
490         return resSub
491     # function to add sibling data to research subject record
492     # input parameters: subject record, GUID list, Upstate ID numbr, index
493     # return subject record with sibling data added
494     def resSub_sibadd(resSub, GUid, ID, indx):
495         resSub[indx[0]] = GUid
496         resSub[indx[1]] = ID
497         return resSub
498
499     # function to backup any old information and create a new file
500     # input parameters: output directory and filename to create
501     # nothing to return. Older csv files are archived (to avoid overwriting)
502     def create_file(outdir, filename):
503         fileExt = 0
504         # if file already exists, archive the previous file
505         while os.path.exists("%s%s.csv" % (outdir, filename)):
506             while os.path.exists("%s%s%d.csv" % (outdir, filename, fileExt)):
507                 fileExt += 1
508             while True:
509                 try:
510                     os.rename("%s%s.csv" % (outdir, filename),
511                               "%s%s%d.csv" % (outdir, filename, fileExt))
512                     break
```

Appendix 4: RDoC-Integr8.py program code (5/10/2017), continued


```
513                 except:
514                     # if output file exists and is open, warn the user
515                     messagebox.showwarning(
516                             title="File %s.csv in use" % filename,
517                             message = 'Please close the file: %s%s.csv so I can continue'
518                             % (outdir, filename) )
519         # create the output file
520         open("%s%s.csv" % (outdir, filename), 'a').close()
521
522     # functions to write output for Res_Subj and fsd.csv
523     # Res_Subj parameters: records, output directory, filename, index (how many)
524     def write_Res_Subj(data, dir, fname, indx):
525         for i in range(indx + 1):
526             write_output(data[i], dir, fname)
527     # fsd parameters: family record, output directory, filename
528     def write_output(data, dir, fname):
529         with open("%s%s.csv" % (dir, fname), 'a') as File:
530             File.write("%s\n" % ','.join(data))
531
532     # function to prompt user for a source file
533     # input parameter: which file
534     # returns name of the source file
535     def source_file(type):
536         okcancel = messagebox.askokcancel(title="Specify Input", message =
537                 'Choose the \"%s\" source file when prompted:' % type )
538         if okcancel == False:
539             exit()
540         while True:
541             try:
542                 sourcefile = filedialog.askopenfile(
543                     title = "Select source %s data file" % type,
544                     filetypes = (("csv files","*.csv"),("all files","*.*")))
545                 return sourcefile.name
546             except:
547                 okcan2 = messagebox.askokcancel(
548                     title="No Source File",
549                     message = 'No source "%s" file is found.\nTry again?' % type )
550                 if okcan2 == False: exit()
551
552     # function to prompt user for a output directory
553     # no input parameter, returns name of the output directory
554     def working_dir():
555         messagebox.askokcancel(title="Choose a directory", message =
556                 'Choose the directory for your output when prompted' )
557         workingdir = filedialog.askdirectory(
558                 title = "Specify output directory" )
559         return "%s\\" % workingdir
560
561     """
562     Place holder for future application update (error log)
563     def error-log(dir, data):
564         with open("%s%s.csv" % (dir, 'output-log'), 'a') as File:
565             File.write("%s\n" % ','.join(data))
566     """
567
568     # this is the main program, args is for system errors
569     def main(args):
```

Appendix 4: RDoC-Integr8.py program code (5/10/2017), continued

C:\Users\Teva\program\RDoC-Integr8.py
Page 11 of 11 5/10/2017 9:39:43 PM

