TRAC-M-TM-14-004
31 October 2013

Study to Assess Risk and Resiliency in Soldiers (STARRS) Validation

TRADOC Analysis Center
700 Dyer Road
Monterey, CA 93943

This study cost the
Department of Defense approximately
$73,000 expended by TRAC in
Fiscal Year 13-14.
Prepared on 20131119
TRAC Project Code # 060056

THIS PAGE INTENTIONALLY LEFT BLANK

TRAC-M-TM-14-004
31 October 2013

Study to Assess Risk and Resiliency in Soldiers (STARRS) Validation

Authors

MAJ(P) Tom Deveans
Dr. Sam Buttrey

PREPARED BY:	**APPROVED BY:**
THOMAS M. DEVEANS **MAJ(P), US Army** **TRAC-MTRY**	**CHRISTOPHER M. SMITH** **LTC, US Army** **Director, TRAC-MTRY**

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)* 31-10-2013	**2. REPORT TYPE** Technical Memorandum	**3. DATES COVERED** *(From - To)* 01-10-2012 -- 31-10-2013

4. TITLE AND SUBTITLE
Study to Assess Risk and Resiliency in Soldiers (STARRS) Validation

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
MAJ(P) Tom Deveans
Dr. Sam Buttrey

5d. PROJECT NUMBER
060056

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Training and Doctrine Command Analysis Center---Monterey
700 Dyer Road
Monterey, CA 93943-0692

8. PERFORMING ORGANIZATION REPORT NUMBER
TRAC-M-TM-14-004

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Army Analytics Group (AAG)
20 Ryan Ranch Road, Suite 290,
Monterey, CA 93940

10. SPONSOR/MONITOR'S ACRONYM(S)
AAG

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

This memo documents work done in Fiscal Year (FY) 2013 by the Training and Doctrine Command (TRADOC) Analysis Center – Monterey (TRAC-MTRY) as well as the Naval Postgraduate School (NPS) in support of the Army STARRS project. This project reproduced the original data set. As a result, a number of the insights we provide speak more to the usefulness and growth in capability of the Person-Event Data Environment (PDE) than to the analysis of the models themselves. In this section we provide the scope of the original project and our methodology as well as describing the PDE very briefly. Section 2 will provide our problem definition and our operating set of constraints, limitations, and assumptions. In section 3, we list some of the obstacles that made this project less successful than it might have been. It is our hope that future analysts will benefit from being aware of some of these issues early on. In the final section we provide some recommendations for both system and code improvement.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON** LTC Thomas Deveans
a. REPORT Unclassified	**b. ABSTRACT** Unclassified	**c. THIS PAGE** Unclassified	SAR	23	**19b. TELEPHONE NUMBER** *(include area code)* (831) 656-2452

THIS PAGE INTENTIONALLY LEFT BLANK

NOTICES

DISCLAIMER

Findings of this report are not to be construed as an official Department of the Army (DA) position unless so designated by other authorized documents.

REPRODUCTION

Reproduction of this document, in whole or part, is prohibited except by permission of the Director, TRAC, ATTN: ATRC, 255 Sedgwick Avenue, Fort Leavenworth, Kansas 66027-2345.

DISTRIBUTION STATEMENT

Approved for public release; distribution is unlimited.

DESTRUCTION NOTICE

When this report is no longer needed, DA organizations will destroy it according to procedures given in AR 380-5, DA Information Security Program. All others will return this report to Director, TRAC, ATTN: ATRC, 255 Sedgwick Avenue, Fort Leavenworth, Kansas 66027-2345.

