## Contents

The Personnel Inventory Aging and Promotion (PIAP) model ........1

SourceData.mdb....................................................................................3

Yr0.mdb .................................................................................................3

Running the PIAP promotion model ..................................................6
  Paygrade .............................................................................................7
  Outputs..............................................................................................7
  What the model does .................................................................8
  PIAPM.mdb — the PIAP model ......................................................11
  Controller form..............................................................................12
  Increase Decrease Personnel Form .......................................13
  Increase Decrease Manpower Targets form .......................14
  Warning form.................................................................................15

PIAP data processor ............................................................................17
  PIAPM.xls.......................................................................................17
  Using the driver...............................................................................18
  Importing new data.........................................................................20
  Compiling multiple runs.............................................................27
  Sensitivity data...............................................................................27

The model is user configurable .........................................................31

Appendix A: Yr0.mdb programming code .................................33
  Table of contents .........................................................................33
  MakeData.......................................................................................34

Appendix B: PIAPM.mdb programming code .............................39
  Table of contents .........................................................................39
  Main.................................................................................................41
  Preliminaries................................................................................52
  Stats.................................................................................................58
  Utilities ............................................................................................62
  Controller form..............................................................................66
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Personnel form</td>
<td>68</td>
</tr>
<tr>
<td>Manpower form</td>
<td>69</td>
</tr>
<tr>
<td>Appendix C: PIAPM.xls programming code</td>
<td>71</td>
</tr>
<tr>
<td>Table of contents</td>
<td>71</td>
</tr>
<tr>
<td>Main</td>
<td>73</td>
</tr>
<tr>
<td>Compile</td>
<td>78</td>
</tr>
<tr>
<td>Formatting</td>
<td>88</td>
</tr>
<tr>
<td>Robust</td>
<td>92</td>
</tr>
<tr>
<td>Utilities</td>
<td>100</td>
</tr>
<tr>
<td>Choice form</td>
<td>108</td>
</tr>
<tr>
<td>Workbook</td>
<td>110</td>
</tr>
<tr>
<td>Sheet4</td>
<td>111</td>
</tr>
<tr>
<td>Sheet6</td>
<td>112</td>
</tr>
<tr>
<td>References</td>
<td>113</td>
</tr>
<tr>
<td>List of figures</td>
<td>115</td>
</tr>
<tr>
<td>List of tables</td>
<td>117</td>
</tr>
</tbody>
</table>
The Personnel Inventory Aging and Promotion (PIAP) model

This manual describes not only how to use and maintain the PIAP model, but it also discusses its development, structure, usage, and outputs. Additionally, the manual provides guidance for interpreting the results.

The PIAP model can be used to examine the effect of various man-power policy implementations and their future consequences to the Navy’s personnel profile. The user may analyze how policy changes will affect promotion tempo, promotion rates, likelihood of promotion, separation rates, and gaps between requirements and personnel.

The PIAP model incorporates several files in two different formats: Access and Excel\(^1\). The Access database, SourceData.mdb, contains the base data compiled from the original data from the Defense Manpower Data Center (DMDC). A second database, Yr0.mdb, links to SourceData.mdb and prepares the data for processing by the PIAP model, which is contained in a third Access file named PIAPM.mdb. The model generates numerous Excel outputs that are compiled by PIAPM.xls to produce statistics and charts describing the PIAP model’s results. Figure 1, below, is a simple schematic that depicts the full process from input data to final results.

---

\(^1\) We developed the model using Access 2000; its data processor was developed with Excel 2000. We tested both with the Office 2003 versions of these applications and found that they have full functionality.
Figure 1. Schematic of the PIAP model and data processor

HYT and TIS/TIG requirements

Attrition rates

Loss data from DMDC

DMDC data

Clean data

Full database

Yr0 database

Process

Model

Outputs

Data processor

Sensitivity data

Compiled data

Data preparation phase

Yr0 preparation phase

Promotion model phase

Excel phase

\(^2\) High Year Tenure and Time in Service/Time in Grade
SourceData.mdb

In the interest of limiting the model’s file size, the base dataset, SourceData.mdb, contains the data from DMDC and should be held inviolate. It must contain the fields and data types as listed in table 1:

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ssn</td>
<td>Text</td>
<td>Psuedo SSN for tracking individuals year by year</td>
</tr>
<tr>
<td>rate</td>
<td>Text</td>
<td>The two- or three-character rating</td>
</tr>
<tr>
<td>grade</td>
<td>Text</td>
<td>Paygrade</td>
</tr>
<tr>
<td>mos</td>
<td>Integer</td>
<td>Months of service</td>
</tr>
<tr>
<td>yos</td>
<td>Integer</td>
<td>Years of service</td>
</tr>
<tr>
<td>mig</td>
<td>Integer</td>
<td>Months in grade</td>
</tr>
<tr>
<td>yig</td>
<td>Integer</td>
<td>Years in grade</td>
</tr>
</tbody>
</table>

Note that the model only uses paygrades E3 to E9 and that all E1s and E2s have been “promoted” to E3. We believe this to be valid since promotion to E3 is virtually automatic after 1 year of service [1], and the model utilizes actual Time in Service and Time in Grade for promotions and separations. Since the model only handles the enlisted community, the grade field is of the form E03, E04, etc., but the actual format is irrelevant (it may be either numeric or text) as long the numeric part of the paygrade (i.e., 3, 4, etc.) is in the farthest right-hand position.

Yr0.mdb

The purpose of the Yr0.mdb database is to clean the source data and prepare it for the PIAP model. It contains a link to the SourceData table in SourceData.mdb, and it also has one physical table named RealAttrRates with the following structure:

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>rate</td>
<td>Text</td>
<td>The two- or three-character rating</td>
</tr>
<tr>
<td>pg</td>
<td>Byte</td>
<td>Paygrade</td>
</tr>
<tr>
<td>yos</td>
<td>Integer</td>
<td>Years of service</td>
</tr>
<tr>
<td>prob_sep</td>
<td>Double</td>
<td>Probability of separation in the next year</td>
</tr>
</tbody>
</table>
The attrition rates in this table are based on a 5-year weighted average\textsuperscript{3} in the DMDC data through fiscal year 2007 with adherence to the Navy’s High Year Tenure rules\textsuperscript{4} \cite{DMDC2007}. 

Figure 2. Yr0 database window

In addition, this database contains three macros accessible from the Database Window: AutoExec, MakeRealYr0, and MakeYr0. AutoExec automatically executes when the database opens and refreshes the link to SourceData.mdb as long as the file is somewhere in the model’s file path. MakeRealYr0 cleans the source data by deleting records with invalid paygrades, by converting the paygrades to a numeric field if necessary, and by creating a new table named RealYr0. This macro need only be run when SourceData.mdb has been updated with new data. The programming code for these macros and all supporting functions and subroutines can be found in appendix A.

\textsuperscript{3} The weighting formula for attrition rates is:

\[ \text{Rate}_{2003} \times 2 + \text{Rate}_{2004} \times 3 + \text{Rate}_{2005} \times 4 + \text{Rate}_{2006} \times 5 \]

\textsuperscript{4} The 1 July 2005 change and Grandfather Clause to High Year Tenure for Navy E5s is handled programmatically.
Make Yr0

The MakeYr0 macro is used to create the rating specific data that will be accessed by the PIAP model. On activation, this macro

1. Asks the user to input a two- or three-character rating; if a three-character rating is entered, it is assumed that it is a compressed rating [3] that becomes a rating denoted by the first two characters at higher paygrades.

2. Determines the paygrade where the rating becomes compressed and calculates the ratio for distributing the compressed paygrades.

3. Queries the RealYr0 table for all records with the rate field equal to either the two- or three-character rating and stores them in a new table named Yr0.

4. “Promotes” all E1s and E2s to E3.

5. Deletes all records where any of the Time in Service or Time in Grade fields are missing or if Time in Grade is greater than Time in Service.

6. Warns the user if there is an overlap or a gap between the compressed and uncompressed paygrades. The user is asked to resolve the problem and code execution ceases.

7. Identifies each three-character rating that feeds into the compressed paygrades and calculates the proportion of the chosen rating among the uncompressed paygrades. It then randomly selects records among the compressed paygrades in this proportion. Those not selected are dropped from Yr0, and the rating is changed to the chosen three-character rating for those selected. Since the selection for distribution is random, one execution of this macro will not result in the same dataset as that of another execution.

---

For example, suppose the ratings ZZA, ZZB, and ZZC (composed of 10,000 sailors) combine into the rating ZZ at E9 (with 100 sailors) and that there are 5000 ZZAs, 3000 ZZBs, and 2000 ZZCs at paygrades E3-E8. If the user chose the rating ZZB, the model would randomly assign approximately 30 E9s to the ZZB rating.
8. Creates two new tables in the database.
   
a. PGRollup has nine records containing the number of records in Yr0 for each paygrade.
   
b. RateRollup has one record with the total number of records in Yr0.
   
9. Selects records from the RealAttrRates table that have the three-character rating for uncompressed paygrades and the two-character rating for compressed paygrades, and it puts them into a new table named AttrRates.

   The final Yr0 table has the following structure:

   Table 3. Yr0 fields

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ssn</td>
<td>Text</td>
<td>Pseudo SSN for tracking individuals year by year</td>
</tr>
<tr>
<td>pg</td>
<td>Integer</td>
<td>Paygrade</td>
</tr>
<tr>
<td>rate</td>
<td>Text</td>
<td>The two- or three-character rating</td>
</tr>
<tr>
<td>yos</td>
<td>Integer</td>
<td>Years of service</td>
</tr>
<tr>
<td>vig</td>
<td>Integer</td>
<td>Years in grade</td>
</tr>
<tr>
<td>months</td>
<td>Integer</td>
<td>Months of service</td>
</tr>
<tr>
<td>mos_pg</td>
<td>Integer</td>
<td>Months in grade</td>
</tr>
<tr>
<td>Drop</td>
<td>Long</td>
<td>Unused</td>
</tr>
</tbody>
</table>

Running the PIAP promotion model

   The model produces three sets of outputs for each run, and each set uses a different promotion rule.

   1. The Junior Rule first promotes those with the least Time in Service, assuming they meet the minimum requirements, and then it promotes progressively older individuals. See table 4 for minimum Time in Service / Time in Grade requirements [1].

   2. The Benchmark Rule first promotes those individuals whose Time in Service is closest to the established Navy benchmarks. See table 5 for the benchmarks currently in
use. These can be changed by altering the ENs\(^6\) function in the Main module. The programming code for the model can be found in appendix B.

3. The Senior Rule first promotes those with the most Time in Service and then promotes progressively younger individuals.

These rules establish a range for the promotion tempos that could be achieved with a given force profile.

Table 4. Service requirements for promotion (in months)

<table>
<thead>
<tr>
<th>Paygrade</th>
<th>Minimum Time in Service</th>
<th>Minimum Time in Grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>24</td>
<td>6</td>
</tr>
<tr>
<td>5</td>
<td>36</td>
<td>12</td>
</tr>
<tr>
<td>6</td>
<td>84</td>
<td>36</td>
</tr>
<tr>
<td>7</td>
<td>132</td>
<td>36</td>
</tr>
<tr>
<td>8</td>
<td>132</td>
<td>36</td>
</tr>
<tr>
<td>9</td>
<td>228</td>
<td>36</td>
</tr>
</tbody>
</table>

Table 5. Navy time to promotion benchmarks [4]

<table>
<thead>
<tr>
<th>Paygrade</th>
<th>Months</th>
</tr>
</thead>
<tbody>
<tr>
<td>E4</td>
<td>26</td>
</tr>
<tr>
<td>E5</td>
<td>53</td>
</tr>
<tr>
<td>E6</td>
<td>108</td>
</tr>
<tr>
<td>E7</td>
<td>148</td>
</tr>
<tr>
<td>E8</td>
<td>222</td>
</tr>
<tr>
<td>E9</td>
<td>266</td>
</tr>
</tbody>
</table>

**Outputs**

The model produces outputs that allow for the analysis of

- Personnel profile by Time in Service, paygrade, promotion rule, and year.

- Time in Service and Time in Grade by paygrade, promotion rule, and year.

- Promotions by paygrade, promotion rule, and year.

---

\(^{6}\) ENs is the name of a function in the model’s programming and does not refer to the Navy’s Engineman rating.
• Gaps between personnel and requirements by paygrade, promotion rule, and year.
• Separations by paygrade, promotion rule, and year.
• Time to promotion by paygrade, promotion rule, and year.
• Likelihood of Promotion to the next paygrade for the current personnel inventory by Time in Service, paygrade, and promotion rule.

The model also produces a table for each year named Yr1, Yr2, etc. The tables contain individual results so that an individual’s career may be followed on a year-by-year basis. These tables have the following naming convention: Each is prefixed with the first letter of the rule and the rating currently being analyzed. For example, running the GSE rating for 5 years would produce

- JGSEYr1, JGSEYr2,…, JGSEYr5
- MGSEYr1, MGSEYr2,…, MGSEYr5
- SGSEYr1, SGSEYr2,…, SGSEYr5

What the model does

The following nested pseudo code provides a simplified summary of the model’s process, beginning with the current personnel inventory:

For each run

  For each promotion rule

    Make preparations

  For each year

    Separate E9s


\(^7\) For formatting purposes in the final output, the Benchmark Rule is identified by the letter “M” rather than “B” in order to allow Excel to use its default alphabetical ordering.
For each paygrade E8-E3

    Separate
    Promote
    Next paygrade
    Access new E3s
    Compile data for year

    Next year
    Compile data for all years
    Cleanup
    Next promotion rule
    Next run

When the user clicks the Run button, any remaining tables from previous implementations are deleted and the target numbers for each paygrade are calculated simply by taking the number in each paygrade in [Yr0]\(^8\) and adjusting for changes in requirements as entered in the Increase Decrease Manpower Targets form. New tables are created to hold the output data. Each year up to the Number of Years input are handled in turn. First, the E9s in [Yr0] are loaded into a temporary table named “temp” and merged with data for separation probabilities and the manpower requirement [target] for E9. Each record in [temp] is either separated or aged depending on the value of a random number (0 <= \(\alpha\) <1). If this number is less than the separation probability for that rating ([rate]), paygrade ([pg]), and years of service ([yos]) combination, then [target] is set to NULL; otherwise, [yig] is increased by 1 and [months] is

\(^8\) It is common custom to denote database tables and fields by enclosing their names in brackets. When it is necessary to refer to a field in a specific table, the convention is to use the table name in brackets, an exclamation mark, and the field name in brackets, e.g., [table]![field].
increased by 12. Now, an SQL\(^9\) statement deletes records where [target] = NULL, and the model calculates the number of E8s that need to be promoted to reach the target for E9s. The remaining records are loaded into a new table [Yr1].

Paygrades E8 to E3 are then handled in descending order. Each paygrade in turn is loaded into [temp] from [Yr0], along with the separation probabilities and the manpower requirements (in the process destroying the old [temp] table). These are sorted and indexed by [months] depending on the promotion rule. If it is the Junior Rule, then the records are in ascending order; they are in descending order if it is the Senior Rule. For the Benchmark Rule, the records are in ascending order of the absolute value of the difference between Time in Service ([months]) and the benchmark for that paygrade. The records are chosen for separation in the manner described above and deleted from [temp], and the number of promotions needed for the next lower paygrade is calculated. [months] and Time in Grade ([mos_pg]) are increased by 12 for the remaining records, and the program moves through the sorted records promoting each individual that is eligible until either there is no further need for more promotions or until the end of the data is reached.

Those promoted have Years in Grade ([yig]) and [mos_pg] set to 0; otherwise, [yig] increases by 1 and [mos_pg] increases by 12. The records are loaded into [Yr1], and the next paygrade is processed. At this point, data for Time in Service, Time in Grade, separations, and promotions are collected in tables that will be output later. Finally, the number of required accessions to E3 is calculated on the basis of current end strength, predicted first year attrition, and the manpower requirements entered by the user. The new E3s are added to [Yr1], and each is assigned a unique identifier in the [ssn] field. These are easily identified in the yearly table because the first character is the letter “A.”

When this process is completed for each year, the statistics are compiled and loaded into new tables. First, [AllYrs] is created with the

---

\(^{9}\) Structured Query Language—the industry standard language used by Access to manipulate database tables.
help of temporary queries. For each year and paygrade, the target number and actual resulting count is calculated by the first query. The second query calculates average number of months to promotion, by paygrade, for those promoted in that year. These two queries are combined and loaded into [AllYrs].

At the completion of the final year of the run, the [Likelihood] and [Expected] tables are created. The first step in this process is to dynamically build an SQL statement, based on the Number of Years, which creates a temporary table adding promotion results to the data for each paygrade/years of service cohort in the [Yr0] table. This table has individual-level, longitudinal records. Next, a dynamically built SQL statement based on the Number of Years, creates a new data structure and calculates counts and yearly averages by cohort in [Likelihood]. Another one calculates the overall likelihood of promotion. [Expected] is created in the same manner but uses only those records where there has been a promotion at some point. **Only the FIRST promotion for an individual is considered for the [Likelihood] table.** At the end of each run, the data tables are renamed and exported to Excel for further processing by PIAPM.xls.