```
570
571         Tk().withdraw() # no Tk GUI at this time - future update
572
573         sourcecsv = source_file("subjects") # ask for source subjects csv
574         sourceGUID = source_file("GUID")    # ask for source GUID csv
575         workingDir = working_dir()          # ask for output directory
576
577         #define list for all subjects
578         subjects = []
579         subjects = process_csv(sourcecsv) # create subjects list
580         guid = {}
581         guid = process_GUID(sourceGUID)   # create array for GUID
582
583         output_create(workingDir, subjects, guid) # integrate and write output
584
585     if __name__ == "__main__":
586         # libraries needed for this application
587         import csv
588         import os
589         import sys
590         from datetime import date
591         from tkinter import Tk
592         from tkinter import filedialog
593         from tkinter import messagebox
594         sys.exit(main(sys.argv))
595
596
```

- 11 -

Appendix 5: Project presentation (slides 1-6),
http://pi.cs.oswego.edu/~jmiles3/bhi/ETL-Integrate.pdf

Data Extraction, Transformation, and Integration for Loading Data into a Psychiatric Research Data Warehouse

Data Analytics: Biomedical and Health Informatics 509

Presentation by Joseph Miles

Introduction

"The decentralized nature of our scientific communities and healthcare systems has created a sea of valuable but incompatible electronic databases." (Sujansky, 2001)

Data Warehouse: (Schmidt and Prado, 2014)

1. Accuracy
2. Reliability
3. Consistency
4. Accessibility

Also: completeness, and interpretability

Sujansky, W. (2001). Heterogeneous database integration in biomedicine. *Journal Of Biomedical Informatics*. 34(4): 285-298. Doi:10.1006/jbin.2001.1024

Schmidt, SO; Prado, EPV. (2014) IT Architecture and Information Quality in Data Warehouse and Business Intelligence Environments. *IGI Global*. 6: 121-127. DOI: 10.4018/978-1-4666-4892-0.ch06

Integration

Heterogeneous methods of labeling and storing data confound data integration

- Different labels, same meaning
- Same labels, different meaning
- Same label, same meaning but different semantics

ETL: Extraction, Transformation, and Loading

- To maintain the integrity of the data, accurate ETL is of highest importance (Schmidt and Prado, 2014)

Schmidt, SO; Prado, EPV. (2014) IT Architecture and Information Quality in Data Warehouse and Business Intelligence Environments. *IGI Global*. 6: 121-127. DOI: 10.4018/978-1-4666-4892-0.ch06

Heterogeneous Grouping

Patients with the same diagnosis do not always have the same problems

- Depression:
  - Sad
  - Anxious
  - Empty
  - Hopeless
  - Irritable
  - Feelings of guilt, worthlessness, or helplessness
  - Decreased energy or fatigue
  - Feeling restless or having trouble sitting still
  - Difficulty sleeping
  - Oversleeping

NIMH: Depression. https://www.nimh.nih.gov/health/topics/depression/index.shtml

RDoC (Research Domain Criteria)

We tend to treat patients by diagnosis rather than treating the problem(s) that established the original diagnosis.

RDoC

- RDoC *"integrates many levels of information (from genomics to self-report) to better understand basic dimensions of functioning underlying the full range of human behavior from normal to abnormal."* (NIMH, RDoC)
- Beyond categorization:
  - *"The best time to address a mental illness is before the appearance of symptoms that disrupt daily life."* https://www.nimh.nih.gov/about/strategic-planning-reports/strategic-objective-2.shtml

NIMH. Research Domain Criteria (RDoC). https://www.nimh.nih.gov/research-priorities/rdoc/index.shtml

RDoC Studies

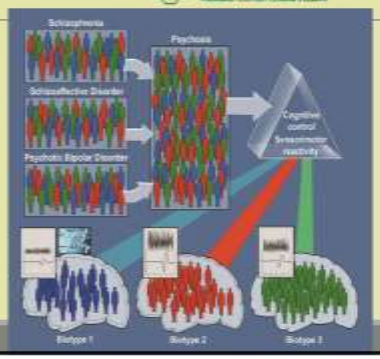
Accumulate data:

1. Genomics and inheritance
   Currently known candidate genes in the psychiatric "Positive Valence System" are only loosely correlated with observed behaviors (phenotype) [r^2 values ranging from 0.181 to 0.233, Hess et al, 2016].
2. Environmental Considerations
   - Highest level of education
   - Employment status
   - Household income
3. Self-report of symptomology
   - Discover phenotype – genotype linkage
   - Assess differences in clinical presentation at different patient ages
     - Longitudinal study

Hess, JL; Kawaguchi, DM; Wagner, KE; Faraone, SV; Glatt, SJ. (2016). The influence of genes on "positive valence systems" constructs: A systematic review. *American Journal of Medical Genetics Part B: Neuropsychiatric Genetics*. 171(1), 90-110.

Appendix 5: Project presentation (slides 7-12),
http://pi.cs.oswego.edu/~jmiles3/bhi/ETL-Integrate.pdf

RDoC Studies



Upstate RDoC Family Study

SUNY Upstate Medical University has been evaluating constructs from the Positive Valence System

- Research subject (proband) is a child with a "Mental Health Issue."
- Family Study:
  - Also examine biological parent(s)
  - Also examine biological siblings (up to 4 siblings)
- Each research subject (proband and family members) is documented in a spreadsheet with 96 variables
  - As of 10/28/2016, there were 465 families and 1362 total research subjects
  - 1334 subjects either donated whole blood or saliva for genetic testing

What are we creating, short term?

NIMH requires RDoC research sites to submit cumulative data to the data warehouse every 6 months (> 80 hours?!?)

Upstate is submitting 2 files:

1. Family Studies Demographics (53 variables per record)
   - A comma separated variables (csv) file with each record representing a family
2. Research Subjects Demographics (51 variables per record)
   - A csv file with each records representing a research subject
   - Research subject data also requests parent/sibling ID data

ElementName	DataType	Size	Required	ElementDescription	ValueRange
subjectkey	GUID		Required	The NDAR Global Unique Identifier (GUID) for research subject	NDAR*
src_subject_id	String	20	Required	Subject ID how it's defined in lab/project	
interview_date	Date		Required	Date on which the interview/genetic test/sampling/imaging was completed. MM/DD/YYYY	
interview_age	Integer		Required	Age in months at the time of the interview/test/sampling/imaging.	0 :: 1260
gender	String	20	Required	Sex of the subject	M;F
race	String	30	Recommended	Race of study subject	American Indian/Alaska Native; Asian; Hawaiian or Pacific Islander; Black or African American; White; More than one race; Unknown or not reported
ethnicity	String	30	Recommended	Ethnicity of participant	Hispanic or Latino; Not Hispanic or Latino; Unknown
subjectkey_father	GUID		Recommended	The NDAR Global Unique Identifier (GUID) for subject's biological father	NDAR*
src_father_id	String	100	Recommended	site specific father ID	

Data dictionary, Family Studies Demographics

0. subjectkey	17. subjectkey_mother	35. ques_age2