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

AAG	Army Analytics Group
AR	Army Regulation
CORM	Concentration of Risk Model
EEA	Essential Elements of Analysis
FY	Fiscal Year
GB	Gigabytes
HQDA	Headquarters Department of the Army
NPS	Naval Postgraduate School
NIMH	National Institute of Mental Health
PDE	Person-Event Data Environment
PII	Personally Identifiable Information
SAS	Statistical Analysis Software
STARRS	Study to Assess Risk and Resilience in Soldiers
TB	Terabytes
TRAC	Training and Doctrine Command Analysis Center
TRAC-MTRY	Training and Doctrine Command Analysis Center – Monterey
TRADOC	Training and Doctrine Command

THIS PAGE INTENTIONALLY LEFT BLANK

SECTION 1. INTRODUCTION

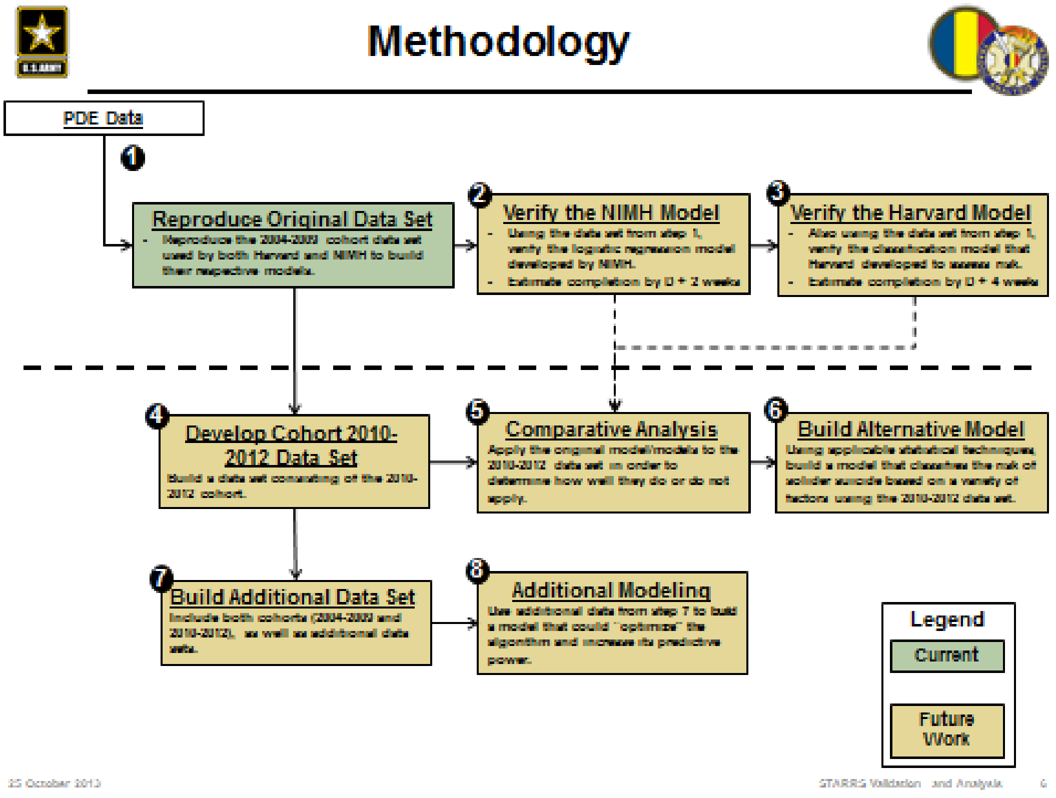
Promoting mental health and reducing suicide risk are important goals for Americans from all walks of life. Historically, the suicide rate among Army personnel has been lower than that of the civilian population. In 2004, however, the suicide rate among Soldiers began rising, reaching record levels in 2007 and again in 2008 and 2009. The situation prompted the Army to engage the National Institute of Mental Health (NIMH) in helping to address this issue. The Army Study to Assess Risk and Resilience in Soldiers (STARRS) is a direct response to the Army's request that NIMH enlist the most promising scientific approaches to better understand psychological resilience, mental health, and risk for self-harm among Soldiers.

Army STARRS is the largest study of mental health risk and resilience ever conducted among military personnel. Army STARRS investigators are using five separate study components – the Historical Data Study, New Soldier Study, All Army Study, Soldier Health Outcomes Study and Special Studies – to identify factors that help protect a Soldier's mental health and factors that put a Soldier's mental health at risk. The length and scope of Army STARRS means the study generates a vast amount of information. It also allows investigators to focus on periods in a military career that are known to be high risk for psychological problems. The information gathered from volunteer participants throughout the study will help researchers identify both potential risk factors and potential "protective" factors. Because promoting mental health and reducing suicide risk are important for all Americans, the findings from Army STARRS will benefit not only service members but the nation as a whole.