**PIAPM.mdb — the PIAP model**

PIAPM.mdb contains four linked tables, five forms, one macro (the same AutoExec as in Yr0.mdb, described above), and six code modules. The linked tables (Yr0, AttrRates, OccRollup, and PGRollup) are linked to the Yr0.mdb database so as not to make the model unnecessarily large. Due to Access’ inherent inefficiencies, the model’s file size grows rapidly, so the user must compact it frequently. To do this, click on the menu bar: Tools > Database Utilities > Compact and Repair Database. The database will perform this automatically when it is closed.
When the model opens, the Controller form automatically opens. The form has two text boxes for user input, three visible buttons, and two hidden buttons in the bottom left- and right-hand corners. The user enters the Number of Years (required) into the future for which the model will project and generate data. The user also enters the number of runs that the model will make in order to smooth out the variations that occur. We explain why this is necessary later in this document. A subdirectory for each run will be created in the model’s directory to hold its outputs. The Kill Tables button deletes all tables and queries created by the model, including those that were not destroyed during program execution because of an error or user intervention leading to program termination. All of these tables are deleted at the beginning of each run, but this button allows the user do so at will, usually before closing the file to reduce its storage size. The hidden button in the bottom right-hand corner deletes the output files in the subdirectories. These files are also deleted at the beginning of each run. This button requires a double-click because these deletions are permanent and the files cannot be recovered from the Recycle Bin. The hidden button at the bottom left-hand corner requires only a single-click and exports all of the form and code modules to a subdirectory named Modules that must already exist in the same directory where the model resides. For both of these hidden buttons, a message box alerts the user that the operation was successful. Clicking the Change Req’s-Pers button opens the Increase Decrease Personnel form, and clicking the “Run” button begins the model’s execution.
This form allows the user to add or subtract accessions. The user can select a one-time change, a permanent-step change (the same number or percent change every year), or a constant rate of change. By default, the model accesses to E3 the number that it predicts it will need to meet the personnel end strength requirements.

By entering a number in the first text box, the model will access that number over and above requirements. This is equivalent to changing the E3 requirement on the Increase Decrease Personnel form by the same amount. Entering a decimal (e.g., .1 to increase by 10 percent) in the second text box will do the same on a percentage basis. Enter a negative number to effect an equivalent decrease in accessions. If there are values in both text boxes, the first box will be used. Choosing 1-Time Change causes this increase to be applied to the first year only; choosing Permanent Step Change applies the increased accessions to each year; and choosing Constant Rate of Change will increase the accessions by this number or percentage in each year, compounding the change in the case of a percentage change.

Clicking the Use These button opens the Increase Decrease Personnel Manpower Targets form.
Increase Decrease Manpower Targets form

This form allows the user to alter the manpower requirements, promotion rules, and attrition rates. The first row of this matrix allows the user to enter an annual change in requirements for each paygrade, and the fourth row denotes the year in which the increase/decrease will end. For example, if there are currently 100 E5s and the user enters 10 in the first row of the E5 column and 5 in the fourth row, the requirements will be 110 in the first year, 120 in the second, and so on. In the fifth year and beyond, the requirements will be 150. Correctly choosing a Type of Increase/Decrease is necessary since this example would increase requirements 100 percent each year if By Percentage were erroneously checked; this would likely cause the database to exceed its maximum size of 2Gb and make it permanently unusable. In this example, an alternative method of reaching 150 in the fifth year is to calculate the percentage change necessary in each year to reach that goal:

\[
\left( \frac{150}{100} \right)^{t} - 1 = .08447
\]
Enter that number into the first row and choose By Percentage for Type of Increase/Decrease. In the fifth row, the user can vary the overall assumed attrition rates for each pay grade. For example, if E5 has an overall loss rate of 10 percent, entering .2 in the fifth row under E5 will result in an overall attrition rate of 8 percent. For ease of entry, rows one, four, and five have buttons to the far left that will copy the values in the E3 column to all columns. The second and third rows of the matrix allow variations in Time in Service and Time in Grade minimums (in months) for promotion to each pay grade.

**Warning form**

Since the consequences of reaching Access’ maximum file size are so dire (permanent file corruption and nonfunctionality), we have included programming that monitors the size of the database and warns the user when it reaches 50 percent, 75 percent, and 90 percent of the size limit. In addition, the Warning form projects the size of the database at the end of its runs, and, should its projected size exceed the limit, it warns the user at each of these points with increasingly urgent (and apocalyptic) messages on the Warning form. Inexplicably, Access itself provides no such warnings; it simply continues to add data until it stops functioning.

---

10 Recall that attrition rates are based on rating, pay grade, and years of service, so the effect may be, for example, to reduce the rate for a sailor in his tenth year from 20 percent to 16 percent and to reduce the rate for a sailor in his eleventh year from 5 percent to 4.5 percent. However, since these are linear transformations, the seniority profile of the E5s will not affect the overall change in attrition.
Figure 6. Database exceeds 1Gb

Figure 7. Database exceeds 1.5 Gb and is projected to fail
PIAPM.xls

The Excel workbook, PIAPM.xls is the driver for 1) importing the spreadsheets that were output by the PIAP model; 2) processing the data; 3) resetting the pivot tables, charts, and control objects; and 4) creating the sensitivity data. This file must be in the same directory as PIAPM.mdb and the Runs subdirectories containing the new data. When the file opens, it creates a new toolbar at the bottom left of the window with three buttons captioned Import New Data, Compile Multiple Files, and Sensitivity Data. When the file closes, this toolbar is destroyed.

The driver compiles the data exported by PIAPM.mdb to produce datasheets, pivot tables, and charts depicting

- Mean Time in Service and Time in Grade
- Percentage of individuals in each paygrade who promote
- Gaps between manpower and requirements
- Attrition rates
- Time to Promotion ranges achievable under the current promotion requirements and personnel profile
- Likelihood of promotion in each year for the individuals in the current inventory

Each of these metrics can be examined as year-by-year trends, by paygrade, and under any of our assumed promotion rules. The programming for PIAPM.xls is in appendix C.
Using the driver

PIAPM.xls contains templates for receiving the newly imported data, and these templates are updated to handle the configuration of the variably structured data (in terms of years and the number of runs that the user entered in the PIAP model).

Figure 8. Import Compile and Sensitivity buttons

When the user clicks the Import New Data button, he/she is prompted to enter the number of runs that the model had executed to produce the outputs, and the driver will import from each of the subdirectories up to this number. After importing and processing the data, the driver will save the file with the name Results_date_times.xls where date is the current date and time is the current time. This results in one Excel file for each run. **Note: The driver opens and creates literally hundreds of workbooks and must keep track of each, so it is strongly recommended that the user allow the program to complete without interference, i.e., the user**

11 Date is in the format “MMDDYY,” and time is in the format “HHMMSS.”
should not attempt to use or activate any other application until it is finished; otherwise, the driver will likely fail.

Clicking the Compile Multiple Files button imports the data from files created by the Import New Data procedure. It will attempt to import all Excel files whose name begins with “Results_*,” so it is necessary to remove all of the files created by a previous run. If these old files used a different number of years, the program will inform the user and abort, but if the same number of years were used but with different inputs, the consequence will be a mixture of results. The user could inadvertently attempt to analyze data produced from multiple, possibly contradictory, assumptions. The compiling procedure takes the results of these files, averages them, and calculates the standard deviations, minimums, and maximums. We do this to smooth out the variation among the PIAP model’s runs. These smoothed data are used to produce a new file containing all of the datasheets, pivot tables, and charts contained in the individual files, plus the extra metadata statistics created during the compiling process. The driver saves the compiled data to a file named Compiled_runs_Files_years_Yrs.xls where runs is the number of runs and years is the number of years used in the PIAP model.

While promotions in this model are deterministic, the separation aspect is random at the individual level making the model a stochastic process. The probability of separation is assigned to each individual by rating, paygrade, and years of service based on a weighted 5-year average. Thus, it is necessary to get an idea of the robustness of the model.

With the file created by the Compile Multiple Files procedure open and activated, clicking the Sensitivity Data button will produce a new tool for examining the variation across all runs of the PIAP model.
Importing new data

Upon clicking the Import New Data button, the program loops through the runs and promotion rules, opening each spreadsheet produced by the PIAP model and saving a new file for each run. We now examine the result of a single run. The YOS_PG Chart shows the number of sailors with a GSE$^{12}$ rating in each year of service by stacked paygrade in the first year using the Benchmark Rule.

Figure 9. Years of service by paygrade

---

$^{12}$ Gas Turbine System Technician, Electrical
Figure 10. Time in Service / Time in Grade

Figure 10, shows the average Time in Service and Time in Grade for GSEs year by year, in each paygrade, and for each promotion rule. Paygrade and year are along the x-axis, and the dropdown boxes provide a way to examine the data in more detail by deselecting values in any field. For example, the user can uncheck the 3, 4, 8, and 9 boxes to see just E5-E7.

The Prom Chart in figure 11 shows the percentage of each paygrade that promotes in each year under each promotion rule. The numerators in these percentages are the number of individuals in each paygrade that promoted in that year, and the denominator includes the number in that paygrade at the beginning of the year plus the accessions to E3. Thus, the individuals not promoting include separations. A comparison with figure 13, the Separation chart, gives an indication of what percentage in each paygrade and year failed to promote, either through lack of need or by not being eligible due to Time in Service or Time in Grade requirements.
Figure 12, the Shortage chart, tells us that the PIAP model predicts we will have a shortage of 42 E6 GSEs in the fourth year under the Junior Rule. This is a 17.4 percent gap between personnel and requirements, and it indicates a significant problem will occur in the future unless steps are taken. Since the model accesses to end strength and not simply to fill the E3 billets, we see overages (represented by negative shortages) in the first and fourth year due to the gaps for E4 and E6, respectively. The variation from zero in the second and third years for E3s is the result of imperfectly predicting first-year attrition for the new accessions.

In this example, when all ten runs are compiled (as described later), the average gap for E6s in the fourth year is 33.7 (approximately 14 percent).

These variations from requirements are in the 1 to 2 percent range.
Figure 13. Sep Chart
The Sep chart shows loss rates year by year, in each paygrade, and for each promotion rule. The unexpectedly large E9 attrition in the first year is reflective of both the small number of E9s in this rating and the large percentage of E9s in our data that are just reaching retirement eligibility in the first year of the model. All of our E9s and nearly three-fourths of our E8s are eligible to retire.

The TIS\textsuperscript{15} Chart gives the user a view, by paygrade and year, of time to paygrade, in months, at the time of promotion for those who promoted in that year. The stacked bars show time for the Junior, Benchmark, and Senior Promotion Rules in blue, dark blue, and light blue, respectively.

\textbf{Figure 14.} TIS Chart

\textsuperscript{15} Time in Service
Note the bars for E6s in the second year that extend below the 0 months line. This is not, of course, negative months; these times are relative to that of the Junior Rule. In this case, the Junior Rule yielded a mean of 142 months to E6, 130 months\footnote{-12 relative to the Junior Rule} for the Benchmark Rule, and 105\footnote{-37 relative to the Junior Rule} months for the Senior Rule.

How can promoting older sailors result in a lower mean time to promotion than promoting younger sailors? This seeming anomaly sometimes results in the out years. In the first year under the Junior Rule, all of the younger sailors are promoted, leaving the older ones and those not yet eligible. Likewise, using the Senior Rule, all of the older sailors are promoted, leaving the younger ones and those not yet eligible. If there are relatively few E5s that become eligible for promotion in the next year, the reservoir of older sailors previ-
ously passed over using the Junior Rule must be promoted, increasing the average Time in Service. Again, likewise, all of the younger sailors previously passed over using the Senior Rule must be promoted, decreasing the average Time in Service.

Figure 15. Likelihood chart

The Likelihood chart differs from all of the other outputs in that it deals only with the current inventory. The model tracks the individuals in the Yr0 table through the years and finds their first promotion. In this example, we see that of our initial E5 cohort with 6 years of service, 44 percent promoted to E6 in the first year, 20 percent in the second, and 26 percent in the third using the Junior Rule. Under the Senior Rule, none promoted in the first year, 60 percent in the second, and 28 percent in the third.
**Compiling multiple runs**

In order to smooth out the results and reduce the variation that will occur among the individual runs, they should be compiled into a single file and averaged. The resulting file is named Compiled_rating_runs_Files_years_Years.xls. It contains the same charts and data as the files for the individual runs, but it also has statistics for minimums, maximums, and standard deviations.

**Sensitivity data**

As previously noted, PIAPM.xls is fully functional under Excel 2003; however, a particular setting may need to be adjusted to allow the Sensitivity program to run. On the Menu Bar, under Tools → Macro → Security, the user should click on the Trusted Publishers tab and check the “Trust access to Visual Basic Project” checkbox. This security feature is disabled by default in Excel 2003.

To produce a new file in order to examine the robustness of the model, the user should open the file created by clicking the Compile button and click the Sensitivity Data button. Clicking this button **while the previously created compiled data file is active** will provide the user with a new tool to examine the robustness of the PIAP model, and it will help the user determine whether there is too much variation among the runs to be useful. If the user concludes this to be the case, he/she may try increasing the number of runs in the model. If this does not improve the results, it may be that the number of individuals in that rating is too small to model reliably. This new file will be named Sensitivity_Data_for_rating_runs_Files_years_Years.xls. The sensitivity data are presented in the form of bar charts with indicators for mean, minimum, maximum, and plus or minus one standard deviation.
In Figure 16, we see the variation in promotion rates for E4s in each year under each promotion rule. To see the promotion rates for other paygrades, the user should click the Open Chart button at the top right of the screen, shown in Figure 16, to open a context sensitive form and explore the data. As shown in Figure 17, the user should select the desired inputs and click the Create Chart button to add a new chart for comparison or for export. Checking the “Delete all old charts” box will leave just the new chart, making the file size much smaller; while leaving it unchecked makes it possible to create a large number of charts for examination or for copying and pasting into another application. The user must close the form to examine another sheet. Since the form is context sensitive in the sense that its controls depend on which worksheet is active when the Open Chart button is clicked, it is created as modal so the

---

A modal form or window requires the user to take some action before continuing. It may be operating-system wide (i.e. not allowing the user to use any application) or it may be specific to a particular application.
user cannot do anything else in Excel while the form is open; otherwise, errors may occur. If the user wishes to change this behavior and is familiar with Visual Basic for Applications (VBA) or some other integrated development environment (IDE), he/she can set the form’s ShowModal property to False in the Visual Basic Editor.

Figure 17. Open Chart Form button
The charts produced by PIAPM.xls provide a quick, graphical glimpse into the data output by the PIAP model, but a thorough analysis requires a rich dataset. As such, the data behind the charts are provided and transparent for both the compiled outputs and on the individual run level, as are the pivot tables behind the charts. For a more in-depth analysis, the user can access the yearly tables in PIAPM.mdb and examine the data at the individual level.
The model is user configurable

The PIAP model was designed to be versatile and scalable. The programming code modules for Yr0.mdb, PIAPM.mdb, and PIAPM.xls are unprotected and available to the user for additions and adjustments. If the user wishes to substitute updated source data, he/she should provide it in the format described earlier for SourceData.mdb and prepare it for the model using the macros in Yr0.mdb. Due to the change in the Navy’s E5 High Year Tenure rules and their grandfathering of older sailors, there is a line in the Main module of PIAPM.mdb that must be changed. The line is near the top of the module, in the Global Variables section, above the Main subroutine. The line is

'Months since 7/1/2005 to 10/1/2007 (our current data)
Const E5ADJ = 27

and must be changed to reflect the number of elapsed months since July 2005, the beginning of the grandfather clause.
This page intentionally left blank.
Appendix A: Yr0.mdb programming code

Table of contents

**MakeData**
Function MakeYr0
Private Function GetHighPGRatios
Function DistributeHighPGs
Function MakeRealYr0
Private Sub GetPGRateCount
Private Sub Attrition
Function RefreshLinks
Private Function GetTableName
Private Function HigherPath
Sub XportMods
Declare MakeData

Option Compare Database
Option Explicit

Private sngPGCount(8) As Single, strPG() As String

Function MakeYr0()
Dim rating As String, LowHighPct As Single, LowPG As Byte, HighPG As Byte

rating = UCase(InputBox("Enter Rating"))
DoCmd.SetWarnings False

LowHighPct = GetHighPGRatios(rating)

'Select records on rate
DoCmd.RunSQL "SELECT RealYr0.ssn, RealYr0.pg, " &
& "RealYr0.rate, RealYr0.yos, RealYr0.yig, " &
& "RealYr0.months, RealYr0.mos_pg, 0 AS drop INTO Yr0 FROM RealYr0 " &
& "WHERE ((RealYr0.rate)=""" & rating & """ & rating & """ OR ((RealYr0.rate)=""" &
& Left(rating, 2) & """);"

'Change E1s & E2s to E3
DoCmd.RunSQL "UPDATE Yr0 SET Yr0.pg = 3 WHERE ([Yr0].[pg]<3);"

'Drop bad records
DoCmd.RunSQL "DELETE [Yr0].[pg], [Yr0].[rate], " &
& "[Yr0].[yos], [Yr0].[yig], [Yr0].[months], [Yr0].[mos_pg]" &
& "FROM Yr0 WHERE ((([Yr0].[pg]) Is Null)) Or ((([Yr0].[rate]) Is Null)) " &
& "Or ((([Yr0].[yos]) Is Null)) Or ((([Yr0].[months]) Is Null)) " &
& "Or ((([Yr0].[mos_pg]) Is Null));"

'Drop if TIG>TIS
DoCmd.RunSQL "DELETE Yr0.pg, Yr0.yos, Yr0.yig FROM Yr0 WHERE (Yr0.yig > [yos]);"

'PGs & ratio for compressed ratings
LowPG = Int(LowHighPct / 10)
HighPG = Int(LowHighPct) Mod 10
LowHighPct = LowHighPct - Int(LowHighPct)

'Check for bad data
If HighPG - LowPG <> 1 Then
    MsgBox "There is a problem with the PG distribution for rating " & rating &
    " & vbCrLf & "The high PG for the rating is " & HighPG &
    " but the low PG for " & Left(rating, 2) & " is " & LowPG &
    Exit Function
End If