1. src_subject_id	18. src_mother_id	36. ques_gender2
2. interview_date	19. ques_ageben	37. sib2_race
3. interview_age	20. ques_genderbm	38. sib2_ethnicity
4. gender	21. bio_mother_race	39. subjectkey_sibling3
5. race	22. bio_mother_ethnicity	40. src_sibling3_id
6. ethnicity	23. et2a	41. ques_age3
7. subjectkey_father	24. dem_08a	42. ques_gender3
8. src_father_id	25. mo_maritalstatus	43. sib3_race
9. ques_agebf	26. mo_householdincome	44. sib3_ethnicity
10. ques_genderbf	27. subjectkey_sibling1	45. subjectkey_sibling4
11. bio_father_race	28. src_sibling1_id	46. src_sibling4_id
12. bio_father_ethnicity	29. ques_age1	47. ques_age4
13. et2b	30. ques_gender1	48. ques_gender4
14. dem_08b	31. sib1_race	49. sib4_race
15. fa_maritalstatus	32. sib1_ethnicity	50. sib4_ethnicity
16. fa_householdincome	33. subjectkey_sibling2	51. mother_other_race
	34. src_sibling2_id	52. father_other_race

Data dictionary, Research Subject Demographics

0. ndar_subjectkey	17. subjectkey_sibling1	34. sample_id_biorepository
1. src_subject_id	18. src_sibling1_id	35. patient_id_biorepository
2. interview_date	19. sibling_type1	36. cell_id_original
3. interview_age	20. subjectkey_sibling2	37. cell_id_biorepository
4. gender	21. src_sibling2_id	38. agre_subject_id
5. race	22. sibling_type2	39. sfari_subject_id
6. ethnic_group	23. subjectkey_sibling3	40. cpea_site
7. phenotype	24. src_sibling3_id	41. cpea_id
8. phenotype_description	25. sibling_type3	42. blood_id
9. twins_study	26. subjectkey_sibling4	43. adi_dx
10. sibling_study	27. src_sibling4_id	44. ados_dx
11. family_study	28. sibling_type4	45. agp_family_id
12. family_user_def_id	29. zygosity	46. agp_subject_id
13. subjectkey_mother	30. sample_taken	47. site
14. src_mother_id	31. sample_id_original	48. comments_misc
15. subjectkey_father	32. sample_description	49. week
16. src_father_id	33. biorepository_name	50. study

Appendix 5: Project presentation (slides 13-18), http://pi.cs.oswego.edu/~jmiles3/bhi/ETL-Integrate.pdf

What to Integrate?

Each family has up to 7 people = up to 672 variables to extract from for each family!

- 2 variables are "family ID" and "individual ID" which were used as a primary key and for generating the Upstate ID
- 12 other variables for each individual were also needed.

A second file contains a deidentified NIMH ID code for each subject.

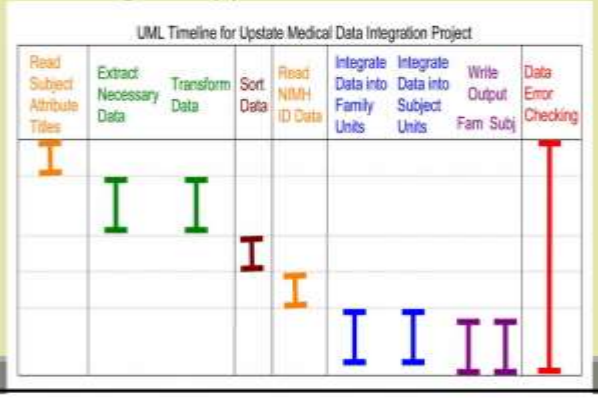
- NIMH ID is called a GUID [Global Unique Identifier] and is required for every research subject in both upload files.

Simplified Data Dictionary

Table 1
Table of input variables needed for the data dictionary

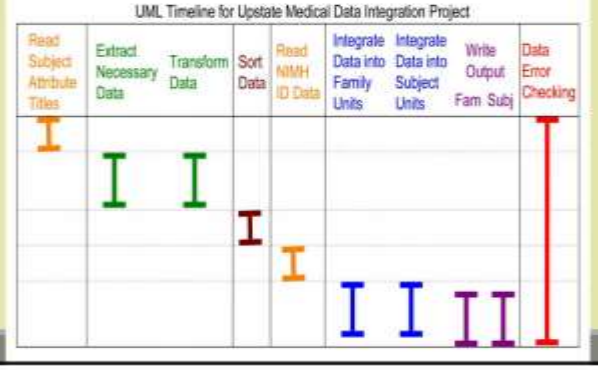
Variable Name	Transformation necessary: Convert data to NIMH format (*Yes* or *No*)
Family ID •(Part of composite key)	No
Individual ID •(Part of composite key)	No
Study Visit Date	No
Mental Health	No
Date of Birth	Yes
Gender	Yes
Racial Category	Yes
Ethnic Category	Yes
Highest Level of Education	Yes
Employment Status	Yes
Marital Status	No
Household Annual Income	No
Blood Sample ID	Yes
Saliva Sample ID	Yes

Planning the application



Read Variable Titles

Planning the application: continued



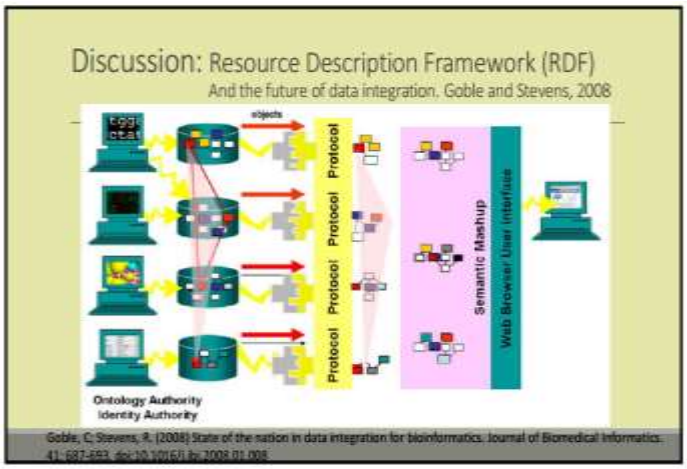
Discussion: Data integration for bioinformatics

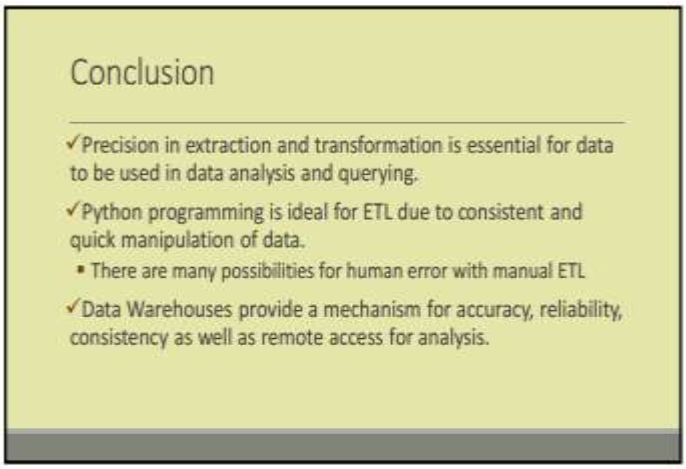
Goble and Stevens, 2008

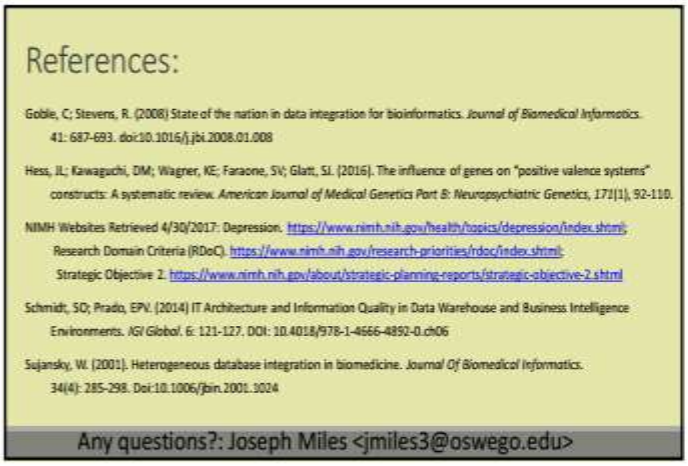
- ❑ Need common, shared identities and names
- ❑ Need shared semantics