This report documents work done in Fiscal Year (FY) 2013 by the Training and Doctrine Command (TRADOC) Analysis Center – Monterey (TRAC-MTRY) as well as the Naval Postgraduate School (NPS) in support of the Army STARRS project. For a number of reasons, we were unable to accomplish any of the original objectives of this effort save the first one; that of reproducing the original data set. As a result, a number of the insights we provide speak more to the usefulness and growth in capability of the Person-Event Data Environment (PDE) than to the analysis of the models themselves. In this section we provide the scope of the original project and our methodology as well as describing the PDE very briefly. Section 2 will provide our problem definition and our operating set of constraints, limitations, and assumptions. In section 3, we list some of the obstacles that made this project less successful than it might have been. It is our hope that future analysts will benefit from being aware of some of these issues early on. In the final section we provide some recommendations for both system and code improvement.

1.1. SCOPE

The original objectives of this project essentially were to: one, reproduce the original data set and subsequently validate the analysis/model that had been done by the Department of Health Care Policy, Harvard Medical School, and two, test the resulting model (called the Concentration of Risk Model (CORM)) on a new data set, with the possibility of some model modifications. Figure 1 outlines the original methodology, with the green box showing the extent of our efforts this FY (due to a variety of reasons that will be discussed later on), and the tan boxes showing what is now future work.



Methodology

PDE Data

1 Reproduce Original Data Set
- Reproduce the 2004-2009 cohort data set used by both Harvard and NIMH to build their respective models.

2 Verify the NIMH Model
- Using the data set from step 1, verify the logistic regression model developed by NIMH.
- Estimate completion by D + 2 weeks

3 Verify the Harvard Model
- Also using the data set from step 1, verify the classification model that Harvard developed to assess risk.
- Estimate completion by D + 4 weeks

4 Develop Cohort 2010-2012 Data Set
Build a data set consisting of the 2010-2012 cohort.

5 Comparative Analysis
Apply the original model/models to the 2010-2012 data set in order to determine how well they do or do not apply.

6 Build Alternative Model
Using applicable statistical techniques, build a model that classifies the risk of soldier suicide based on a variety of factors using the 2010-2012 data set.

7 Build Additional Data Set
Include both cohorts (2004-2009 and 2010-2012), as well as additional data sets.

8 Additional Modeling
Use additional data from step 7 to build a model that could "optimize" the algorithm and increase its predictive power.

Legend
Current
Future Work

25 October 2013

STARRS Validation and Analysis 6

Figure 1: STARRS Analysis Methodology

The analysis done by Harvard in concert with the NIMH and using data from a repository at the University of Michigan, had attempted to identify soldiers at higher-than-average risk of suicide (We will use “Harvard” to mean “the set of analysts at Harvard”). Harvard’s work had consisted of building the data set and then producing and running a model thereupon; as mentioned above, our first task was to construct the very same data ourselves, using Harvard’s code but from the data in the PDE. The PDE is a virtual computer environment set up in association between the Defense Manpower Data Center and the Army Analytic Group. Users connect remotely to a server inside the PDE and, once there, can access data and analysis tools that are otherwise insulated from outside. In this way the security of Personally Identifiable Information (PII) is preserved. The PDE includes an Oracle database from which our data was drawn, and the SAS Enterprise Guide into which Harvard’s code was imported.

The code was delivered in four pieces (“rounds”) at the beginning of the project in October of 2012 and then an updated set was delivered in April 2013 (which was augmented in May). The code also included some documentation, log files (showing what happened when Harvard ran the code at their facility) and, in some cases, data (with, for example, ICD9 codes). A fifth round was delivered in July 2013, near the end of the project.

SECTION 2. PROBLEM DEFINITION

2.1. PROBLEM STATEMENT (ORIGINAL)

To validate and extend the statistical analysis and results from the Army Study to Assess Risk and Resilience in Soldiers (STARRS) based on syntax and variable coding provided by the original STARRS research team.