DistributeHighPGs HighPG, LowHighPct

'Change rate for high PGs
DoCmd.RunSQL "UPDATE Yr0 SET Yr0.rate = " & rating &
End Function
```
GetPGRateCount
Attrition rating, LowPG, HighPG
DoCmd.SetWarnings True
End Function

Private Function GetHighPGRatios(rt As String) As Single
Dim rt2 As String, recs As Long, MinHigh As Byte, MaxLow As Byte
Dim rs As DAO.Recordset
rt2 = Left(rt, 2)

'Get records into temporary table
DoCmd.RunSQL "SELECT RealYr0.pg, RealYr0.rate INTO " & rt
    & " FROM RealYr0 WHERE Left(RealYr0.rate,2) = " & rt2
    & " AND RealYr0.rate <> " & rt2 & ";"
Set rs = CurrentDb.OpenRecordset(rt)
GetHighPGRatios = rs.RecordCount

'Get number in rating
DoCmd.RunSQL "SELECT * into tmp FROM " & rt & " WHERE " & rt
    & ".rate=" & rt2 & ";"
Set rs = CurrentDb.OpenRecordset("tmp")
'Ratio is to right of decimal
GetHighPGRatios = rs.RecordCount / GetHighPGRatios

'Find highest PG in rating
Set rs = CurrentDb.OpenRecordset("SELECT Max(" & rt & ".pg) As pg FROM " & rt & ";")
rs.MoveNext
'High PG of uncompressed in tens place
GetHighPGRatios = GetHighPGRatios + 10 * rs.Fields("pg")
Set rs = Nothing

'Find lowest PG in 2-character rating
DoCmd.RunSQL "SELECT RealYr0.pg, Count(RealYr0.months) AS cnt INTO " & rt & " FROM RealYr0 WHERE RealYr0.rate = " & rt2
    & " GROUP BY " & rt2 & ";"
Set rs = CurrentDb.OpenRecordset("SELECT Min(" & rt & ".pg) As pg FROM " & rt & ";")
rs.MoveNext
'Low PG of compressed in ones place
GetHighPGRatios = GetHighPGRatios + rs.Fields("pg")
Set rs = Nothing

DoCmd.DeleteObject acTable, rt
```
Function DistributeHighPGs(HighPG As Byte, pct As Single)
    Dim rs As DAO.Recordset
    Randomize
    Set rs = CurrentDb.OpenRecordset("Yr0")
    With rs
        .MoveFirst
        Do Until .EOF
            'Select High PG records to delete
            If .Fields("pg") >= HighPG And Rnd > pct Then
                .Edit
                .Fields("drop") = 1
                .Update
            End If
            .MoveNext
        Loop
    End With
    DoCmd.RunSQL "DELETE Yr0.drop FROM Yr0 WHERE (Yr0.drop = 1);"
    Set rs = Nothing
End Function

Function MakeRealYr0()
    DoCmd.SetWarnings False
    DoCmd.RunSQL "SELECT SourceData.ssn, CInt(Right(Trim([grade]),1)) AS pg, " 
    & "SourceData.rate, SourceData.yos, SourceData.yig, SourceData.mos AS months, " 
    & "SourceData.mig AS mos_pg INTO RealYr0 FROM SourceData;"
    DoCmd.RunSQL "DELETE RealYr0.yos FROM RealYr0 WHERE (RealYr0.pg = 0);"
    DoCmd.SetWarnings True
End Function

Private Sub GetPGRateCount()
    'Create temp table with the number in each PG
    DoCmd.RunSQL "SELECT Yr0.pg, Count(Yr0.months) AS cnt INTO PGRollup " _
    & "FROM Yr0 GROUP BY Yr0.pg ORDER BY Yr0.pg;"
    'Create temp table with the number in rating
    DoCmd.RunSQL "SELECT Yr0.rate, Count(Yr0.months) AS cnt INTO RateRollup " _
    & "FROM Yr0 GROUP BY Yr0.rate ORDER BY Yr0.rate;"
End Sub

Private Sub Attrition(rt As String, rtPG As Byte, rt2PG As Byte)
    Dim rt2
    rt2 = Left(rt, 2)
    'Select compressed & uncompressed
    DoCmd.RunSQL "SELECT * INTO AttrRates FROM RealAttrRates WHERE " _
& "((RealAttrRates.rate="" & rt & ") OR (RealAttrRates.rate="" _
& rt2 & ");"

'Delete high PG for uncompressed
DoCmd.RunSQL "DELETE AttrRates.rate FROM AttrRates WHERE ((AttrRates.rate = "" _
& rt & ") AND (AttrRates.pg > " & rtPG & ");"

'Delete low PG for compressed
DoCmd.RunSQL "DELETE AttrRates.rate FROM AttrRates WHERE ((AttrRates.rate = "" _
& rt2 & ") AND (AttrRates.pg < " & rt2PG & ");"

'Make rate consistant
DoCmd.RunSQL "UPDATE AttrRates SET AttrRates.rate = "" & rt 
& "" WHERE ((AttrRates.rate) = "" & rt2 & ");"

End Sub

Function RefreshLinks()
Dim dbs As Database, tdf As TableDef
Dim CurPath As String, TblName As String
CurPath = CurrentProject.Path
' Loop through all tables in the database.
Set dbs = CurrentDb
For Each tdf In dbs.TableDefs
' If the table has a connect string, it’s a linked table.
If Len(tdf.Connect) > 0 Then
    TblName = GetTableName(tdf.Connect)
    tdf.Connect = "DATABASE=" & CurrentProject.Path & "\" _
& TblName
    Err = 0
    On Error Resume Next
    tdf.RefreshLink ' Relink the table.
    ' Can’t find the file, so search up the path
    If Err <> 0 Then
        Do
            CurPath = HigherPath(CurPath)
            Err = 0
            tdf.Connect = "DATABASE=" & CurPath & "\" _
& TblName
            tdf.RefreshLink
        Loop While Err <> 0 And Len(CurPath) > 2
    End If
    If Err <> 0 And tdf.Name = "Yr0" Then
        MsgBox Err.Description
End If

Function GetTableName(Connect)
    Dim TblName
    TblName = Split(Connect, ";")[1]
    GetTableName = TblName
End Function

Function HigherPath(CurPath)
    Dim DirPath
    DirPath = Split(CurPath, "\")_
& "\")
    HigherPath = DirPath & "\")
End Function

37
RefreshLinks = False
End If
End If
Next tdf
End Function
Private Function GetTableName(OldPath As String) As String
'Get table name from full path & file name
Dim bytSlash As Byte
Do
  bytSlash = InStr(OldPath, "\")
  OldPath = Mid(OldPath, bytSlash + 1)
Loop Until bytSlash = 0
GetTableName = OldPath
End Function
Private Function HigherPath(OldPath) As String
'Returns path of parent directory
HigherPath = Left(OldPath, InStrRev(OldPath, "\") - 1)
End Function
Sub XportMods()
  Dim mdl As Variant, strFile As String, strExt As String
  For Each mdl In Application.VBE.ActiveVBProject.VBComponents()
    strFile = ".bas"
    If Left(mdl.Name, 5) = "Form_" Then strFile = ".cls"
  Next
  Set mdl = Nothing
End Sub
Appendix B: PIAPM.mdb programming code

Table of contents

Main
Sub Driver
Sub DoYears
Private Function E9s
Private Sub E9Sep
Private Function ENs
Private Sub ENSep
Private Sub NewE1s
Private Sub AddSep
Private Sub AddProm
Private Function NewAccess
Private Function CalcByoptPers
Private Function Pred

Preliminaries
Sub MakeGuysTable
Sub MakeTables
Private Sub MakeSepTable
Private Sub MakeShortTable
Private Sub MakePromTable
Private Sub MakeTISTIGTable
Private Sub MakeYOS
Private Sub MakeCommonFields
Sub MakeMetaTable
Sub GetTargets
Sub ChangeTargets
Function GetOccCount
Function GetNewGuySepRate

Stats
Public Sub CompileData
Public Sub GetDataForProbs
Public Sub CalcProbs
Public Sub Expected

Utilities
Sub ResetSeed
Sub KillTables
Sub KillReportTables
Sub SQL
Function RefreshLinks
Private Function GetTableName
Private Function HigherPath
Public Sub FeedMeta
Sub KillXL
Sub MakeRunDirs
Function Maximum
Sub XportMods
Private Sub PrntTrgts

Controller form
Private Sub cmdIncDec
Private Sub cmdKill
Private Sub cmdRun
Private Sub cmdKillXL
Private Sub cmdXport

Personnel form
Private Sub cmdUse
Public Sub AssignPers

Manpower form
Private Sub cmdUse
Public Sub AssignMan
Private Sub lblocc0
Private Sub lblocc1
Private Sub lblocc2
Private Sub lblocc3
Private Sub lblocc4
Private Sub CopyAcross
Main

Attribute VB_Name = "Main"
'Programming by Robert W. Shuford, CNA
Option Compare Database
Option Explicit
Option Base 0
'Global variables
Public Const NUMRULES = 3
Const E5ADJ = 27 'Months since 7/1/2005 to 10/1/2007 (our current data)
Public strOcc As String, rs As DAO.Recordset
Public lngTarget(9) As Long, lngOccCount() As Long, strOccArray() As String
Public lngAllOccsPers As Long, occ As Byte, lngCurrCnt As Long
Dim strRule(NUMRULES) As String, bytWarnLevel As Byte, lngAccess As Long, AccumulatedShortage As Long
'For Manpower form
Public frameRules As Byte, optManChange As Byte, min_tig(8) As Integer, min_tis(8) As Integer
Public sngIntDec(8, 6) As Single, optType As Byte, sngLossChange(9) As Single
Public bytStopYr(9) As Byte
'For Personnel form
'1 time, permanent, constant change option
Public optPersChange As Byte
'Number or percent to change--array is by occ
Public sngPers As Single, sngIntDecPers(9) As Single
'Number or percentage by occ option
Public optPers As Byte
Dim E1Sep As Single

Sub Driver(Run As Byte, yrs As Byte) ', MinYIG As Variant)
    Dim i As Byte, ruleocc As String
    Dim varReturn As Variant
    DoCmd.SetWarnings False
    KillTables
    MakeTables
    strOcc = "0"
    lngAllOccsPers = GetOccCount
    strRule(1) = "J"
    strRule(2) = "M"
    strRule(3) = "S"
    'Get form data in case Use buttons not clicked
    [Form_Increase Decrease Personnel].AssignPers False
    [Form_Increase Decrease Manpower Targets].AssignMan
    MakeMetaTable yrs ', MinYIG
    For occ = 0 To UBound(strOccArray())
}
strOcc = strOccArray(occ)
E1Sep = GetNewGuySepRate
For frameRules = 1 To 3
    MakeGuysTable
    'Reset random number generator to be as consistent as possible
    ResetSeed
    GetTargets
    DoYears occ, yrs, frameRules
    Expected Run, yrs, strOcc & strRule(frameRules)

    'Rename tables, prepending rule & occ
    ruleocc = strRule(frameRules) & strOcc
    For i = 1 To yrs
        DoCmd.Rename ruleocc & "Yr" & i, acTable, "Yr" & i
        DoCmd.Rename ruleocc & "EYr" & i, acTable, "EYr" & i
    Next
    DoCmd.Rename ruleocc & "AllYrs", acTable, "AllYrs"
    DoCmd.Rename ruleocc & "Expected", acTable, "Expected"
    DoCmd.Rename ruleocc & "Likelihood", acTable, "Likelihood"
    Next
    DoCmd.Rename "Sep", acTable, "SepData"
    For Each varReturn In Array("Sep", "Shortage", "Metadata", "Prom", _
    "TISTIG", "YOS_PG")
        DoCmd.TransferSpreadsheet acExport, 8, varReturn, CurrentProject.Path _
        & "\" & Run & "\" & varReturn & ".xls", True, ""
    Next
    KillReportTables
    '    varReturn = SysCmd(acSysCmdClearStatus)
    DoCmd.SetWarnings True
End Sub

'Loop through years and PGs
Sub DoYears(occ As Byte, MaxYear As Byte, rule As Byte)
    Dim i As Integer, bytYr As Byte
    Dim lngOldTarget As Long, lngNeed As Long
    ChangeTargets occ
    'Add Yr0 to TIS/TIG table
    SQL "INSERT INTO TISTIG ( rate, rule, yr, pg, cnt, TIS, " _
    & " TIG ) SELECT Yr0.rate AS rate, ""
    & strRule(rule) & "" AS rule, 0 AS yr, Yr0.pg, " _
    & "Count(Yr0.ssn) AS cnt, Avg(Yr0.months) AS TIS, " _
    & "Avg(Yr0.mos_pg) AS TIG FROM Yr0 GROUP BY Yr0.rate, Yr0.pg;"
'Loop through years
For bytYr = 0 To MaxYear - 1
  AccumulatedShortage = 0
  'Increase or decrease manpower
  Select Case optManChange
    '1-time change
    Case 1
      If bytYr = 1 Then GetTargets
    'Permanent change
    Case 2
    'Change every year
    Case 3
      If bytYr > 0 Then ChangeTargets CByte(occ) Mod 10, bytYr
  End Select
  'How many E8s do we need to promote
  lngNeed = E9s(bytYr)
  'Loop through paygrades
  For i = 8 To 3 Step -1
    lngNeed = ENs(lngNeed, i, bytYr, rule)
  Next

  'Create yearly tables for individual data
  SQL "UPDATE Yr" & bytYr + 1 & " SET Yr" & bytYr + 1 & ".prom_mnths = Null " _
  & "WHERE ((Yr" & bytYr + 1 & ".prom_mnths)=0);"
  SQL "UPDATE Yr" & bytYr + 1 & " SET Yr" & bytYr + 1 & ".target = " _
  & lngTarget(3) & " WHERE ((Yr" & bytYr + 1 & ".pg)=3);"
  'Update Shortage table
  SQL "INSERT INTO Shortage ( rate, rule, yr, pg, cnt, target, " _
  & "shortage ) SELECT Yr" & bytYr + 1 & " .rate AS rate, "," _
  & strRule(rule) & "," AS rule, " & bytYr + 1 & " AS yr, Yr" _
  & bytYr + 1 & "," AS pg, Count(Yr" & bytYr + 1 & ".ssn) AS cnt, Yr" _
  & bytYr + 1 & "," AS target, Yr" & bytYr + 1 & "," & target=Count(Yr" _
  & bytYr + 1 & ".ssn) AS shortage " _
  & "GROUP BY Yr" & bytYr + 1 & " .rate, Yr" & bytYr + 1 & "," AS pg, Yr" _
  & bytYr + 1 & "," AS target;"
  'Update TIS/TIG table
  SQL "INSERT INTO TISTIG ( rate, rule, yr, pg, cnt, TIS, " _
  & " TIG ) SELECT Yr" & bytYr + 1 & " .rate AS rate, "," _
  & strRule(rule) & "," AS rule, " & bytYr + 1 & " AS yr, Yr" _
  & bytYr + 1 & "," AS pg, Count(Yr" & bytYr + 1 & ".ssn) AS cnt, " _
& "Avg(\text{Yr} \& \text{bytYr} + 1 \& \text{.months}) \text{ AS TIS, } \_\_
& "Avg(\text{Yr} \& \text{bytYr} + 1 \& \text{.mos\_pg}) \text{ AS TIG } \_\_
& "FROM \text{Yr} \& \text{bytYr} + 1
& " GROUP BY \text{Yr} \& \text{bytYr} + 1 \& \".rate, \text{Yr} \& \text{bytYr} + 1 \& \".pg;"

'Update YOS\_PG table
SQL "INSERT INTO YOS\_PG ( rate, rule, yr, pg, yos, cnt ) SELECT " \_
& \text{strOcc} \& \" AS rate, " \_
& \text{strRule}(\text{rule}) \& \" AS rule, " \_
& \text{bytYr} + 1 \& \" AS yr, " \_
& \text{bytYr} + 1 \& \".pg, \text{Yr} \& \text{bytYr} + 1 \& \".yos, \text{Count(\text{Yr}) } \_
& \text{bytYr} + 1 \& \".ssn) \text{ AS cnt } \& " \_
& "FROM \text{Yr} \& \text{bytYr} + 1
& " GROUP BY \text{Yr} \& \text{bytYr} + 1 \& \".pg, \text{Yr} \& \text{bytYr} + 1 \& \".yos;"

Next
'Create AllYrs
CompileData MaxYear
'Prepare data for Likelihood and Expected tables
GetDataForProbs MaxYear
'Create Likelihood and Expected tables
CalcProbs MaxYear, \text{strRule}(\text{rule})
DoCmd.DeleteObject acTable, \"temp\"
End Sub

'Handle E9s separately since they don't promote—only separate for speed
Private Function E9s(\text{yr As Byte})
'Create temp table with E9 data for occ
SQL "SELECT \text{Yr} \& \text{yr} \& \".ssn, \text{Yr} \& \text{yr} \& \".pg, \text{Yr} \& \text{yr} \& \".rate, " \_
& \"\text{Yr} \& \text{yr} \& \".months, \text{Yr} \& \text{yr} \& \".yig, \text{Yr} \"
& \text{yr} \& \".yos, \text{Yr} \& \text{yr} \& \".mos\_pg, \text{AttrRates.prob\_sep}, " \_
& \"\text{O AS prom\_mnths, } \_\& \text{lngTarget(9) & \" AS target "
& \"\text{INTO temp FROM Yr} \& \text{yr} \& \" INNER JOIN \text{AttrRates } \_\_
& \"\text{ON (Yr} \& \text{yr} \& \".yos = \text{AttrRates.yos) } \_
& \"\text{AND (Yr} \& \text{yr} \& \".rate = \text{AttrRates.rate})\_\_
& \"\text{AND (Yr} \& \text{yr} \& \".pg = \text{AttrRates.pg}) } \_
& \"\text{WHERE (Yr} \& \text{yr} \& \".rate=\"\& \text{strOcc} \& \"}) } \_
& \"\text{AND (((Yr} \& \text{yr} \& \".pg)=9) } \_
& \"\text{ORDER BY Yr} \& \text{yr} \& \".months;"

'Change attrition rates
SQL "UPDATE temp SET temp.prob\_sep = prob\_sep * (1 + " \& \text{sngLossChange(9) } \_
& ") WHERE prob\_sep <> 1;"