- ❑ Need shared access mechanisms
- ❑ Need to adhere to standards
- ❑ Need to explicitly state collection policies and governance
- ❑ Need to balance curation with data usability

Goble, C; Stevens, R. (2008) State of the nation in data integration for bioinformatics. Journal of Biomedical Informatics. 41: 687-693. doi:10.1016/j.jbi.2008.01.008

Appendix 5: Project presentation (slides 19-21),
http://pi.cs.oswego.edu/~jmiles3/bhi/ETL-Integrate.pdf







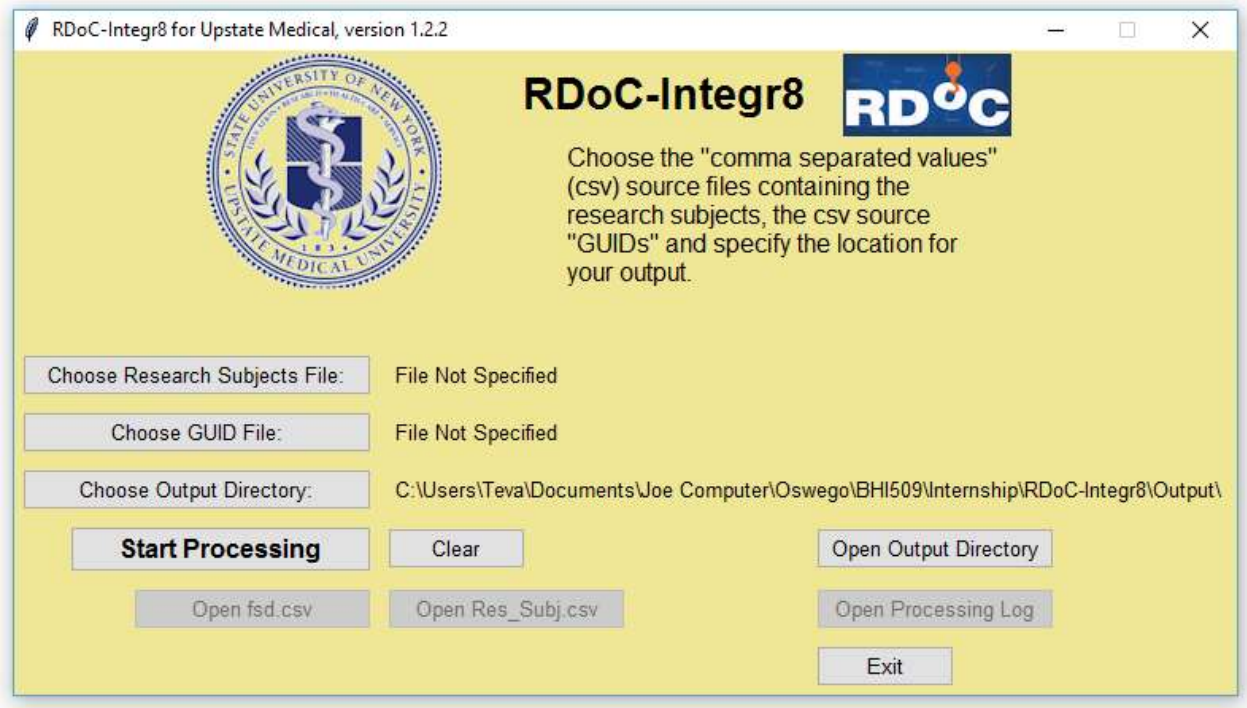
Full presentation downloadable from: http://pi.cs.oswego.edu/~jmiles3/bhi/ETL-Integrate.pdf

A video presentation of the above slides can be viewed on YouTube: https://youtu.be/v1EA4LqbpMU

Addendum: Project performance report and update from July 3, 2017 (page 1)

Since May 11, 2017, two major and two minor updates were made for the RDoC-Integr8 application.

1. Major update one was to generate a feedback file that gives the user information about what was processed each time the application is run. RDoC-Integr8 looks for any potential problems in the source data and reports that back to the user. The application also documents the names and location of all source data for future reproducibility of results.
2. Major update two created a graphical interface for RDoC-Integr8 as seen in the following figure:



After implementing the two major updates to RDoC-Integr8, some idiosyncrasies in the source data were discovered that led to two other minor updates:

i. Minor update one allowed the program to appropriately process research subjects that are uniquely identified in the source data with alternate subject suffixes. Typically, subject IDs have a suffix range of 01 to 07 to identify each subject's role in a nuclear family (02 = father, 03 = mother, etc), but sometimes alternate suffixes are used when the research subject doesn't fit into a predefined family role.

ii. Minor update two allows RDoC-Integr8 to process subjects that are lacking characters in their subject IDs. Due to various ways of recording the source data, leading zeroes in ID fields are sometimes removed by the program recording the data. Subject IDs are 9-digit numbers and are broken into 3 parts: a leading 501 to indicate an Upstate Medical subject, followed by a 4-digit code to indicate the nuclear family ID, and a 2-digit suffix as described in "minor update one" above. RDoC-Integr8 will now normalize all subjects to the 9-digit standard by adding preceding zeroes. For example, subject 2 of family 1 would be processed as 501000102 whether or not preceding zeroes are indicated in the source data.

Addendum: Project performance report and update from July 3, 2017 (page 2: performance results)

RDoC-Integr8 version 1.2.2 was used on July 3, 2017 to process all Upstate Medical University RDoC Research subjects. In many cases, research subjects choose not to give consent for sharing their data with NIMH. Since part of the feedback from RDoC-Integr8 is to notify the user of all unprocessed research subjects, including those that did not give consent for sharing data, it was the choice of the RDoC team to clean the source data of all individuals that had not provided consent. As a result of updates to the application and cleaning the source data, every research subject was processed by RDoC-Integr8: 100% success as seen in the following excerpt from the feedback from RDoC-Integr8.