2.2. CONTRAINTS

2.2.1. The project will be complete by 30 September 2013.

2.2.2. Technical difficulties with the PDE caused the original project objectives to be re-scoped.

2.3. LIMITATIONS

2.3.1. Any analytic accuracy will be greatly impacted by the accuracy of the data.

2.3.2. Reproduction of the study team results may be hindered by the lack of documentation provided by the research team.

2.3.3. SAS server storage size and software versioning will significantly slow progress.

2.4. ASSUMPTIONS

2.4.1. The data collected and provided will be as accurate as possible.

2.4.2. We will be able to accurately reproduce the study team's work without proper documentation.

2.5. STUDY ISSUES AND ESSENTIAL ELEMENTS OF ANALYSIS (ORIGINAL)

Study Issue 1: Is the risk model produced by the previous analysis valid?

EEA 1.1: Can the models be reproduced from the original data?

EEA 1.1.1: The NIMH model.

EEA 1.1.2: The Harvard model.

EEA 1.2: How well does the model perform in classifying those Soldiers at risk of suicide?

EEA 1.2.1: The NIMH model.

EEA 1.2.2: The Harvard model.

Study Issue 2: How does the structure of the model change with the incorporation of additional data?

EEA 2.1: Does the model contain the same factors?

EEA 2.2: How does model performance change with the inclusion of the new data?

SECTION 3. OBSTACLES

In this section we lay out the obstacles that impeded progress on this work. The intent here is to record our experiences as an aid towards future developers in future projects. We are not intending to cast blame on collaborators, nor to exonerate ourselves for portions of the project that took unexpectedly long to perform.

We have divided obstacles into two types, although in fact these overlap. The first set of obstacles includes a number that have now been remedied; again, we record these for the benefit of future developers in other environments. These are system- and connectivity-related issues and refer mostly to the inadequacies and vicissitudes of hardware and networks. The second set of obstacles concerns certain design choices in the code. We believe that this code was developed in a piecemeal manner in order for its users to answer a particular question one time. It was not the result of a rigorous software development effort designed to produce a re-useable product, and we recognize that. Nonetheless some of these design choices made it difficult for us to work on the code and the data, and so we detail those issues in that section.

3.1. SYSTEM ISSUES

In this section we detail the system-related obstacles that future researchers should be aware of. These can be roughly broken into categories representing connectivity issues, on the one hand, and disk space and other system resources, on the other. Of particular import is the behavior of SAS on abnormal termination, so we discuss that briefly as well.

3.1.1. Connectivity Issues

The nature of the PDE is such that a lot of computers need to be in communication with one another more or less continuously during processing. These computers include:

- The client's computer (at which the analyst sits physically);

- The SAS Desktop (which serves as the virtual home for the analyst in the PDE, and from which the analyst can run the SAS Enterprise Guide);
- The SAS Server (on which SAS itself actually runs);
- The Oracle Server (where the Oracle database is maintained and served).

While we do not have much information about the internal construction of the PDE, it certainly makes sense to maintain a mental model that looks like this picture:

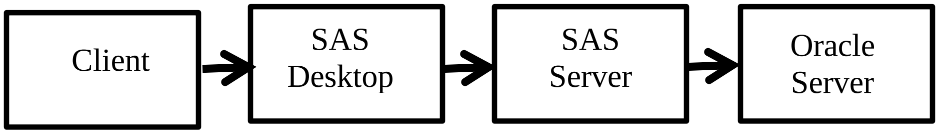


Figure 2: PDE Model

When our work on the project began, a failure of any component (either a computer or a link) was, we believe, enough to cause the SAS system to stop working. The connection between the client machine and the SAS Desktop was particularly troublesome. That connection, made via Citrix software, was very fragile, and, again, at the beginning of the project, any disconnection caused SAS to terminate abnormally (See "SAS WORK and What Happens On Termination," below). A restart of the client computer, as another example, would terminate the entire job, and these restarts were distressingly frequent as a result of, for example, software updates at the administrative level.

After some investigation, the PDE technical staff was able to find and alter certain timeout settings in such a way as to keep SAS Desktop sessions open for some length of time even when the client connection was temporarily lost. These actions, most of which took place in March 2013, constituted one of two main changes that made it possible for this work ever to be done (We learned that a second set of policies was needed for the SAS Server; these policies were modified in August 2013). The second main change was the addition of a large amount of disk space; see "Memory and Processing bottlenecks" below.