'Separate guys
E9Sep yr
'Calculate needs
E9s = lngTarget(9) - rs.RecordCount
Set rs = Nothing
'Load E9s into new YrX table
SQL "SELECT temp.ssn, temp.rate, temp.pg, temp.months, temp.yig, temp.yos, " 
& "temp.mos_pg, temp.prom_mnths, temp.target " 
& "INTO Yr" & yr + 1 & " from temp " 
& "ORDER BY temp.months;"
End Function
Private Sub E9Sep(yr As Byte)
Dim losses As Long, sep As DAO.Recordset
losses = 0
Set rs = CurrentDb.OpenRecordset("temp")
'Go through each record and either separate or age
With rs
  .MoveFirst
  Do Until .EOF
    .Edit
    'Separate
    If Rnd() < !prob_sep Then
      !target = Null
      losses = losses + 1
    Else
    'Age
      !yos = !yos + 1
      !yig = !yig + 1
      !months = !months + 12
      !mos_pg = !mos_pg + 12
    End If
    .Update
  .MoveNext
  Loop
End With
AddSep yr + 1, 9, losses, rs.RecordCount
'Delete seps
SQL "DELETE temp.target FROM temp WHERE ((temp.target) Is Null);"
End Sub
Private Function ENs(Need As Long, pg As Integer, yr As Byte, rule As Byte)
Dim lngProms As Long, strRule(3) As String, Benchmarks As Variant
strRule(1) = "ASC"
strRule(3) = "DESC"
Benchmarks = Split("0 0 2.2 4.4 9 14.8 18.5 22.2")
For lngProms = 0 To 8
  Benchmarks(lngProms) = Benchmarks(lngProms) * 12
Next
lngProms = 0
Set rs = Nothing

If rule = 2 Then
  SQL "SELECT Yr" & yr & ".ssn, Yr" & yr & ".pg, Yr" & yr & ".rate, " _
  & "Yr" & yr & ".months, Yr" & yr & ".yig, Yr" _
  & yr & ".yos, Yr" & yr & ".mos_pg, AttrRates.prob_sep, " _
  & "0 AS prom_mnths, " & lngTarget(pg) & " AS target, ABS(Yr" _
  & yr & ".months + 12 - " & Benchmarks(pg) & ") AS bm " _
  & 
  INTO temp FROM Yr" & yr & " INNER JOIN AttrRates " _
  & "ON (Yr" & yr & ".yos = AttrRates.yos) " _
  & "AND (Yr" & yr & ".rate = AttrRates.rate) " _
  & "AND (Yr" & yr & ".pg = AttrRates.pg) " _
  & "WHERE (Yr" & yr & ".rate=" & strOcc & ")") _
  & "AND ((Yr" & yr & ".pg)=" & pg & ") " _
  & "ORDER BY ABS(Yr" _
  & yr & ".months + 12 - " & Benchmarks(pg) & ") " & strRule(1) & "" _
  & 
  SQL "UPDATE temp SET temp.prob_sep = prob_sep * (1 + " & sngLossChange(pg) _
  & ") WHERE prob_sep <> 1;"
'Change attrition rates

'Maintain correct sorting
SQL "CREATE INDEX kKey on temp (bm ASC, ssn)"

Else

SQL "SELECT Yr" & yr & ".ssn, Yr" & yr & ".pg, Yr" & yr & ".rate, " _
  & "Yr" & yr & ".months, Yr" & yr & ".yig, Yr" _
  & yr & ".yos, Yr" & yr & ".mos_pg, AttrRates.prob_sep, " _
  & "0 AS prom_mnths, " & lngTarget(pg) & " AS target " _
  & "INTO temp FROM Yr" & yr & " INNER JOIN AttrRates " _
  & "ON (Yr" & yr & ".yos = AttrRates.yos) " _
  & "AND (Yr" & yr & ".rate = AttrRates.rate) " _
  & "AND (Yr" & yr & ".pg = AttrRates.pg) " _
  & "WHERE (Yr" & yr & ".rate=" & strOcc & "")") " _
  & "AND ((Yr" & yr & ".pg)=" & pg & ") " _
  & "ORDER BY Yr" & yr & ".months " & strRule(rule) & "", Yr" _
  & 
  SQL "UPDATE temp SET temp.prob_sep = prob_sep * (1 + " & sngLossChange(pg) _
  & ") WHERE prob_sep <> 1;"
'Change attrition rates

'Adjust attrition rates for change in E5 HYT
If pg = 5 Then
  SQL "UPDATE temp SET temp.prob_sep = 1 WHERE temp.months < " _
& 120 + E5ADJ + 12 * yr & " AND temp.yos >= 14;"

End If
'Maintain correct sorting
SQL "CREATE INDEX kKey on temp (months " & strRule(rule) & ", ssn)"
End If

Set rs = CurrentDb.OpenRecordset("temp")
'New accessions
lngCurrCnt = rs.RecordCount
If pg = 3 Then
  lngAccess = NewE1s yr, Need
  Set rs = CurrentDb.OpenRecordset("temp")
End If
'rs.Index = "kKey"
ENSEp yr, pg
'Calculate needs
ENS = lngTarget(pg) - rs.RecordCount
Set rs = Nothing

'Promote
'Age months of service
SQL "UPDATE temp SET temp.months = [months] + 12, temp.yig = [yig] + 1, " _
  & "temp.yos = [yos] + 1, temp.mos_pg = [mos_pg] + 12;"
Set rs = CurrentDb.OpenRecordset("temp")
'Maintain correct sorting
rs.Index = "kKey"

AccumulatedShortage = AccumulatedShortage + Need
With rs
  .MoveFirst
  Do Until .EOF Or Need <= 0
    If !months >= min_tis(pg - 1) And !mos_pg >= min_tig(pg - 1) Then
      'Promote
      .Edit
      !pg = pg + 1
      !yig = 0
      !mos_pg = 0
      lngProms = lngProms + 1
      !prom_mnths = !months
      !target = lngTarget(pg + 1)
    End If
  End If
  .MoveNext
  Do Until .EOF Or Need <= 0
    If !months >= min_tis(pg - 1) And !mos_pg >= min_tig(pg - 1) Then
      'Promote
      .Edit
      !pg = pg + 1
      !yig = 0
      !mos_pg = 0
      lngProms = lngProms + 1
      !prom_mnths = !months
      !target = lngTarget(pg + 1)
    End If
  End If
  Need = Need - !months
  .MoveNext
  Do Until .EOF Or Need <= 0
    If !months >= min_tis(pg - 1) And !mos_pg >= min_tig(pg - 1) Then
      'Promote
      .Edit
      !pg = pg + 1
      !yig = 0
      !mos_pg = 0
      lngProms = lngProms + 1
      !prom_mnths = !months
      !target = lngTarget(pg + 1)
    End If
  End If
End With
.Update
   Need = Need - 1
End If
.MoveNext
Loop
End With
Set rs = Nothing
AccumulatedShortage = AccumulatedShortage - lngProms
If pg = 3 Then ENs = lngCurrCnt Else ENs = ENs + lngProms
'Append to YrX table
   SQL "INSERT INTO Yr & yr + 1 & " (ssn, pg, rate, months, " _
   & "yig, yos, mos_pg, prom_mnths, target) " _
   & "SELECT temp.ssn, temp.pg, temp.rate, temp.months, " _
   & "temp.yig, temp.yos, temp.mos_pg, temp.prom_mnths, temp.target " _
   & "FROM temp;"
'Add to Promotion table
   AddProm yr + 1, pg, lngProms, lngCurrCnt
End Function
Private Sub ENSep(yr As Byte, pg As Integer)
   Dim losses As Long, cnt As Long, sep As DAO.Recordset
   losses = 0
   With rs
      .MoveFirst
      'Separate
      Do Until .EOF
         .Edit
         'Go through each record and decide whether to separate
         If Rnd() < !prob_sep Then
            !target = Null
            losses = losses + 1
         End If
         .Update
         .MoveNext
      Loop
   End With
   AddSep yr + 1, pg, losses, lngCurrCnt
   'Delete seps
   SQL "DELETE temp.target FROM temp WHERE ((temp.target) Is Null);"
End Sub
'New accessions
Private Sub NewEls(yr As Byte, promoted As Long)
   Static id As Long

lngAccess = Round((NewAccess(yr) + promoted + Pred + AccumulatedShortage) / (1 - E1Sep))
Dim i As Long
With rs
    For i = 1 To lngAccess
        id = id + 1
        .AddNew
        'Create unique ID
        !ssn = “A” & Format(id, “00000000”)
        !pg = 3
        !Rate = strOcc
        !months = 0
        !yig = 0
        !yos = 0
        !mos_pg = 0
        !prob_sep = E1Sep
        !prom_mnths = Null
        !target = lngTarget(3)
        .Update
    Next
End With
Set rs = Nothing
If yr = 0 Then SQL “INSERT INTO Guys (ssn, pg) SELECT temp.ssn, temp.pg “ & 
& “FROM temp WHERE (temp.ssn Like “*A*”);”
FeedMeta strOcc & strRule(frameRules) & “ Year “ & yr + 1 & “ Accessions”, lngAccess
End Sub
Private Sub AddSep(yr As Byte, pg As Integer, losses As Long, cnt As Long)
    ‘Add to Separation table
    Dim sep As DAO.Recordset
    Set sep = CurrentDb.OpenRecordset(“SepData”)
    With sep
        .AddNew
        .Fields(“rate”).Value = strOcc
        .Fields(“rule”) = strRule(frameRules)
        .Fields(“pg”) = pg
        .Fields(“yr”) = yr
        .Fields(“cnt”) = cnt
        .Fields(“seps”) = losses
        .Fields(“pct”) = losses / cnt
        .Update
    End With
    Set sep = Nothing
Private Sub **AddProm** (yr As Byte, pg As Integer, promotions As Long, cnt As Long)
  'Add to Separation table
  Dim sep As DAO.Recordset
  Set sep = CurrentDb.OpenRecordset("Prom")
  With sep
    .AddNew
    .Fields("rate").Value = strOcc
    .Fields("rule") = strRule(frameRules)
    .Fields("pg") = pg
    .Fields("yr") = yr
    .Fields("cnt") = cnt
    .Fields("proms") = promotions
    .Fields("pct") = promotions / cnt
    .Update
  End With
  Set sep = Nothing
End Sub

Private Function **NewAccess** (yr As Byte) As Long
  Select Case optPersChange
  Case 1 '1-time change
    NewAccess = lngTarget(3) - lngCurrCnt
  Case 2 'Permanent change
    NewAccess = CalcByoptPers(1)
  Case 3 'Constant change
    NewAccess = CalcByoptPers(yr + 1)
  End Select
  If yr = 0 Then
    NewAccess = Maximum(CalcByoptPers(1) + lngCurrCnt - lngTarget(3), NewAccess)
  End If
End Function

Private Function **CalcByoptPers** (yrnum As Byte) As Long
  Select Case optPers
  Case 1 'By number in occ array
    CalcByoptPers = yrnum * sngIntDecPers(0) _
                    + lngTarget(3) - lngCurrCnt
  Case 2 'By percentage in occ array
    CalcByoptPers = Round(lngOccCount(0) _
                          * ((1 + sngIntDecPers(0)) ^ yrnum - 1), 0) _
                    + lngTarget(3) - lngCurrCnt
  Case 3 'By number as global value
    CalcByoptPers = Round(yrnum * sngPers) _
  End Case
End Function
CalcByoptPers = Round(lngOccCount(0) * ((1 + sngPers) ^ yrnum - 1), 0) + lngTarget(3) - lngCurrCnt

Case Else
   CalcByoptPers = lngTarget(3) - lngCurrCnt
End Select

Private Function Pred() As Single
   Dim rs1 As DAO.Recordset
   Set rs1 = CurrentDb.OpenRecordset("SELECT Sum(temp.prob_sep) AS SumOfprob_sep FROM temp;")
   With rs1
      .MoveFirst
      Pred = rs1!SumOfprob_sep
   End With
   Set rs1 = Nothing
End Function
Preliminaries

Attribute VB_Name = “Preliminaries”
‘Programming by Robert W. Shuford, CNA
Option Compare Database
Option Explicit
Sub MakeGuysTable()
    SQL “SELECT Yr0.ssn, Yr0.pg, Yr0.yos INTO Guys FROM Yr0 “ _
    & “WHERE Yr0.rate=’” & strOcc & “’;”
End Sub
Sub MakeTables()
    MakeYOS_PGTable
    MakeSepTable
    MakeShortTable
    MakePromTable
    MakeTISTIGTable
End Sub
Private Sub MakeSepTable()
    Dim tdf As TableDef
    Set tdf = CurrentDb.CreateTableDef(“SepData”)
    With tdf
        MakeCommonFields tdf
        .Fields.Append .CreateField(“pct”, dbDouble)
    End With
    CurrentDb.TableDefs.Append tdf
End Sub
Private Sub MakeShortTable()
    Dim tdf As TableDef
    Set tdf = CurrentDb.CreateTableDef(“Shortage”)
    With tdf
        MakeCommonFields tdf
    End With
    CurrentDb.TableDefs.Append tdf
End Sub
Private Sub MakePromTable()
    Dim tdf As TableDef
    Set tdf = CurrentDb.CreateTableDef(“Prom”)
MakeCommonFields tdf
   .Fields.Append .CreateField("proms", dbDouble)
   .Fields.Append .CreateField("pct", dbDouble)
End With
CurrentDb.TableDefs.Append tdf
End Sub
Private Sub MakeTISTIGTable()
   Dim tdf As TableDef
   Set tdf = CurrentDb.CreateTableDef("YOS_PG")
   With tdf
      MakeCommonFields tdf
      .Fields.Append .CreateField("yos", dbByte)
   End With
   CurrentDb.TableDefs.Append tdf
End Sub
Private Sub MakeYOS_PGTable()
   Dim tdf As TableDef
   Set tdf = CurrentDb.CreateTableDef("TISTIG")
   With tdf
      MakeCommonFields tdf
      .Fields.Append .CreateField("TIS", dbDouble)
      .Fields.Append .CreateField("TIG", dbDouble)
   End With
   CurrentDb.TableDefs.Append tdf
End Sub
Private Sub MakeCommonFields(td As TableDef)
   With td
      .Fields.Append .CreateField("rate", dbText)
      .Fields.Append .CreateField("rule", dbText)
      .Fields.Append .CreateField("pg", dbByte)
      .Fields.Append .CreateField("yr", dbByte)
      .Fields.Append .CreateField("cnt", dbLong)
   End With
End Sub
Sub MakeMetaTable(yrs As Byte) ', AltYIG As Variant)
   'Make new table
   Dim tdf As TableDef
   Set tdf = CurrentDb.CreateTableDef("Metadata")
   With tdf
      .Fields.Append .CreateField("f1", dbText)
      .Fields.Append .CreateField("f2", dbText)
   End With
End With
CurrentDb.TableDefs.Append tdf
Set tdf = Nothing
‘Add info from Controller
FeedMeta "Date", Now
FeedMeta "Years", yrs
FeedMeta "Rating", strOccArray(0)
Set rs = CurrentDb.OpenRecordset("Yr0")
rs.MoveFirst

‘Add info from Personnel
Dim str As String, i As Byte, j As Byte
Select Case optPersChange
Case 1
    str = “1-time change”
Case 2
    str = “Permanent change”
Case 3
    str = “Constant change”
End Select
FeedMeta "Personnel Change Type", str
Select Case optPers
Case 1, 3
    str = “Number”
Case 2, 4
    str = “Percentage”
End Select
FeedMeta "Personnel Change Number Type", str
Select Case optPers
Case 1, 2
    For i = 0 To 8
        FeedMeta "Change", CStr(sngIntDecPers(i))
    Next
    Case 3, 4
        FeedMeta "Personnel Change”, sngPers
End Select

‘Add info from Manpower
Select Case optManChange
Case 1
    str = “1-time change”
Case 2
str = "Permanent change"
Case 3
str = "Constant change"
End Select
FeedMeta "Manpower Change Type", str
Select Case optType
Case 1
str = "Percentage"
Case 2
str = "Number"
End Select
FeedMeta "Manpower Change Number Type", str
For i = 0 To 8
For j = 3 To 9
If sngIntDec(i, j - 3) <> 0 Then FeedMeta _
"E" & j & "" Change", sngIntDec(i, j - 3)
Next
Next
End Sub

'Get number in each grade to be used as a target in each year
Sub GetTargets()
Dim i As Byte
'Create temp table with the number in each grade to be used as a target in each year
SQL "SELECT Yr0.pg, Count(Yr0.pg) AS cnt " _
& "INTO temp FROM Yr0 " _
& "GROUP BY Yr0.pg, Yr0.rate " _
& "HAVING (Yr0.rate="" & strOcc & "") " _
& "ORDER BY Yr0.pg;"

'Load into lngTarget array
Set rs = CurrentDb.OpenRecordset("temp")
rs.MoveFirst
For i = 3 To 9
lngTarget(i) = rs!cnt
rs.MoveNext
Next
Set rs = Nothing
End Sub

'Increase or decrease manpower by the values from the Increase Decrease Targets form
Sub ChangeTargets(oc As Byte, Optional yr As Byte = 0) ' Byte)
Dim i As Byte
Select Case optType
Case 1 'By percentage
For i = 3 To 9
    If yr < bytStopYr(i) Then lngTarget(i) = Round(lngTarget(i) _
        * (1 + sngIntDec(oc, i - 3)), 0)
Next
Case 2 'By number
For i = 3 To 9
    If yr < bytStopYr(i) Then lngTarget(i) = lngTarget(i) _
        + sngIntDec(oc, i - 3)
Next
Case Else
End Select
End Sub
'Get number in each occ
Function GetOccCount() As Long
    Dim i As Byte
    'Load into lngOccCount array
    SQL “SELECT RateRollup.rate, RateRollup.cnt “ _
    & “INTO temp FROM RateRollup “ _
    & “ORDER BY RateRollup.rate;”
    Set rs = CurrentDb.OpenRecordset("temp")
    ReDim lngOccCount(rs.RecordCount - 1)
    ReDim strOccArray(rs.RecordCount - 1)
    rs.MoveFirst
    For i = 0 To UBound(strOccArray())
        strOccArray(i) = rs!Rate
        lngOccCount(i) = rs!cnt
        GetOccCount = GetOccCount + lngOccCount(i)
    Next
    rs.MoveNext
    Next
    Set rs = Nothing
End Function
'Get separation rate for new accessions
Function GetNewGuySepRate() As Single
    Dim i As Byte
    'Create temp table with the number in each occ
    DoCmd.RunSQL "SELECT AttrRates.prob_sep INTO temp FROM AttrRates WHERE " _
        & "((AttrRates.rate="" & strOcc & ")") AND (AttrRates.pg=3) " _
        & "AND (AttrRates.yos=0));"
    Set rs = CurrentDb.OpenRecordset("temp")
rs.MoveNext
GetNewGuySepRate = rs!prob_sep
Set rs = Nothing
End Function
Stats
Attribute VB_Name = "Stats"
'Programming by Robert W. Shuford, CNA
Option Compare Database
Option Explicit
'Create AllYrs table
Public Sub CompileData(MaxYear As Byte)
   Dim qdf As QueryDef, i As Byte