```
1761 rows read
of which, 0 rows did not give consent for NIMH upload
and 0 records were empty.
Total of 1761 records processed
End subject source data processing

Begin GUID source data processing:

Column numbers of found titles *case insensitive, spaces ignored
Title text: Column Number
src_subject: 2
subjectkey: 1

1763 GUIDs processed
End GUID source data processing



BEGIN OUTPUT CREATION:

C:\Users\RoeC\Desktop\RDoC-Integr8\RDoC-Integr8\Output\fsd.csv created

C:\Users\RoeC\Desktop\RDoC-Integr8\RDoC-Integr8\Output\Res_Subj.csv created

***Family 0074 is lacking a 01 proband: 007402
***Family 0148 is lacking a 01 proband: 014802
^^^Subject 060742 has an individual id outside of scope 01-07: 42
***Family 0648 is lacking a 01 proband: 064803

1761 of 1761 records processed
"!!!" 0 record(s) were skipped (subject will NOT be included in any output)
"***" 3 family(ies) were skipped (not included in fsd.csv, subjects will be in Res_Subj.csv file only)
"^^^" 1 subject(s) have ind.id outside range of 01 to 07 (subject will be in Res_Subj.csv file only)
Integration complete, output written to C:\Users\RoeC\Desktop\RDoC-Integr8\RDoC-Integr8\Output\
```

An explanation of the above results:

- 1761 research subjects were in the source file and all were processed
- Family IDs 0074, 0148, and 0648 have two fathers (suffix of 02) and one mother (suffix of 03) respectively and are included in the research subject data but not directly linked to a family. Due to mixed family dynamics, sometimes parents may not have genetic relationships with the other members of the family and need to be handled individually.
- Subject (501)060742 has a suffix of 42 which is an idiosyncrasy that needs to be addressed manually. The subject's data is processed, but the RDoC team may choose to update information for that individual before uploading data to NIMH.