Of course, each of the computers and connections in our mental picture is subject to planned events like shutdowns for maintenance and unplanned events like system crashes. Even if the frequency of any one event is low, the chance of even one interruption during a three-day job is not always insignificant. We developed a number of strategies to protect against the delays caused by interruption, although in retrospect none of them were particularly successful. We note that the connection from the client to the SAS desktop was very much the weakest link in this chain. We did observe failures of all sorts, but, as we said in the previous paragraph, real progress was very difficult until the PDE staff was able to alter timeout settings so that SAS Desktop sessions could remain working even when clients disconnected.

Without our having spent much time on quantifying this, our intuition is that, before these timeout settings were altered, about 80% of stoppages were the result of the client disconnecting from the SAS Desktop; perhaps another 10% were the result of the client machine crashing; and most of the rest appeared to be associated with issues on the SAS Desktop itself. We did, however, see occasional evidence of the SAS Server crashing (or at least disconnecting from the SAS desktop) and of the Oracle server crashing or being inaccessible. Again, because of the large number of complex entities and connections, a certain crash rate is to be expected; the issue was, for us, the general inability to recover.

It is worth noting that there is a set of timeout settings for access to the SAS Server itself that appear to be separate from those for the SAS Desktop. One approach we used was to try to run SAS jobs in “native” SAS directly on the SAS Server, rather than use Enterprise Guide from the SAS Desktop. This approach required that the Desktop’s own security settings be properly adjusted.

3.1.2. Memory and Processing Bottlenecks

A second set of issues arose when the resources of the PDE were inadequate to handle the demands of the code (In part this may have been due to inefficiencies in the code, but there is no way to get around the fact that this is a large problem). At the beginning of the project (October 2012) the available space on the SAS server was in the range of about 100MB (megabytes). The issue of insufficient space was detected in the opening months. After an upgrade at the end of March 2013, the available disk space on the SAS Server was around 1.85 TB (terabytes). We learned in August 2013 that Harvard's server required 2 TB of data in order for their scripts to run. The PDE technical staff was able to secure and install the required disk space at the beginning of September 2013 – only, of course, after this requirement was made known to them. Disk space tended to be used up by intermediate products that were not removed because SAS terminated abnormally. This created another burden on the PDE technical staff; that of policing the disk drive in search of large files that could be safely removed. When the disk on the SAS Server filled, SAS terminated abnormally.

In addition to the lack of disk space we suspect that the SAS Desktop's stability could be endangered by the presence of multiple users. We observed a number of crashes that did not appear to be attributable to other causes, or to bad luck. As the PDE grows the number of machines dedicated to handling users should probably grow as well.

3.1.3. SAS Work and Termination

Intermediate products computed by SAS are stored in a directory named WORK which resides on the SAS Server. These products are regular disk files with extension `sas7bdat` and they persist until either they are explicitly deleted inside the SAS code, or SAS terminates normally. When SAS terminates, all the members of the WORK directory are deleted from the disk. If SAS terminates abnormally, items in WORK that have been completed will normally be complete and useable in a future session, although we were wary of using SAS items recovered after a crash, and would need to be explicitly removed in order to free up disk space (Certainly an item that is in use when SAS terminates abnormally will be unusable, because it will be either corrupt or incomplete).

3.1.4. Failure Strategy

Before it became clear that the PDE lacked sufficient disk space to run the Harvard code in the form in which it was presented, we attempted to run code through a makeshift strategy of breaking scripts into pieces and running them bit by bit. This approach turned out to raise problems of its own; see section 3.2.5, “Breaking Scripts into Pieces,” and section 3.2.5.3, “Removal of Intermediate Products,” below.

3.2. CODE ISSUES

In this section we describe some of the issues that required attention or correction before Harvard code could be used in the PDE.

It must be said that Harvard’s code was clearly not the product of a software development process designed to produce streamlined, well-tested code. Instead it had the hallmarks of a set of ad hoc programs put together in order to solve a particular problem one time. This is understandable given the nature of the project, but it made the code’s re-use difficult.