   '1st year
   With CurrentDb
      Set qdf = .CreateQueryDef("b1", "SELECT 1 AS yr, Yr1.pg, Yr1.rate" _
         & ", Avg(Yr1.target) AS target, Count(Yr1.pg) AS cnt " _
         & "FROM Yr1 GROUP BY 1, Yr1.pg, Yr1.rate" _
         & ";")
      Set qdf = .CreateQueryDef("b2", "SELECT Yr1.pg, Avg(Yr1.prom_mnths) " _
         & "AS prom_mnths_ave FROM Yr1 GROUP BY Yr1.pg;")
      SQL "SELECT b1.*, b2.prom_mnths_ave INTO AllYrs " _
         & "FROM b1 INNER JOIN b2 ON b1.pg = b2.pg;"
   End With
   DoCmd.DeleteObject acQuery, "b1"
   DoCmd.DeleteObject acQuery, "b2"

   'Append other years
   For i = 2 To MaxYear
      With CurrentDb
         Set qdf = .CreateQueryDef("b1", "SELECT " & i & " AS yr, Yr" & i _
            & ".rate," & " Yr" & i & ".pg, Avg(Yr" & i _
            & ".target) AS target, " _
            & "Count(Yr" & i & ".pg) AS cnt FROM Yr" & i _
            & " GROUP BY " & i & ", Yr" & i & ".pg, Yr" & i & ".rate" _
            & ";")
         Set qdf = .CreateQueryDef("b2", "SELECT Yr" & i & ".pg, Avg(Yr" & i _
            & ".prom_mnths) AS prom_mnths_ave FROM Yr" & i _
            & " GROUP BY Yr" & i & ".pg;")
         SQL "INSERT INTO AllYrs (prom_mnths_ave) SELECT b1.*, b2.prom_mnths_ave " _
            & "FROM b1 INNER JOIN b2 ON b1.pg = b2.pg;"
      End With
      DoCmd.DeleteObject acQuery, "b1"
      DoCmd.DeleteObject acQuery, "b2"
   Next i
End Sub
Public Sub GetDataForProbs(yrs As Byte)
    ‘Compiles data for Yr0 PG, months cohots
    Dim i As Byte
    Dim strSelect As String, strFrom As String, strWhere As String
    SQL "UPDATE Guys SET Guys.yos = 0 WHERE (Guys.ssn Like "A*")};
    ‘Create SQL statement in segments
    strSelect = "SELECT Guys.ssn, Guys.pg, Guys.yos"
    strFrom = "FROM ">
    For i = 1 To yrs
        strFrom = strFrom & "("
    Next
    strFrom = strFrom & "Guys ">
    For i = 1 To yrs
        strSelect = strSelect & "pg AS pg" & i & ", Yr" & i & _
        & ".prom_mnths AS prom_mnths" & i
        strFrom = strFrom & "LEFT JOIN Yr" & i & " ON Guys.ssn = Yr" & i & _
        & "".ssn) "
    Next
    For i = 1 To yrs
        strSelect = strSelect & "0 AS prom_" & i
    Next
    strSelect = strSelect & "0 AS promoted INTO temp"
    ‘Whew, finally run it
    SQL strSelect & " & strFrom & " & strWhere
    DoCmd.DeleteObject acTable, "Guys"
End Sub
Public Sub CalcProbs(yrs As Byte, rule As String)
    Dim i As Byte, strSQL As String
    Set rs = CurrentDb.OpenRecordset("temp")
    SQL "UPDATE temp SET temp.prom_1 = 1, temp.promoted = 1 " _
    & "WHERE ((temp.prom_mnths1) Is Not Null);"
    For i = 2 To yrs
        SQL "UPDATE temp SET temp.prom_" & i & " = 1, temp.promoted = 1 " _
        & "WHERE (((temp.prom_mnths" & i & ") Is Not Null) " _
        & "AND ((temp.promoted) = 0));"
    Next
    ‘Likelihood
    strSQL = "SELECT " & strOcc & " as rate, " & rule _
    & " as rule, temp.pg, temp.yos, Count(temp.pg) AS cnt"
For i = 1 To yrs
    strSQL = strSQL & "", Avg(temp.prom_" & i & ") AS av_" & i
Next
strSQL = strSQL & ", 0.1111111111111111 AS likelihood INTO Likelihood " 
& "FROM temp GROUP BY temp.pg, temp.yos;"
SQL strSQL
strSQL = "UPDATE Likelihood SET Likelihood.likelihood = Likelihood.av_1"
For i = 2 To yrs
    strSQL = strSQL & " + Likelihood.av_" & i
Next
strSQL = strSQL & ";"
SQL strSQL
"Expected
strSQL = "SELECT temp.pg, temp.yos, Count(temp.pg) AS cnt"
For i = 1 To yrs
    strSQL = strSQL & "", Avg(temp.prom_" & i & ") AS av_" & i
Next
strSQL = strSQL & ", 0.11111 AS 
& "FROM temp GROUP BY temp.pg, temp.yos, temp.promoted " 
& "HAVING ((temp.promoted)=1);"
SQL strSQL
strSQL = "UPDATE Expected SET Expected.likelihood = Expected.yos + Expected.av_1"
For i = 2 To yrs
    strSQL = strSQL & " + (Expected.av_" & i & " & i & " * " & i & ")"
Next
strSQL = strSQL & ";"
SQL strSQL
Set rs = Nothing
End Sub
Public Sub Expected(Run As Byte, NumYrs As Byte, occrule As String)
    Dim qdf As QueryDef, i As Byte
    Dim fso
    Set fso = CreateObject("Scripting.FileSystemObject")
    On Error Resume Next
    On Error GoTo 0
    For i = 1 To NumYrs
        Set qdf = CurrentDb.CreateQueryDef("tmpQry", _
            "SELECT Yr" & i & ".rate, " & ".prom_mnths AS cnt, Avg(Yr" & i 
            & ") AS av_" & i_
& ".prom_mnths) AS AvgOfprom_mnths INTO EYr"
& i & " FROM Yr" & i & " GROUP BY Yr" & i & ".rate, Yr" & i & " pg HAVING ((Count(Yr" & i & ".prom_mnths)) Is Not Null));"
DoCmd.OpenQuery "tmpqry", acNormal, acEdit
DoCmd.TransferSpreadsheet acExport, 8, "EYr" & i, CurrentProject.Path _
& " Expected" & occrule & ".xls", True, ""
DoCmd.DeleteObject acQuery, "tmpQry"
Next
DoCmd.TransferSpreadsheet acExport, 8, "Likelihood", CurrentProject.Path _
& " Likelihood" & occrule & ".xls", True, ""
Set fso = Nothing
Set qdf = Nothing
End Sub
Utilities

Attribute VB_Name = "Utilities"
'Programming by Robert W. Shuford, CNA
Option Compare Database
Option Explicit
'Minor Subs & Utilities
Sub ResetSeed()
    Randomize
End Sub
'Delete individual year tables
Sub KillTables()
    Dim tdf As TableDef
    On Error Resume Next
    For Each tdf In CurrentDb.TableDefs
        If Left(tdf.Name, 2) = "Yr" And tdf.Name <> "Yr0" Then  
            DoCmd.DeleteObject acTable, tdf.Name
        If Left(tdf.Name, 3) = "EYr" Then DoCmd.DeleteObject acTable, tdf.Name
        If Left(tdf.Name, 1) = "J" Then DoCmd.DeleteObject acTable, tdf.Name
        If Left(tdf.Name, 1) = "M" Then DoCmd.DeleteObject acTable, tdf.Name
        If Left(tdf.Name, 1) = "S" Then DoCmd.DeleteObject acTable, tdf.Name
    Next
    DoCmd.DeleteObject acTable, "Guys"
    On Error GoTo 0
KillReportTables
End Sub
'Delete individual year tables
Sub KillReportTables()
    Dim tdf As TableDef
    On Error Resume Next
    For Each tdf In CurrentDb.TableDefs
        If Left(tdf.Name, 4) <> "MSys" Then
            If Mid(tdf.Name, 5, 1) = "A" Then DoCmd.DeleteObject acTable, tdf.Name
            If Mid(tdf.Name, 5, 1) = "D" Then DoCmd.DeleteObject acTable, tdf.Name
            If Mid(tdf.Name, 5, 1) = "E" Then DoCmd.DeleteObject acTable, tdf.Name
            If Mid(tdf.Name, 5, 2) = "Li" Then DoCmd.DeleteObject acTable, tdf.Name
            If Mid(tdf.Name, 1, 2) = "YO" Then DoCmd.DeleteObject acTable, tdf.Name
        End If
    Next
    DoCmd.DeleteObject acTable, "Sep"
    DoCmd.DeleteObject acTable, "Shortage"
DoCmd.DeleteObject acTable, "Prom"
DoCmd.DeleteObject acTable, "TISTIG"
On Error GoTo 0
End Sub

'Shortcut to run SQL
Sub SQL(strSQL As String)
  Dim varReturn As Variant
  Debug.Print strSQL
  DoCmd.RunSQL strSQL
End Sub

Function RefreshLinks()
  Dim dbs As Database, tdf As TableDef
  Dim CurPath As String, TblName As String
  CurPath = CurrentProject.Path
  ' Loop through all tables in the database.
  Set dbs = CurrentDb
  For Each tdf In dbs.TableDefs
    ' If the table has a connect string, it’s a linked table.
    If Len(tdf.Connect) > 0 Then
      TblName = GetTableName(tdf.Connect)
      tdf.Connect = ";DATABASE=" & CurrentProject.Path & "\" _ & TblName
      Err = 0
      On Error Resume Next
      tdf.RefreshLink ' Relink the table.
      ' Can’t find the file, so search up the path
      If Err <> 0 Then
        Do
          CurPath = HigherPath(CurPath)
          Err = 0
          tdf.Connect = ";DATABASE=" & CurPath & "\" _ & TblName
          tdf.RefreshLink
        Loop While Err <> 0 And Len(CurPath) > 2
      End If
      If Err <> 0 And tdf.Name = "Yr0" Then
        MsgBox Err.Description
        RefreshLinks = False
      End If
    End If
  Next tdf
End Function
Private Function GetTableName(OldPath As String) As String
    Dim bytSlash As Byte
    Do
        bytSlash = InStr(OldPath, vbCrLf)
        OldPath = Mid(OldPath, bytSlash + 1)
    Loop Until bytSlash = 0
    GetTableName = OldPath
End Function

Private Function HigherPath(OldPath) As String
    HigherPath = Left(OldPath, InStrRev(OldPath, vbCrLf) - 1)
End Function

Public Sub FeedMeta(f1 As String, ByVal f2 As String)
    DoCmd.RunSQL "INSERT INTO Metadata ( f1, f2 ) SELECT "" & f1 & vbCrLf & vbCrLf & f2 & vbCrLf & vbCrLf AS f1, "" & f2 & vbCrLf & vbCrLf & f2 & vbCrLf & vbCrLf;"
End Sub

Sub KillXL(runs As Byte)
    On Error Resume Next
    Dim Run As Byte, i As Byte, fn As Variant
    fn = Split("Expected* Likelihood* Shortage Sep Metadata Prom TISTIG")
    For Run = 1 To runs
        For i = 0 To UBound(fn)
            Kill CurrentProject.Path & vbCrLf & Run & vbCrLf & fn(i) & vbCrLf.xls"
        Next
    Next
End Sub

Sub MakeRunDirs(runs As Byte)
    On Error Resume Next
    Dim Run As Byte, i As Byte, fn As Variant
    KillXL runs
    For Run = 1 To runs
        MkDir CurrentProject.Path & vbCrLf & Run
    Next
End Sub

Function Maximum(ParamArray Values() As Variant)
    Dim i As Integer
    ' Use UBound function to determine upper limit of array.
    For i = 1 To UBound(Values())
        If Values(i) > Maximum Then Maximum = Values(i)
    Next i
Sub XportMods()
    Dim mdl As Variant, strFile As String, strExt As String
    For Each mdl In Application.VBE.ActiveVBProject.VBComponents()
        strFile = "\bas"
        If Left(mdl.Name, 5) = "Form_" Then strFile = "\cls"
    Next
    Set mdl = Nothing
End Sub

Private Sub PrntTrgts(o As Byte, y As Integer)
    Dim i As Byte
    For i = 3 To 9
        Debug.Print o & " " & y & " " & i & " " & lngTarget(i)
    Next
End Sub
Controller form
VERSION 1.0 CLASS
BEGIN
   MultiUse = -1 'True
END
Attribute VB_Name = "Form_Controller"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = False
'Programming by Robert W. Shuford, CNA
Option Compare Database
Option Explicit
Private Sub cmdIncDec_Click()
   DoCmd.OpenForm "Increase Decrease Personnel", acNormal, "", "", , acNormal
End Sub
Private Sub cmdKill_Click()
   KillTables
   On Error Resume Next
   DoCmd.DeleteObject acTable, "AllYrs"
   DoCmd.DeleteObject acTable, "Likelihood"
   DoCmd.DeleteObject acTable, "Expected"
   DoCmd.DeleteObject acTable, "Temp"
   DoCmd.DeleteObject acTable, "tARates"
   DoCmd.DeleteObject acQuery, "b1"
   DoCmd.DeleteObject acQuery, "b2"
   DoCmd.DeleteObject acQuery, "tmpQry"
   On Error GoTo 0
   DoCmd.SetWarnings True
End Sub
Private Sub cmdRun_Click()
   Dim StartTime As Double, e As Double
   Dim hr As Byte, min As Byte, runs As Byte, i As Byte
   StartTime = Timer
   If IsNull(NumYrs) Or NumYrs < 1 Then
      MsgBox "Enter Number of Years", , "Hold On, There"
      Exit Sub
   End If
   runs = CByte(txtRuns)
   MakeRunDirs runs
For i = 1 To runs
    Driver i, NumYrs ', Array(E7Min, E8Min, E9Min, E7SMin, E8SMin, E9SMin)
Next

e = Timer - StartTime
hr = Int(e / 3600)
e = e - 3600 * hr
min = Int(e / 60)
e = e - 60 * min
MsgBox “Done “ & vbCrLfb & Format(hr, “00”) & “:“ & vbCrLf & Format(min, “00”) & “:“ & vbCrLf & Format(e, “00”) & “ Elapsed.”

End Sub

Private Sub cmdKillXL_DblClick(Cancel As Integer)
    KillXL CByte(txtRuns)
    MsgBox “Files Deleted”, , “Delete”
End Sub

Private Sub cmdXport_Click()
    XportMods
    MsgBox “Modules Exported”, , “Export”
End Sub
Private Sub cmdUse_Click()
    Dim varReturn As Variant
    If IsNull(frmChange.Value) Then GoTo NoChange
    AssignPers True
    DoCmd.OpenForm "Increase Decrease Manpower Targets", acNormal, "", "", , acNormal
    Exit Sub
NoChange:
    MsgBox "Choose a Type of Year-to-Year Change", , "Not So Fast, My Friend"
End Sub
Public Sub AssignPers (warn As Boolean)
    Dim i As Byte
    optPersChange = frmChange.Value
    If Not IsNull(txtNumPers) Then
        sngPers = CLng(txtNumPers)
        optPers = 3
    ElseIf Not IsNull(txtPctPers) Then
        sngPers = CSng(txtPctPers)
        optPers = 4
    Else
        If warn Then GoTo NoType
    End If
    Exit Sub
NoType:
    MsgBox "Enter an Amount to Increase/Decrease", , "Hang On"
End Sub
Private Sub cmdUse_Click()
    Dim varReturn As Variant
    If IsNull(framPctNum.Value) Then GoTo NoType
    optType = framPctNum.Value
    DoCmd.SelectObject acForm, "Controller", False
    Exit Sub
NoType:
    MsgBox "Choose a Type of Increase/Decrease", , "Hang On"
    varReturn = SysCmd(acSysCmdClearStatus)
End Sub

Public Sub AssignMan()
    Dim i As Byte, j As Byte
    optManChange = frmChange
    For j = 3 To 8
        sngIntDec(i, j - 3) = Controls.Item("txt" & i & j)
        min_tis(j - 1) = Controls.Item("txt1" & j + 1)
        min_tig(j - 1) = Controls.Item("txt2" & j + 1)
        sngLossChange(j) = Controls.Item("txt3" & j)
        bytStopYr(j) = Controls.Item("txt4" & j)
        If bytStopYr(j) = 0 Then bytStopYr(j) = 100
    Next
    sngLossChange(9) = Controls.Item("txt39")
    bytStopYr(9) = Controls.Item("txt49")
    If bytStopYr(9) = 0 Then bytStopYr(9) = 100
    min_tis(8) = 400
    min_tig(8) = 400
End Sub
Private Sub lblocc0_Click()
    CopyAcross 0
End Sub
Private Sub lblocc1_Click()
    CopyAcross 1
End Sub
Private Sub lblocc2_Click()
    CopyAcross 2
End Sub
Private Sub lblocc3_Click()
    CopyAcross 3
End Sub
Private Sub lblocc4_Click()
    CopyAcross 4
End Sub
Private Sub CopyAcross(rw As Byte)
    Dim i As Byte, txtval As String
    With Controls.Item("txt" & rw & "3")
        .SetFocus
        txtval = .Text
    End With
    For i = 4 To 9
        With Controls.Item("txt" & rw & i)
            .SetFocus
            .Text = txtval
        End With
    Next
    Controls.Item("txt" & rw & "3").SetFocus
End Sub
# Appendix C: PIAPM.xls programming code

## Table of contents

### Main
- **Sub** LoopDirs
- **Sub** CompileData
- **Private** Sub TISHeader
- **Private** Sub FixGetExpected
- **Private** Sub FixLikely
- **Public** Sub GetLikely
- **Private** Sub GetNewData
- **Private** Function Metadata
- **Sub** CopyAllSheets

### Compile
- **Sub** CompileAllData
- **Private** Sub MakeNewWB
- **Private** Sub CommonData
- **Private** Sub CopyData
- **Sub** MakePivot
- **Sub** CommonFields
- **Private** Sub NewField
- **Private** Sub FieldSets
- **Private** Sub LikFieldSets
- **Private** Sub CleanPivot
- **Private** Sub KillBadRow
- **Private** Sub FillCols
- **Private** Sub MoveData
- **Private** Sub KillLikSeries

### Formatting
- **Public** Sub FixLikChart
- **Public** Sub FixTISChart
- **Public** Sub FixAxis
- **Private** Sub SetSrc
- **Private** Sub Blue

### Robust
- **Sub** Robustness
- **Private** Sub ShortSepTIS
- **Private** Sub MakeCharts
Public Function MakeChart
Public Sub LikChart
Sub EMakeBoxPlot
Private Sub EBoxPlotFormat
Private Sub EOutliers
Private Sub ESeriesOrder
Private Sub KillYr0

Utilities
Public Function ELastCell
Public Sub RefreshPivot
Public Sub KillCmdBar
Sub XportMods
Function EMax
Public Sub KillCharts
Function GetRating
Private Function GetTableName
Function CopyModule
Sub AddProcedureToModule
Private Sub NewModLine
Sub AddReference
Sub ListReferencePaths

Choice form
Private Sub UserForm
Private Sub cmdChart
Private Sub spnYOS
Private Sub txtYOS
Private Function WhichData

Workbook
Private Sub Workbook
Private Sub NewButton
Private Sub Workbook

Sheet4
Private Sub cboLikPG
Private Sub cboLikYOS

Sheet6
Private Sub cboPG
Sub LoopDirs()
    Dim i As Byte
    Application.ScreenUpdating = False
    For i = 1 To 10
        CompileData i
    Next
    Application.ScreenUpdating = True
End Sub

Sub CompileData(numrun As Byte)
    Dim yrs As Byte, strPath As String, rate As String
    strPath = ActiveWorkbook.Path
    Set ws_t = ActiveWorkbook.Sheets("TIS Data")
    Set ws_l = ActiveWorkbook.Sheets("Likelihood Data")
    run = numrun

    'Delete old data
    ws_t.Activate
    Range(Cells(1, 1), Cells.SpecialCells(xlLastCell)).ClearContents
    TISHeader
    ws_l.Activate
    ActiveSheet.AutoFilterMode = False
    Range(Cells(1, 1), Cells.SpecialCells(xlLastCell)).ClearContents
    rating = GetRating
    For Each rule In Array(“J”, “M”, “S”)
        On Error Resume Next
        Workbooks.OpenText Filename:=ActiveWorkbook.Path & ”\" & run & “\Expected” & rating & rule & “.xls”
        If Err <> 1004 Then
            On Error GoTo 0
            yrs = ActiveWorkbook.Worksheets.Count
            For yr = 1 To yrs
                FixGetExpected
            Next
            Workbooks(“Expected” & rating & rule & “.xls”).Close False
        End If
    Next rule
    Workbooks.Close False
End Sub
Workbooks.OpenText Filename:=ActiveWorkbook.Path & \"\" & run & \"\" & \"Likelihood\" & rating & rule & \".xls\"
Set wb = ActiveWorkbook
FixLikely
GetLikely
Workbooks("Likelihood" & rating & rule & \".xls").Close False
End If
Next
On Error GoTo 0
GetNewData "Sep"
RefreshPivot "Sep"
GetNewData "Shortage"
RefreshPivot "Shortage"
GetNewData "Prom"
RefreshPivot "Prom"
GetNewData "TISTIG"
RefreshPivot "TISTIG"
GetNewData "YOS_PG"
RefreshPivot "YOS_PG"
ActiveSheet.PivotTables("PivotTable1").PivotFields("rule").CurrentPage = "M"
ActiveSheet.PivotTables("PivotTable1").PivotFields("yr").CurrentPage = "1"
rate = Metadata
Sheets("Metadata").Activate
CopyAllSheets
Set ws_l = ActiveWorkbook.Sheets("Likelihood Data")
FixLikChart yrs
FilterLikely
FixTISChart
FixAxis yrs
RefreshPivot "Sep"
RefreshPivot "Shortage"
Sheets("Metadata").Select
ActiveWorkbook.SaveAs strPath & \"\"Results\"\" & rate & \"_\" & Format(Now, "mddyy_hhmss") & \".xls\"
ActiveWorkbook.Close False
ws_t.Activate
FixLikChart yrs
FilterLikely
FixTISChart
FixAxis yrs
Sheets("Metadata").Select
Set ws_t = Nothing
Set ws_l = Nothing
Set wb = Nothing
End Sub

Private Sub TISHeader()
    With Cells(1, 1)
        .Value2 = "rating"
        .Offset(0, 1).Value2 = "rule"
        .Offset(0, 2).Value2 = "pg"
        .Offset(0, 3).Value2 = "yr"
        .Offset(0, 4).Value2 = "cnt"
        .Offset(0, 5).Value2 = "AvgOfprom_mnths"
    End With
End Sub

Private Sub FixGetExpected()
    Workbooks("Expected" & rating & rule & ".xls").Activate
    ActiveWorkbook.Sheets("EYr" & yr).Activate
    'Move data to driver
    Range("A3:F8").Copy
    ws_t.Activate
    'Move to first empty row
    Cells(ELastCell(ws_t).Row + 1, 1).Select
    ActiveSheet.Paste
    Application.CutCopyMode = False
End Sub

Private Sub FixLikely()
    wb.Activate
    'Move column headers to driver
    Rows("1:1").Select
    Selection.Cut
    ws_l.Activate
    Cells(1, 1).Select
    ActiveSheet.Paste
    wb.Activate
    Selection.Delete shift:=xlUp
End Sub

Private Sub GetLikely()
    'Move data to driver
    Selection.CurrentRegion.Select
    Selection.Copy
    ws_l.Activate
    Cells(ELastCell(ActiveSheet).Row + 1, 1).Select
ActiveSheet.Paste
Application.CutCopyMode = False
End Sub

Public Sub FilterLikely(Optional shift As Byte = 0)
Sheets("Likelihood Data").Activate 'ws_l.Activate
Selection.CurrentRegion.Select
'Sort by rating, rule, pg
Selection.Sort Key1:=Range(“B2”), Order1:=xlAscending, Key2:=Range(“C2”) _
, Order2:=xlAscending, Key3:=Range(“D2”), Order3:=xlAscending, Header:= _
 xlGuess, OrderCustom:=1, MatchCase:=False, Orientation:=xlTopToBottom
Selection.AutoFilter
' Selection.AutoFilter Field:=3, Criterial:=Cells(2, 3).Value2
Selection.AutoFilter Field:=4, Criterial:=Cells(2, 4).Value2
Cells(1, 1).Select
End Sub

Private Sub GetNewData(dat As String)
Dim s_address As String
Set ws_l = ActiveWorkbook.Sheets(dat & “ Data”)
ws_l.Activate
Range(Cells(1, 1), Cells.SpecialCells(xlLastCell)).ClearContents
Workbooks.OpenText Filename:=ActiveWorkbook.Path _
 & “\” & run & “\” & dat & “.xls”
Cells(1, 1).CurrentRegion.Select
Selection.Copy
ws_l.Activate
Cells(1, 1).Select
ActiveSheet.Paste
Application.CutCopyMode = False
Workbooks(dat & “.xls”).Close False
Sheets(dat & “ Chart”).Select

NoPivot:
Set ws_l = Nothing
End Sub

Private Function Metadata() As String
Dim s_address As String
Set ws_l = ActiveWorkbook.Sheets(“Metadata”)
ws_l.Activate
Range(Cells(1, 1), Cells.SpecialCells(xlLastCell)).ClearContents
Workbooks.OpenText Filename:=ActiveWorkbook.Path _
 & “\” & run & “\” & “Metadata.xls”
Range(Cells(1, 1), Cells(1, 2)).ClearContents
Cells(2, 1).CurrentRegion.Select
Dim c As Range
For Each c In Selection
    c.Value2 = c.Value2
Next
Selection.Copy
ws_1.Activate
Cells(1, 1).Select
ActiveSheet.Paste
Application.CutCopyMode = False
Workbooks("Metadata.xls").Close False
Range("B1").NumberFormat = "m/d/yy h:mm AM/PM"
Columns("A:B").EntireColumn.AutoFit
Metadata = Cells(3, 2).Value2
Cells(1, 1).Select
End Function
Sub CopyAllSheets()
    Sheets(Array("YOS_PG Chart", "YOS_PG Pivot", "YOS_PG Data", _
             "TISTIG Chart", "TISTIG Pivot", "TISTIG Data", _
             "Prom Chart", "Prom Pivot", "Prom Data", _
             "Shortage Chart", "Shortage Pivot", "Shortage Data", _
             "Sep Chart", "Sep Pivot", "Sep Data", _
             "TIS Chart", "TIS", "TIS Pivot", "TIS Data", _
             "Likelihood Chart", "Likelihood Data", "Metadata")).Copy
End Sub
Compile

Attribute VB_Name = "Compile"
'Programming by Robert W. Shuford, CNA
Option Explicit
Option Base 1
Const DATASHEET = 7
Dim wb As Workbook, wbNew As Workbook, wbDriver As Workbook
Dim bytYrs As Byte
Public blnDisableEvents As Boolean
Sub CompileAllData()
    Dim intFiles As Integer, strPath As String, bytLikCol As Byte, i As Integer
    Dim rate As String
    Application.ScreenUpdating = False
    Set wbDriver = ActiveWorkbook
    bytLikCol = 0
    strPath = ActiveWorkbook.Path & "\"
    Worksheets("Likelihood Data").Select
    Set wbNew = Workbooks.Add
    With Application.FileSearch
        .NewSearch
        .LookIn = strPath
        .Filename = "Results_*.xls"
        ' Loop through files
        If .Execute() > 0 Then
            For i = 1 To .FoundFiles.Count
                Workbooks.Open Filename:=.FoundFiles(i)
                Set wb = ActiveWorkbook
                Sheets("Metadata").Activate
                If i = 1 Then rate = Cells(3, 2).Value2
                If bytLikCol = 0 Then
                    bytLikCol = Cells(2, 2) + 6
                    MakeNewWB bytLikCol
                ElseIf bytLikCol <> Cells(2, 2) + 6 Then
                    MsgBox wb.Name & " does not contain the same " & "number of years as the previous file(s)."
                    & vbCrLf & vbCrLf & "All spreadsheets in " & strPath & " beginning with Results_ must " & "have the same number of years."
                    & "Houston, we have a problem!"
                ElseIf bytLikCol <> Cells(2, 2) + 6 Then
                    MsgBox wb.Name & " does not contain the same " & "number of years as the previous file(s)."
                    & vbCrLf & vbCrLf & "All spreadsheets in " & strPath & " beginning with Results_ must " & "have the same number of years."
                    & "Houston, we have a problem!"
    wb.Close False
GoTo BadYrs
End If

CopyData "Shortage"
CopyData "Sep"
CopyData "TIS"
CopyData "Likelihood"
CopyData "Prom"
CopyData "TISTIG"
CopyData "YOS_PG"
Application.CutCopyMode = False
wb.Close False

Next
Else
MsgBox "No files found in current directory"
GoTo BadYrs
End If
End With

blnDisableEvents = True
MakePivot "Shortage"
MakePivot "Sep"
MakePivot "TIS"
MakePivot "Likelihood"
MakePivot "Prom"
MakePivot "TISTIG"
MakePivot "YOS_PG"
wDriver.Activate
Application.DisplayAlerts = False

CopyAllSheets
Sheets("Metadata").Delete
FixLikChart bytYrs
FilterLikely
FixTISChart
FixAxis bytYrs
RefreshPivot "Sep"
RefreshPivot "Shortage"
RefreshPivot "Prom"
RefreshPivot "TISTIG"
RefreshPivot "YOS_PG"
Application.DisplayAlerts = True
ActiveWorkbook.SaveAs strPath & "Compiled_" & rate & "_"
& Application.FileSearch.FoundFiles.Count & "_Files_ "
& bytYrs & "_Yrs" & ".xls"

BadYrs:
  blnDisableEvents = False
  wbNew.Close False
  Set wb = Nothing
  Set wbNew = Nothing
  Set wbDriver = Nothing
  Application.ScreenUpdating = True
End Sub

Private Sub MakeNewWB(bytLikCols As Byte)
  Dim i As Integer
  wbNew.Activate
  Application.DisplayAlerts = False
  For i = Worksheets.Count To 2 Step -1
    Worksheets(i).Delete
  Next
  For i = 2 To DATASHEETS
    Sheets.Add After:=Worksheets(Worksheets.Count)
  Next
  i = 1
  Sheets(i).Select
  ActiveSheet.Name = “YOS_PG Data”
  CommonData
  Cells(i, 6) = “yos”
  i = i + 1

  Sheets(i).Select
  ActiveSheet.Name = “TISTIG Data”
  CommonData
  Cells(i, 6) = “tis”
  Cells(i, 7) = “tig”
  i = i + 1

  Sheets(i).Select
  ActiveSheet.Name = “Prom Data”
  CommonData
  Cells(i, 6) = “proms”
  Cells(i, 7) = “pct”
  i = i + 1
Sheets(i).Select
ActiveSheet.Name = "Shortage Data"

CommonData
Cells(1, 6) = "target"
Cells(1, 7) = "shortage"
i = i + 1

Sheets(i).Select
ActiveSheet.Name = "Sep Data"

CommonData
Cells(1, 6) = "seps"
Cells(1, 7) = "pct"
i = i + 1

Sheets(i).Select
ActiveSheet.Name = "TIS Data"

CommonData
Cells(1, 6) = "AvgOfprom_mnths"
i = i + 1

Sheets(i).Select
ActiveSheet.Name = "Likelihood Data"

CommonData
i = i + 1

For i = 6 To bytLikCols - 1
    Cells(1, i) = "Yr " & i - 5
Next
bytYrs = bytLikCols - 6
Cells(1, bytLikCols) = "likelihood"

End Sub
Private Sub CommonData()
    Cells(1, 1) = "rating"
    Cells(1, 2) = "rule"
    Cells(1, 3) = "pg"
    Cells(1, 4) = "yr"
    Cells(1, 5) = "cnt"
End Sub
Private Sub CopyData(str As String)
    wbNew.Activate
    Worksheets(str & " Data").Select
    Cells(ELastCell(ActiveSheet).Row + 1, 1).Select
wb.Activate
Worksheets(str & " Data").Select
Selection.AutoFilter
Range(Cells(2, 1), ELastCell(ActiveSheet)).Copy
wbNew.Activate
ActiveSheet.Paste
End Sub

Sub MakePivot(str As String)
Dim i As Byte, bytBadCol As Byte
Worksheets(str & " Data").Select
Cells(1, 1).Select
Selection.CurrentRegion.Select
ActiveWorkbook.PivotCaches.Add(SourceType:=xlDatabase, SourceData:= _
Selection.Address).CreatePivotTable TableDestination:="", _
TableName:="PivotTable1"
ActiveSheet.PivotTableWizard TableDestination:=ActiveSheet.Cells(3, 1)
ActiveSheet.Cells(3, 1).Select
ActiveSheet.PivotTables("PivotTable1").SmallGrid = False
ActiveSheet.Name = str & " Pivot"
bytBadCol = 3
Select Case str
Case “YOS_PG”
CommonFields
With ActiveSheet.PivotTables("PivotTable1").PivotFields("yos")
.Orientation = xlRowField
End With
FieldSets 5, “cnt”, “pg”
bytBadCol = 3
Case “TISTIG”
CommonFields
FieldSets 4, “tis”, “tig”
Case “Prom”
CommonFields
FieldSets 4, “cnt”, “proms”, “pct”
Case “Shortage”
CommonFields
FieldSets 4, “cnt”, “target”, “shortage”
Case “Sep”
Case "TIS"
   CommonFields
   FieldSets 4, "cnt", "seps", "pct"
End Select
Case "Likelihood"
   CommonFields
   LikFieldSets bytYrs
End Select
With ActiveSheet.PivotTables("PivotTable1").PivotFields("Data")
   .Orientation = xlColumnField
   .Position = 1
End With
CleanPivot str, bytBadCol
End Sub
Sub CommonFields()
   Dim i As Byte
   i = 1
   With ActiveSheet.PivotTables("PivotTable1")
      With .PivotFields("rule")
         .Orientation = xlRowField
         .Position = i
         i = i + 1
      End With
      With .PivotFields("pg")
         .Orientation = xlRowField
         .Position = i
         i = i + 1
      End With
      With .PivotFields("yr")
         .Orientation = xlRowField
         .Position = i
         i = i + 1
      End With
   End With
End Sub
Private Sub NewField(fld As String, pos As Byte)
   With ActiveSheet.PivotTables("PivotTable1")
      With .PivotFields(fld)
      End With
   End With
End Sub
Private Sub FieldSets(bytDatCol As Byte, ParamArray fields() As Variant)
Dim i As Byte, bytStrt As Byte, bytFldNum As Byte, xl As Variant
bytStrt = 1
bytFldNum = 1
For Each xl In Array(xlAverage, xlStDev, xlMin, xlMax)
    For i = LBound(fields) To UBound(fields)
        NewField CStr(fields(i)), bytFldNum
        bytFldNum = bytFldNum + 1
    Next
    For i = bytStrt To bytFldNum - 1
        With ActiveSheet.PivotTables("PivotTable1")
            .PivotFields(Cells(i + 3, bytDatCol).Value2).Function = xl
        End With
    Next
    bytStrt = bytFldNum
Next
End Sub