The code was only sporadically documented (Documentation on the outside, describing the flow from one program to the next, was somewhat better). Certain coding standards that we have come to expect (the use of meaningful variable names that are not re-used from one task to a different one) were not always met. Again, we understand that code is often developed under time pressure and the rewards for comprehensive documentation are small. Nonetheless, the code is clearly not of commercial quality or presentable to clients.

3.2.1. Mismatched File and Library Names

Tables in Harvard’s data center often had slightly different names than the corresponding tables in our Oracle database. References to tables with different names had to be found and changed. In principle this might have been done automatically after constructing a table with one row per table and two columns giving the different names, but in practice we did this by hand.

It is also the case that Harvard’s own internal table- and library-naming conventions were inconsistent. For example the final script from round 1, script 2 is named `final1` and stored in a library named `linuxdrv`. In script 5 this is read in from a library named `new1`, and, separately from `new`. We learned that these libraries were maintained by different analysts. It might be worth noting that, also in round 1 script 5, another dataset named `final1` is created. This is not the same as the earlier dataset by that name. We detected some of these differences only through conversation with Harvard. Others – particularly the issue of libraries having different names – were widespread enough to be mentioned in several places in Harvard's documentation.

3.2.2. Changing the ID Name

The simplest change we had to make to the code was to convert all references to Harvard's ID (which they called PID_CHPPM) to the ID in the PDE (called PID_PDE). For most scripts, a simple global change was all that was necessary. However, we did encounter several issues with IDs. First, in a number of places the Harvard code specified particular IDs, presumably to resolve issues with individual records. Because we lacked the ability to convert CHPPMs to PIDs, we ignored all of these references.

Second, in a number of cases the code creates additional ID columns (in, for example, `dcips_injury_afmets_death_agregation`). These new columns were given width 9 in Harvard's code, whereas we require that they have width 12. These references had to be found and changed.

Third, in a few instances the SAS code would refer to the column names directly, which caused problems with case-sensitivity. SAS variable names are not case-sensitive, so we got into the habit of making a global, case-sensitive conversion of all instances of PID_CHPPM (or pid_chppm, or whatever; we saw plenty of each) to `pid_pde` in every script. But, for example, on line 1798 of `allsx_cpt_v7_linux_3` we see this macro code:

```
%let var = %sysfunc(varname(&dsid, &i));

%if &var ne yearmonth & &var ne PID_PDE &then …
```

Here the value contained in var is being compared to the strings `yearmonth` and `PID_PDE`. This comparison is a case-sensitive one, so if in this case an instance of `PID_CHPPM` (upper-case) was converted to `pid_pde` (lower-case), this code will fail. In short, even the act of converting one ID name to another could cause problems.

3.2.3. Code That Didn’t Work

3.2.3.1 Direct Sorting of Oracle Data

In some cases code that we were provided did not run in our environment. One type of difference arose from the different ways that data was being accessed by the SAS engine – Harvard used an ODBC connection, while we used the SAS/ACCESS interface to Oracle. If `tmds` were the name of an ODBC connection, then Harvard’s code would use that library and execute a command like this (this example comes from `make_tmds_constructs_1.sas`):

`Proc sort data=tmds.tmds_pem_inpt_dt` (other clauses here)

We presume that this call was successful for Harvard, but it does not appear to be valid in the case where `tmds` is an Oracle connection. Instead, we needed to read the data into SAS first, and sort in a separate step, like this:

```
Data holder; set tmds.tmds_pem_inpt_dt; run;
```

`Proc sort data=holder` (other clauses here)

3.2.3.2. Transposition Without Prefix

In some cases (e.g. `make_tmds_constructs_1.sas`) the Harvard code used `Proc Transpose` to transpose a matrix. The `PREFIX`= option specifies a prefix to be added to the front of the names of the newly constructed columns. This prevents SAS from constructing columns names that are invalid (because, for example, they start with a numeric character). In this case, the new columns were constructed with names of ICD9 codes, which in many cases start with a numeric character. No `PREFIX` option was used, so we supplied one. It is not clear how this code could have worked at Harvard.

3.2.4. Diagnoses With No Soldiers

In script 4 of round 2 (`alldx_merge_master_4`), we encountered a number of errors that we attributed to referring to diagnoses for which there were no soldiers recorded. This might have been caused by differences between our data’s and Harvard’s, but the ultimate genesis of this issue is still unresolved. In the meantime, we have removed references to these diagnoses.

3.2.5. Breaking Scripts Into Pieces

During the time that we found ourselves unable to run jobs to their completion, because of connection or resource issues, we tried to break the scripts into pieces and run the pieces one at a time. This approach, too, ran into difficulties. These difficulties included the use of global variables, macros, and deletion of intermediate products.

3.2.5.1. Global Variables

Some scripts used global variables. These are variables that do not belong to a SAS data set. They can be set or retrieved anywhere in the code. As an example of their use, imagine trying to construct a dataset with one row for each individual and one column for each separate diagnosis for that individual. It would be useful to know the maximum number of diagnoses that any individual had, in order to dimension the resulting data set. One way to accomplish this would be to read all diagnoses, accumulating counts by ID, and saving the maximum number of diagnoses to a global variable.

The issue is that when running code in pieces, any global variable that will be needed in a particular piece has to be identified and set before the piece can be started. A more general solution might have been to write these variables to a small text file or SAS data set for more permanent storage between runs.

3.2.5.2. Use of Macros

A SAS macro is a sub-function that allows the re-use of code in different contexts. Macros can be very useful and we will not argue for their elimination. However, SAS Macros are written in a modified form of the language that makes reading and understanding macro code difficult. We believe that long complicated macros deserve particularly strong documentation.

If a section of code needs to be run several times, a macro is a natural choice. However, it is difficult to run only a piece of a macro, which is best seen as an indivisible unit. So a script consisting of a macro definition followed by a single invocation confers no advantage in efficiency over “regular” code and suffers from being very difficult to divide into pieces. We encourage developers to avoid this.

3.2.5.3. Removal of Intermediate Products

In a number of instances, the SAS code explicitly removes intermediate products. This is a good practice when disk space is at a premium, but a bad one when crashes are common, since if all intermediate products up through a particular point can be preserved, the program can be re-started at that point with no loss. We do not fault Harvard for their choices in this matter, but, seemingly inevitably, we found ourselves both running out of disk space and being unable to restart at intermediate points.

3.2.5.4. Copying Results Across File Systems

Because of insufficient disk space, we would often attempt to run a piece of the SAS code, copy intermediate products over to another file system on the network, delete the product from the SAS server, and then resume. While this makes sense in principle, it is painful in execution. These disk copy operations often took several orders of magnitude longer than would an equivalent copy within a file system – and, as we have said, the danger of a crash during any extended operation is non-zero. Still, when disk space was at a premium something had to be done.

SECTION 4. RECOMMENDATIONS

We look at this project as part of the PDE's startup cost. Had the PDE been configured at the beginning of the project as it is now, we expect that this project could have been completed in a few months. Primary PDE improvements listed below.

4.1. SUFFICIENT DISK SPACE

The initial configuration of the SAS server was insufficient for the project. The current setup, at about 2TB on the main disk, is just barely adequate. Given that disk space is comparatively cheap, we urge administrators to obtain and make available as much disk space as possible.

4.2. PERMISSION CONTROL

When PDE technical staff was able to arrange for SAS Desktop sessions to continue after a disconnection, progress was hugely improved. It is important to note that the settings for the SAS Server are distinct. We thank the technical staff for their flexibility and recommend that these settings be maintained the way they are now.

4.3. INCREASED PROCESSING POWER

We suspect, but cannot prove, that some crashes were caused by contention among users for resources, either those of SAS, or Oracle, or of the network. As the number of PDE users grows, so too will this contention. We recommend that PDE administrators continue to have the environment grow in terms of the processing resources available. Perhaps parallel computing architectures can be brought to bear for the big jobs we expect to see in the future.

4.4. CODE

As mentioned earlier, the code given to us by Harvard was inadequate for the task. We recommend that quality control be performed on the code and the documentation as part of any contract. The code not being able to run because of Harvard's own internal naming system inconsistencies is less than acceptable.