Private Sub LikFieldSets(yrs As Byte)
Dim i As Byte, bytStrt As Byte, bytFldNum As Byte, xl As Variant
NewField "cnt", 1
bytStrt = 1
bytFldNum = 2
For Each xl In Array(xlAverage, xlStDev, xlMin, xlMax)
    For i = 1 To yrs
        NewField "Yr " & i, bytFldNum
        bytFldNum = bytFldNum + 1
    Next
    For i = bytStrt To bytFldNum - 1
        With ActiveSheet.PivotTables("PivotTable1")
            .PivotFields(Cells(i + 3, 4).Value2).Function = xl
        End With
    Next
    bytStrt = bytFldNum
Next
End Sub

Private Sub CleanPivot(str As String, col As Byte)
Selection.CurrentRegion.Select
Sheets.Add
ActiveSheet.Name = str
Sheets(str & " Pivot").Select
Selection.Copy
Sheets(str).Select
Selection.PasteSpecial Paste:=xlValues, Operation:=xlNone, SkipBlanks:= _
False, Transpose:=False
Rows("1:1").Select
Application.CutCopyMode = False
Selection.Delete shift:=xlUp

KillBadRow col
FillCols col - 1
SearchOrder:=xlByRows, MatchCase:=False
ActiveCell.CurrentRegion.Columns.AutoFit
Cells(1, 1).Select
If str = "Likelihood" Then Cells(1, 3).Value2 = "yos"
If str = "YOS_PG" Then
  Range("F:F,H:H,J:J,L:L").Select
  Selection.Delete shift:=xlToLeft
  Range("A1").Select
End If
End If
MoveData str

Private Sub KillBadRow(c As Byte)
Dim R As Integer
For R = ELastCell(ActiveSheet).Row To 1 Step -1
  If Cells(R, c).Value2 = "" Then
    Rows(R).EntireRow.Select
    Selection.Delete shift:=xlUp
  End If
Next
End Sub

Private Sub FillCols(col As Byte)
Dim R As Integer, c As Byte, intLastRow
intLastRow = ELastCell(ActiveSheet).Row
For c = 1 To col
  For R = 1 To intLastRow
    Cells(R, c).Activate
    If ActiveCell.Value2 = "" Then ActiveSheet.Paste
    ActiveCell.Copy
  Next
Private Sub MoveData(str As String)
    wbDriver.Sheets(str & " Data").Activate
    ActiveSheet.AutoFilterMode = False
    Range(Cells(1, 1), Cells.SpecialCells(xlLastCell)).ClearContents
    Cells(1, 1).Select
    ActiveCell.Value2 = "rating"
    wbNew.Activate
    ActiveCell.CurrentRegion.Copy
    wbDriver.Activate
    If str = "Likelihood" Then Cells(1, 2).Select
    ActiveSheet.Paste
    ActiveCell.CurrentRegion.Columns.AutoFit

    Select Case str
        Case "YOS_PG", "TISTIG", "Prom", "Shortage", "Sep"
            RefreshPivot str
        Case "TIS"
            RefreshPivot str
            FixTISChart
            FixAxis bytYrs
        Case "Likelihood"
            Sheets(str & " Data").Select
            With Selection
                .AutoFilter
                .AutoFilter Field:=4, Criteria1:=Cells(2, 4).Value2
            End With
            FixLikChart bytYrs
            KillLikSeries
    End Select

    wbNew.Activate
End Sub

Private Sub KillLikSeries()
    ActiveSheet.ChartObjects("Chart 1").Activate
    ActiveChart.ChartArea.Select
    On Error GoTo Done
    Do While GoTo Done
ActiveChart.SeriesCollection(bytYrs + 1).Delete
Loop
Done:
End Sub
Public Sub FixLikChart(yrs As Byte)
    Dim l_cols As Byte, l_rows As Long, l_address As String, i As Byte
    Sheets("Likelihood Data").Activate
    'Determine number of years
    ActiveSheet.AutoFilterMode = False
    l_cols = yrs + 5
    Range(Cells(1, 6), Cells(1, l_cols)).Select
    Range(Selection, Selection.End(xlDown)).Select
    l_address = Selection.Address
    l_rows = Selection.Rows.Count
    ActiveWorkbook.Sheets("Likelihood Chart").Activate
    'Set source data for chart
    ActiveSheet.ChartObjects("Chart 1").Activate
    With ActiveChart
        For i = 1 To l_cols - 4
            .SetSourceData Source:=Sheets("Likelihood Data").Range(l_address)
        Next
    End With
    ActiveChart.SeriesCollection(1).XValues = 
    & "=’Likelihood Data’!R2C2:R" & l_rows & "C3"
    ActiveSheet.Cells(1, 3).Select
    ActiveSheet.Shapes("cboLikYOS").Select
    Selection.ListFillRange = "B1:B31"
    ActiveSheet.OLEObjects("cboLikYOS").Object.Value = Cells(1, 2)
End Sub

Public Sub FixTISChart()
    'Refresh pivot table
    Sheets("TIS Pivot").Select
    RefreshPivot "TIS"
    'Set source data for controls
Public Sub FixAxis(numyrs As Byte)
    Dim i As Byte, o As Byte, ws_tp As Worksheet
    Set ws_tp = ActiveWorkbook.Sheets("TIS Pivot")
    ActiveWorkbook.Sheets("TIS").Activate
    Range(Cells(1, 1), Cells.SpecialCells(xlLastCell)).ClearContents
    Cells(1, 1).Select
    ActiveCell.Value2 = "PG"
    ActiveCell.Offset(0, 1).Select
    For i = 1 To numyrs
        ActiveCell.Value2 = "Junior" & i
        ActiveCell.Offset(0, 1).Select
        ActiveCell.Value2 = "Bench" & i
        ActiveCell.Offset(0, 1).Select
        ActiveCell.Value2 = "Senior" & i
        ActiveCell.Offset(0, 1).Select
    Next
    ActiveCell.Value2 = "Year"
    ActiveCell.Offset(1, 0).Select
    For o = 4 To 9
        For i = 1 To numyrs
            ActiveCell.Value2 = "y" & i
            ActiveCell.Offset(1, 0).Select
        Next
        ActiveCell.Offset(1, 0).Select
        Next
    Cells(2, 1).Value2 = "E4"
    For o = 5 To 9
        Cells((o - 4) * (numyrs + 1) + 2, 1).Value2 = "E" & o
    Next
    For o = 4 To 9
        Cells((o - 4) * (numyrs + 1) + 2, 2).Select
        For i = 1 To numyrs
            ActiveCell.FormulaR1C1 = "='TIS Pivot'!R" & o + 2 & "C" & 4 * i - 2
            ActiveCell.Offset(0, 1).FormulaR1C1 = "='TIS Pivot'!R" & o + 2 & "C" & 4 * i - 1 & "-'TIS Pivot'!R" & o + 2 & "C" & 4 * i - 2
            ActiveCell.Offset(0, 2).FormulaR1C1 = =IF("_ & ActiveCell.Offset(0, 1).Address(ReferenceStyle:=xlR1C1) & ">=0,'TIS Pivot'!R" & o + 2 & "C" & 4 * i - 1 & "-'TIS Pivot'!R" & o + 2 & "C" & 4 * i - 2 
    Next
End Sub
& "-'TIS Pivot'!R" & 0 + 2 & "C" & 4 * i - 2 & "")"
ActiveCell.Offset(1, 3).Select
Next
ActiveCell.Offset(-1, -1).Select
SetSrc numyrs
End Sub
Private Sub SetSrc(yrs As Byte)
Dim rng As Range, i As Byte
ActiveWorkbook.Sheets("TIS").Activate
Set rng = Range(ActiveCell, Cells(1, 1))
ActiveWorkbook.Sheets("TIS Chart").Activate
ActiveSheet.ChartObjects("Chart 1").Activate
ActiveChart.ChartArea.Select
ActiveChart.SetSourceData Source:=Sheets("TIS").Range(rng.Address)
For i = 1 To ActiveChart.SeriesCollection.Count
ActiveChart.SeriesCollection(i).XValues =
ActiveChart.SeriesCollection(i).Select
    Blue 37
Next
For i = 1 To yrs
    ActiveChart.SeriesCollection(i * 3 - 1).Select
    Blue 41
Next
For i = 1 To yrs
    ActiveChart.SeriesCollection(i * 3).Select
    Blue 25
Next
ActiveChart.Deselect
Set rng = Nothing
End Sub
Private Sub Blue(clr As Byte)
With Selection.Border
    .ColorIndex = 2
    .Weight = xlThin
    .LineStyle = xlContinuous
End With
Selection.Shadow = False
Selection.InvertIfNegative = False
With Selection.Interior
    .ColorIndex = clr
End With
.Pattern = xlSolid
End With
End Sub
Robust

Attribute VB_Name = "Robust"
'Programming by Robert W. Shuford, CNA
Option Explicit
Option Base 1
Const A = 1
Const B = 2
Const c = 3
Const D = 4
Const E = 5
Const F = 6
Const G = 7
Const H = 8
Const i = 9
Const j = 10
Const k = 11
Const L = 12
Const M = 13
Const N = 14
Const o = 15
Const p = 16
Const Q = 17
Const R = 18
Const S = 19
Const T = 20
Sub Robustness()
    Dim strFN As String, strPath As String
    Application.ScreenUpdating = False
    strPath = ActiveWorkbook.Path
    strFN = ActiveWorkbook.Name
    Sheets(Array("TISTIG Data", "Prom Data", "Shortage Data", "Sep Data", "TIS Data", "Likelihood Data")).Copy
    Sheets("Sep Data").Copy Before:=Sheets(1)
    ActiveSheet.Name = "Count Data"
    Sheets("TISTIG Data").Copy Before:=Sheets(2)
    ActiveSheet.Name = "AveTIS Data"
    Sheets("TISTIG Data").Name = "AveTIG Data"
    Sheets("TISTIG Data").Name = "AveTIG Data"
    ShortSepTIS "Count", D, j, D, G, M, D 'E, K, E, H, N, E
    '1st data column, min As Byte, base stat, std, max, id
    ShortSepTIS "AveTIS", D, H, D, F, j, D
    #11Yr0
```
Private Sub ShortSepTIS(dat As String, col As Byte, min As Byte, _
    base As Byte, std As Byte, max As Byte, id As Byte)
    Dim stat As Variant, bytcols As Byte
    Sheets(dat & " Data").Select
    Sheets.Add
    ActiveSheet.Name = dat & " Chart"
    bytcols = ELastCell(ActiveSheet).Column
    For Each stat In Array(min, base, std, max)
        Columns(stat).Select
        Selection.Copy
        Columns(stat + bytcols).Select
        ActiveSheet.Paste
        Next
    Min
    Columns(min + bytcols).Select
    Selection.Copy
    Columns(col).Select
    ActiveSheet.Paste
    ‘base stat
    Columns(base + bytcols).Select
    Selection.Copy
    Columns(col + 2).Select
    ActiveSheet.Paste
```
'Max
Columns(max + bytcols).Select
Selection.Copy
Columns(col + 4).Select
ActiveSheet.Paste
'
-2
Cells(2, col + 1).Select
With ActiveCell
    .FormulaR1C1 = "=RC" & base + bytcols & "-1*RC" & std + bytcols
    .Copy
    Range(ActiveCell, Cells(ELastCell(ActiveSheet).Row, .Column)).Select
    ActiveSheet.Paste
    Selection.Copy
    Selection.PasteSpecial Paste:=xlValues, Operation:=xlNone, _
        SkipBlanks:=False, Transpose:=False
End With
'
+2
Cells(2, col + 3).Select
With ActiveCell
    .Formula = "=RC" & base + bytcols & "+RC" & std + bytcols
    .Copy
    Range(ActiveCell, Cells(ELastCell(ActiveSheet).Row, .Column)).Select
    ActiveSheet.Paste
    Selection.Copy
    Selection.PasteSpecial Paste:=xlValues, Operation:=xlNone, _
        SkipBlanks:=False, Transpose:=False
End With
Cells(1, col).Value2 = "Min"
Cells(1, col + 1).Value2 = "-1 STD"
Cells(1, col + 3).Value2 = "+1 STD"
Cells(1, col + 4).Value2 = "Max"

Range(Cells(1, col + 5), ELastCell(ActiveSheet)).ClearContents
'InsertID id
Range(“A1”).Select
End Sub
Private Sub MakeCharts()
    MakeChart "Count"
    MakeChart "AveTIS"
    MakeChart "AveTIG"
    'MakeChart "YOS_PG"
    MakeChart "Prom"
Public Function MakeChart(dat As String, Optional pg As Byte = 4)
    Dim yrs As Byte, rng As Range
    Sheets(dat & " Chart").Select
    Range.Cells(1, 1), ELastCell(ActiveSheet)).ClearContents
    Sheets(dat & " Data").Select
    Cells(2, 3).Select
    yrs = EMax
    Selection.CurrentRegion.Select
    Selection.AutoFilter Field:=1, Criteria1:=rating
    Selection.AutoFilter Field:=2, Criteria1:=pg
    Cells(1, 3).Select
    Range.Select, ELastCell(ActiveSheet)).Select
    SelectionCopy
    Sheets(dat & " Chart").Select
    Cells(1, 2).Select
    ActiveSheet.Paste
    Cells(1, 1).Value2 = "rule"
    Cells(2, 1).Value2 = "J"
    Cells(yrs + 2, 1).Value2 = "M"
    Cells(2 * yrs + 2, 1).Value2 = "S"
    Cells(2, 2).Select
    Range.Select, Selection.End(xlDown)).Select
    For Each rng In Selection
        rng.Value2 = "" & rng.Value2
    Next
    ActiveCell.CurrentRegion.Select
End Function

Public Sub LikChart(Optional pg As Byte = 3, _
    Optional yrs As Byte = 0)
    Dim i As Byte, j As Byte, k As Byte, yrs As Byte, rng As Range, stat() As Single
    Sheets("TIS Data").Select
    Cells(2, 3).Select
Range(Selection, Selection.End(xlDown)).Select
yrs = EMax
ReDim stat(3, 4, yrs)
Sheets("Likelihood Chart").Select
Range(Cells(1, 1), ELastCell(ActiveSheet)).ClearContents
Sheets("Likelihood Data").Select
Cells(1, 1).CurrentRegion.Select
 Selection.AutoFilter Field:=1, Criteria1:=rating
 Selection.AutoFilter Field:=3, Criteria1:=pg
 Selection.AutoFilter Field:=4, Criteria1:=yos
Cells(1, 6).Select
Range(Selection, ELastCell(ActiveSheet)).Select
 Sheets.Add
 ActiveSheet.Name = "Likelihood Chart"
 Sheets("Likelihood Data").Select
Selection.Copy
Sheets("Likelihood Chart").Select
Cells(1, 1).Select
ActiveSheet.Paste
Cells(2, 1).Select
For i = 1 To 4
 For j = 1 To yrs
  stat(1, i, j) = ActiveCell.Value2
  stat(2, i, j) = ActiveCell.Offset(1, 0).Value2
  stat(3, i, j) = ActiveCell.Offset(2, 0).Value2
  ActiveCell.Offset(0, 1).Select
 Next
Next
Rows("1:4").Select
Selection.ClearContents 'Delete Shift:=xlUp
Cells(1, 1).Value2 = "rule"
Cells(1, 2).Value2 = "year"
Cells(1, 3).Value2 = "'Min"
Cells(1, 4).Value2 = "'-1 STD"
Cells(1, 5).Value2 = "Pct"
Cells(1, 6).Value2 = "'+1 STD"
Cells(1, 7).Value2 = "Max"
Cells(2, 1).Value2 = "J"
Cells(yrs + 2, 1).Value2 = "M"
Cells(2 * yrs + 2, 1).Value2 = "S"
Cells(2, 2).Select
For k = 1 To 3
For \( i = 1 \) To yrs
  With ActiveCell
    .Value2 = "" & i
    .Offset(0, 1).Value2 = stat(k, 3, i)  'Min
    .Offset(0, 2).Value2 = stat(k, 1, i) - stat(k, 2, i)  '-1 STD
    .Offset(0, 3).Value2 = stat(k, 1, i)  'base
    .Offset(0, 4).Value2 = stat(k, 1, i) + stat(k, 2, i)  '+1 STD
    .Offset(0, 5).Value2 = stat(k, 4, i)  'Max
  .Offset(1, 0).Select
  Next
Cells(yrs + 2, 2).Select
Next
Cells(1, 1).CurrentRegion.Select
EMakeBoxPlot pg, yos
End Sub
Sub EMakeBoxPlot (pg As Byte, Optional yos As Byte = 99)
  Dim ws As String, rngR As String, title As String
  On Error Resume Next 'GoTo BoxErr
  title = "PG " & pg
  If yos <> 99 Then title = title & " YOS " & yos
  ws = ActiveSheet.Name
  rngR = Selection.Address
  Charts.Add
  With ActiveChart
    .HasTitle = True
    .ChartTitle.Characters.Text = title
    .ChartType = xlLineMarkers
    .SetSourceData Source:=Sheets(ws).Range(rngR)
    .Location Where:=xlLocationAsObject, Name:=ws
  End With
  With ActiveChart
    .SeriesCollection(1).Select
    With .ChartGroups(1)
      .HasDropLines = False
      .HasHiLoLines = True
      .HasUpDownBars = True
      .GapWidth = 150
    End With
  .ChartGroups(1).UpBars.Select
End Sub
With Selection.Border
    .Weight = xlMedium
    .LineStyle = xlContinuous
End With
With Selection.Interior
    .ColorIndex = 15
    .PatternColorIndex = 1
    .Pattern = xlSolid
End With

.Legend.Select
Selection.Delete
EBoxPlotFormat
End With
Exit Sub
On Error GoTo 0
'BoxErr:
End Sub
Private Sub EBoxPlotFormat()
    Dim sc As Integer, lb As Integer, ub As Integer, i As Integer
    With ActiveChart
        sc = .SeriesCollection.Count
        lb = Int(sc / 2)
        ub = Int((sc / 2)) + 1 + (sc Mod 2)
        'Series outside of box
        For i = 1 To lb - 1
            EOutliers i
        Next
        For i = ub + 1 To sc
            EOutliers i
        Next
        'Median
        If ub - lb = 2 Then EOutliers lb + 1, 3, xlDash, 10
        'Box
        ESeriesOrder lb, 1
        ESeriesOrder ub, sc
    End With
    Range(“A1”).Activate
End Sub
Private Sub EOutliers(series As Integer, Optional color As Byte = 1, _
    Optional style As Integer = xlCircle, Optional size As Byte = 5)
    ActiveChart.SeriesCollection(series).Select
Selection.Border.LineStyle = xlNone
With Selection
  .MarkerBackgroundColor = color
  .MarkerForegroundColor = color
  .MarkerStyle = style
  .MarkerSize = size
End With
End Sub

Private Sub ESeriesOrder(series As Integer, order As Integer)
  ActiveChart.ChartGroups(1).SeriesCollection(series).Select
  Selection.Border.LineStyle = xlNone
  With Selection
    Selection.MarkerStyle = xlNone
    .PlotOrder = order
  End With
End Sub

Private Sub KillYr0()
  Dim i As Long
  For i = 2 To ELastCell(ActiveSheet).Row
    If Cells(i, 3).Value2 = 0 Then
      Rows(i).EntireRow.Select
      Selection.Delete shift:=xlUp
    End If
  Next
End Sub
Utilities
Attribute VB_Name = "Utilities"
'Programming by Robert W. Shuford, CNA
Option Explicit
Option Private Module
Public Function LastCell(TheSheet As Worksheet) As Range
   ' Returns a single-cell range object that represents
   ' the intersection of the last non-empty row and the
   ' last non-empty column
   Dim ExcelLastCell As Range
   Dim Row As Long, col As Integer
   Dim LastRowWithData As Long, LastColWithData As Integer

   ' ExcelLastCell is what Excel thinks is the last cell
   Set ExcelLastCell = TheSheet.Cells.SpecialCells(xlLastCell)

   ' Determine the last row with data in it
   LastRowWithData = ExcelLastCell.Row
   Row = ExcelLastCell.Row
   Do While Application.CountA(TheSheet.Rows(Row)) = 0 And Row <> 1
       Row = Row - 1
   Loop
   LastRowWithData = Row

   ' Determine the last column with data in it
   LastColWithData = ExcelLastCell.Column
   col = ExcelLastCell.Column
   Do While Application.CountA(TheSheet.Columns(col)) = 0 And col <> 1
       col = col - 1
   Loop
   LastColWithData = col

   ' Create the range object
   Set LastCell = TheSheet.Cells(LastRowWithData, LastColWithData)
End Function

Public Sub RefreshPivot(str As String)
   Dim s_address As String
   Sheets(str & " Data").Select
   Cells(1, 1).CurrentRegion.Select
   s_address = Selection.Address(ReferenceStyle:=xlR1C1)
Sheets(str & " Pivot").Select
Cells(4, 1).Activate
ActiveSheet.PivotTableWizard SourceType:=xlDatabase, SourceData:= _
"" & str & " Data'!" & _s_address
If str = "YOS_PG" Then
    ActiveSheet.PivotTables("PivotTable1").PivotFields("rule").CurrentPage = "M"
    ActiveSheet.PivotTables("PivotTable1").PivotFields("yr").CurrentPage = "1"
End If
ActiveSheet.PivotTables(1).RefreshTable
End Sub
Public Sub KillCmdBar()
    On Error Resume Next
    Application.CommandBars("New Data").Delete
    On Error GoTo 0
End Sub
Sub XportMods()
    Dim mdl As Variant, strFile As String, strExt As String
    For Each mdl In Application.VBE.ActiveVBProject.VBComponents()
        strFile = ".bas"
        If Left(mdl.Name, 5) = "Form_" Then strFile = ".cls"
        strFile = Mid(ActiveWorkbook.Name, Len(ActiveWorkbook.Name) - 7, 4) & strFile
        If Left(mdl.Name, 5) <> "Chart" And (Left(mdl.Name, 5) <> "Sheet"
            Or Left(mdl.Name, 6) = "Sheet4"
            Or Left(mdl.Name, 6) = "Sheet6") Then _
            mdl.Export ActiveWorkbook.Path & "\Modules\Excel" & mdl.Name & strFile
        strFile = ".bas"
        If Left(mdl.Name, 5) = "Form_" Then strFile = ".cls"
        If Left(mdl.Name, 5) <> "Chart" And (Left(mdl.Name, 5) <> "Sheet"
            Or Left(mdl.Name, 6) = "Sheet4"
            Or Left(mdl.Name, 6) = "Sheet6") Then _
            mdl.Export ActiveWorkbook.Path & "\Modules\Excel" & mdl.Name & strFile
        Next
    Set mdl = Nothing
End Sub
Function EMax() As Double
    Dim rngRange As Range, rngMax As Range, c As Range
    Dim dblMaxVal As Double
    Set rngRange = Selection
    dblMaxVal = -1.79769313486231E+308
    Set rngMax = ActiveCell
For Each c In rngRange
    If Not IsEmpty(c.Value) And Not IsError(c.Value) And (IsNumeric(c.Value) Or IsDate(c.Value)) Then
        If c.Value > dblMaxVal Then
            dblMaxVal = c.Value
            Set rngMax = c
        End If
    End If
Next
EMax = dblMaxVal
Set rngRange = Nothing
Set rngMax = Nothing
Set c = Nothing
End Function

Public Sub KillCharts()
    Dim i As Integer
    For i = ActiveSheet.ChartObjects.Count To 1 Step -1
        ActiveSheet.ChartObjects(i).Delete
    Next
End Sub

Function GetRating() As String
    'Extracts the rating from the data files in the current directory
    With Application.FileSearch
        .NewSearch
        .LookIn = ActiveWorkbook.Path & "\1\"
        .Filename = "Likelihood*.xls"
        If .Execute > 0 Then
            GetRating = Mid(GetTableName(.FoundFiles(1)), 11)
            GetRating = Left(GetRating, Len(GetRating) - 5)
        End If
    End With
End Function

Private Function GetTableName(OldPath As String) As String
    Dim bytSlash As Byte
    Do
        bytSlash = InStr(OldPath, "\")
        OldPath = Mid(OldPath, bytSlash + 1)
    Loop Until bytSlash = 0
    GetTableName = OldPath
End Function

Function CopyModule(ModuleName As String, _
    FromVBProject As VBIDE.VBProject, _
    ToVBProject As VBIDE.VBProject, _
ToVBProject As VBIDE.VBProject, _
OverwriteExisting As Boolean) As Boolean

' CopyModule
' This function copies a module from one VBProject to
' another. It returns True if successful or False
' if an error occurs.
'
' Parameters:
' --------------------------------
' FromVBProject         The VBProject that contains the module
to be copied.
' '
' ToVBProject           The VBProject into which the module is
to be copied.
' '
' ModuleName            The name of the module to copy.
' '
' OverwriteExisting     If True, the VBComponent named ModuleName
in ToVBProject will be removed before
importing the module. If False and
a VBComponent named ModuleName exists
in ToVBProject, the code will return
False.
'
Dim VBComp As VBIDE.VBComponent
Dim FName As String

' Do some housekeeping validation.
' 'DimSupportActionBar As String
If FromVBProject Is Nothing Then
    CopyModule = False
    Exit Function
End If

If Trim(ModuleName) = vbNullString Then
    CopyModule = False
    Exit Function
End If

End Function
If ToVBProject Is Nothing Then
    CopyModule = False
    Exit Function
End If

If FromVBProject.Protection = vbext_pp_locked Then
    CopyModule = False
    Exit Function
End If

If ToVBProject.Protection = vbext_pp_locked Then
    CopyModule = False
    Exit Function
End If

On Error Resume Next
Set VBComp = FromVBProject.VBComponents(ModuleName)
If Err.Number <> 0 Then
    CopyModule = False
    Exit Function
End If

''''''''''''''''''''''''''''''''''''''''''''''''''''
' FName is the name of the temporary file to be
' used in the Export/Import code.
''''''''''''''''''''''''''''''''''''''''''''''''''''
FName = Environ("Temp") & "\" & ModuleName & ".bas"
If OverwriteExisting = True Then
    ''''''''''''''''''''''''''''''''''''''''''''''''''''
    ' If OverwriteExisting is True, Kill
    ' the existing temp file and remove
    ' the existing VBComponent from the
    ' ToVBProject.
    ''''''''''''''''''''''''''''''''''''''''''''''''''''
    If Dir(FName, vbNormal + vbHidden + vbSystem) <> vbNullString Then
        Err.Clear
        Kill FName
        If Err.Number <> 0 Then
            CopyModule = False
            Exit Function
        End If
End If
End If
With ToVBProject.VBComponents
  .Remove .item(ModuleName)
End With
Else
' OverwriteExisting is False. If there is
' already a VBComponent named ModuleName,
' exit with a return code of False.

Err.Clear
Set VBComp = ToVBProject.VBComponents(ModuleName)
If Err.Number <> 0 Then
  If Err.Number = 9 Then
    ' module doesn’t exist. ignore error.
    Else
    ' other error. get out with return value of False
    CopyModule = False
    Exit Function
  End If
End If
End If
End If

' Do the Export and Import operation using FName
' and then Kill FName.

FromVBProject.VBComponents(ModuleName).Export Filename:=FName
ToVBProject.VBComponents.Import Filename:=FName
Kill FName
CopyModule = True
On Error GoTo 0
End Function
Sub AddProcedureToModule()
Dim VBProj As VBIDE.VBProject
Dim VBComp As VBIDE.VBComponent
Dim CodeMod As VBIDE.CodeModule
Set VBProj = ActiveWorkbook.VBProject
Set VBComp = VBProj.VBComponents("ThisWorkbook")
Set CodeMod = VBComp.CodeModule

NewModLine CodeMod, “Private Sub Workbook_Open()”
Private Sub OpenForm()
    frmChoice.Show
End Sub

Private Sub Workbook_BeforeClose(Cancel As Boolean)
    On Error Resume Next
    Application.CommandBars("Robust").Delete
    On Error GoTo 0
End Sub

Private Sub NewModLine(modl As VBIDE.CodeModule, code As String)
    modl.InsertLines modl.CountOfLines + 1, code
End Sub

Sub AddReference(strGUID As String)
    Dim theRef As Variant, i As Long
    strGUID = "{00020905-0000-0000-C000-000000000046}"
    On Error Resume Next
    Remove any missing references
    For i = ActiveWorkbook.VBProject.References.Count To 1 Step -1
        Set theRef = ActiveWorkbook.VBProject.References.item(i)
        If theRef.IsBroken = True Then
            ActiveWorkbook.VBProject.References.Remove theRef
        End If
    Next
    Err.Clear
    Add the reference
ActiveWorkbook.VBProject.References.AddFromGuid _
GUID:=strGUID, Major:=1, Minor:=0
Select Case Err.Number
    Case Is = 32813
        'Reference already in use. No action necessary
    Case Is = vbNullString
        'Reference added without issue
    Case Else
        'An unknown error was encountered, so alert the user
        MsgBox "A problem was encountered trying to" & vbCrLf _
          & "add or remove a reference in this file" & vbCrLf _
          & "Please check the " & "references in your VBA project!", _
          vbCritical + vbOKOnly, "Error!"
End Select
On Error GoTo 0
End Sub
Sub ListReferencePaths()
    'To determine full path and Globally Unique Identifier (GUID)
    'to each referenced library. Select the reference in the Tools\References
    'window, then run this code to get the information on the reference’s library
    Dim i As Long
    With ActiveSheet
        .Cells.Clear
        .Range("A1") = "Reference name"
        .Range("B1") = "Full path to reference"
        .Range("C1") = "Reference GUID"
    End With
    Cells(2, 1).Select
    For i = 1 To ThisWorkbook.VBProject.References.Count
        With ThisWorkbook.VBProject.References(i)
            ActiveCell = .Name
            ActiveCell.Offset(0, 1) = .FullPath
            ActiveCell.Offset(0, 2) = .GUID
        End With
        ActiveCell.Offset(1, 0).Select
    Next i
    On Error GoTo 0
End Sub
Private Sub UserForm_Initialize()
    Dim strName As String
    strName = WhichData
    'short tis no e3
    Select Case strName
        Case "AveTIS", "AveTIG", "Count", "Shortage", "Sep"
        Case "TIS"
        Case "Prom"
            cboRank.List = Array("E3", "E4", "E5", "E6", "E7", "E8")
        Case "Likelihood"
            cboRank.List = Array("E3", "E4", "E5", "E6", "E7", "E8")
    End Select
End Sub

Private Sub cmdChart_Click()
    Dim strName As String
    Application.ScreenUpdating = False
    If chkDeleteCharts = True Then KillCharts
    strName = WhichData
If strName = "Likelihood" Then
    LikChart CByte(Right(cboRank.Value, 1)), spnYOS.Value
Else
    MakeChart strName, CByte(Right(cboRank.Value, 1))
End If
Application.ScreenUpdating = True
End Sub
Private Sub spnYOS_Change()
    txtYOS.Value = spnYOS.Value
End Sub
Private Sub txtYOS_Exit(ByVal Cancel As MSForms.ReturnBoolean)
    spnYOS.Value = txtYOS.Value
End Sub
Private Function WhichData() As String
    Dim strName As String, lngSpace As Long
    strName = ActiveSheet.Name
    lngSpace = InStr(strName, " ")
    WhichData = Mid(strName, 1, lngSpace - 1)
End Function
Workbook
VERSION 1.0 CLASS
BEGIN
  MultiUse = -1 'True
END
Attribute VB_Name = "ThisWorkbook"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = False
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = True
'Programming by Robert W. Shuford, CNA
Option Explicit
Private Sub Workbook_Open()
  Const BUTTONS = 4
  Dim i As Byte
  On Error GoTo BarExists
  'Add toolbar and button
  For i = 1 To BUTTONS
    Application.CommandBars("New Data").Controls.Add Type:=msoControlButton, id:=2950, Before:=1
    'Set buttons
    NewButton 1, "New Data", "Import New Data", "LoopDirs"
    NewButton 2, "New Data", "Compile Multiple Files", "CompileAllData"
    NewButton 3, "New Data", "Sensitivity Data", "Robustness"
    NewButton 4, "New Data", "Export Modules", "XportMods"
  Next
  'Set buttons
  NewButton 1, "New Data", "Import New Data", "LoopDirs"
  NewButton 2, "New Data", "Compile Multiple Files", "CompileAllData"
  NewButton 3, "New Data", "Sensitivity Data", "Robustness"
  NewButton 4, "New Data", "Export Modules", "XportMods"
BarExists:
End Sub
Private Sub NewButton(item As Byte, bar As String, caption As String, macro As String)
  With Application.CommandBars(bar).Controls(item)
    .style = msoButtonCaption
    .caption = caption
    .OnAction = macro
    .BeginGroup = True
  End With
End Sub
Private Sub Workbook_BeforeClose(Cancel As Boolean)
  KillCmdBar
End Sub
Private Sub cboLikPG_Change()
    Application.ScreenUpdating = False
    If Not blnDisableEvents Then
        Worksheets("Likelihood Data").Activate
        ActiveSheet.Cells(1, 1).Activate
        Selection.AutoFilter Field:=3, Criteria1:=CInt(cboLikPG.Value)
        Worksheets("Likelihood Chart").Activate
    End If
    Application.ScreenUpdating = True
End Sub

Private Sub cboLikYOS_Change()
    Application.ScreenUpdating = False
    If Not blnDisableEvents Then
        Worksheets("Likelihood Data").Activate
        ActiveSheet.Cells(1, 1).Select
        Selection.AutoFilter Field:=4, Criteria1:=CInt(cboLikYOS.Value)
        Worksheets("Likelihood Chart").Activate
    End If
    Application.ScreenUpdating = True
End Sub
Sheet6

VERSION 1.0 CLASS
BEGIN
    MultiUse = -1 'True
END
Attribute VB_Name = “Sheet6”
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = False
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = True
'Programming by Robert W. Shuford, CNA
Option Explicit
Private Sub cboPG_Change()
    If Not blnDisableEvents Then Sheets(“TIS Pivot”).PivotTables(“PivotTable1”).PivotFields(“pg”).CurrentPage = CInt(cboPG.Value)
End Sub
References


[2] MILPERSMAN 1160-120, High Year Tenure, 20 October 2005


   Classifications And Occupational Standards Volume II Navy Enlisted

   Classifications (NECs), January 2004

[4] This information is administratively sensitive and was provided by the project

   sponsor.
List of figures

Figure 1. Schematic of the PIAP model and data processor ..............2
Figure 2. Yr0 database window............................................................4
Figure 3. Controller form..................................................................12
Figure 4. Increase Decrease Personnel Form.................................13
Figure 5. Increase Decrease Manpower Targets form.....................14
Figure 6. Database exceeds 1Gb .......................................................16
Figure 7. Database exceeds 1.5 Gb and is projected to fail ..............16
Figure 8. Import Compile and Sensitivity buttons.......................18
Figure 9. Years of service by paygrade .............................................20
Figure 10. Time in Service / Time in Grade....................................21
Figure 11. Prom chart.....................................................................22
Figure 12. Shortage chart.................................................................23
Figure 13. Sep Chart......................................................................24
Figure 14. TIS Chart ......................................................................25
Figure 15. Likelihood chart..............................................................26
Figure 16. Box plot charts ...............................................................28
Figure 17. Open Chart Form button ...............................................29
Figure 18. Robustness of promotion rates.....................................30
This page intentionally left blank.
List of tables

Table 1. Source Data Fields ...............................................................3
Table 2. RealAttrRates Fields.............................................................3
Table 3. Yr0 fields ...............................................................................6
Table 4. Service requirements for promotion (in months) ..............7
Table 5. Navy time to promotion benchmarks [4].........................7