CIM D0014629.A1/Final
August 2006

Forecasting the Marine Corps' Aviator Inventory

Michael J. Moskowitz
with
Theresa H. Kimble • Robert W. Shuford

CNA
4825 Mark Center Drive • Alexandria, Virginia 22311-1850

Approved for distribution: August 2006

Henry S. Griffis, Director
Workforce, Education and Training Team
Resource Analysis Division

This document represents the best opinion of CNA at the time of issue.
It does not necessarily represent the opinion of the Department of the Navy.

Approved for Pulic Release; Distribution Unlimited. Specific authority: N00014-05-D-0500.
For copies of this document call: CNA Document Control and Distribution Section at 703-824-2123.

Contents

Introduction

Knowing the future aviator inventory is of vital importance because the lead time for "creating" a new aviator is quite long. Commissioned officers typically finish their initial training at The Basic School (TBS) about 1 year after commissioning. Aviation officers then proceed to undergraduate flight training (UFT). Recent analyses have shown that the full training time, from TBS through attainment of Primary Military Occupational Specialty (PMOS), for a fully qualified aviator averages more than 3 years for certain aircraft. For example, the 38 new F/A-18 pilots in the May 2005 to April 2006 period averaged training times of 1,370 days, or 3.8 years.[1] Table 1 includes average training times for new aviation officers who received their PMOSs between May 2005 and April 2006.

Table 1. Training times for aviation PMOSs from TBS entry to PMOS attainment, May 2005 - April 2006

PMOS	Aircraft/ Occupation	Training time (in years)	Number of Marines trained
7509	AV-8B pilot	4.3	5
7523	F/A-18 pilot	3.9	46
7543	EA-6B pilot	3.7	7
7556	C-130 co-pilot	2.6	25
7562	CH-46 pilot	3.1	56
7563	UH-1 pilot	3.0	30
7565	AH-1 pilot	3.0	46
7566	CH-53E pilot	2.7	43
7588	EA-6B Electronic Warfare Officer	3.6	15

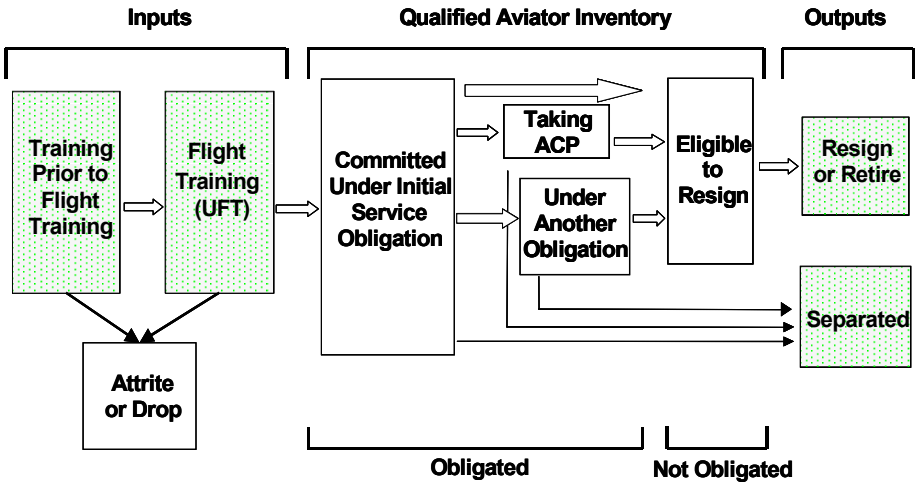
1. The training times in the analysis measure time from arrival at TBS to PMOS attainment. For aviators, this includes time spent in the Fleet Replacement Squadron (FRS) after UFT since they do not receive their final qualified PMOS until after FRS.

Making an aviator

At present, no model is being used to predict the inventory of aviators. Since it is crucial to identify future shortages in the aviation community well ahead of time, we have built a model to project future aviator inventory.

Figure 1 details an aviator's career progression.[2] As mentioned, an officer's military career begins by undergoing training, starting at TBS. Aviation officers enter UFT after completing TBS. At any point during training, a potential aviator may attrite or may switch to a non-aviation specialty and never enter the inventory of qualified aviators. Those who do complete UFT receive their wings and begin serving their Initial Service Obligation (ISO), which is currently a 6-year obligation for all Naval Flight Officers (NFOs) as well as helicopter and fixed-wing pilots, and an 8-year obligation for jet pilots.

Figure 1. Process flow of aviator inventory



2. This diagram and the related discussion by Major William B. Lambert are from CNA's Manpower Critical Indicators Study (CNA Research Memorandum D0006494.A2).

After the completion of the ISO, an aviator may take one of two tracks. In one track, aviators re-obligate by contracting for Aviation Continuation Pay (ACP) or some other program that incurs an obligation (SEP, Tuition Assistance, etc.).[3] If an aviator does not re-obligate, he or she remains in the Marine Corps unobligated. Aviators leave the service either voluntarily when they have no service obligations remaining or involuntarily when they are twice passed over to promotion to a certain grade, regardless of their obligation status. To voluntarily leave the service, an officer must provide notice of his or her intentions a minimum of 4 months and a maximum of 14 months before the desired resignation date. Officers may provide notice of resignation while still obligated as long as the resignation date comes after the officer's obligation has ended.

Figure 1 can also be understood as a process flow of aviator inventory, with aviators in training as inputs to the inventory. The "Qualified Aviator Inventory" category (shown by the bracketed header at the top of the figure) is the main inventory of interest. This category consists of all winged aviators, including both those still fulfilling obligations and those not under obligations and eligible to resign.

> *The important distinction between these groups of qualified aviators is that the Marine Corps can count on the continued service of obligated, qualified aviators until the obligation has ended, but there is uncertainty as to how long non-obligated aviators will remain in the inventory. The model's focus is the obligated, qualified inventory.*

Our model's assumption is that any non-obligated aviator will remain in the inventory for 6 months, to approximate the time necessary to process the resignation. Furthermore, though the qualified aviator inventory is the category of interest, the model focuses on those aviators who have not yet reached the rank of lieutenant colonel (LtCol) because there is little concern at present of shortages at or above this rank in the aviation community.

---

3. Some obligations can be undertaken before the ISO, and the obligations may be fulfilled concurrently.

Forecasting model

The Aviator Inventory Forecasting Model consists of Visual Basic programs in two Microsoft Excel workbooks. The first program acts on the data gathered from Operational Data Store Enterprise (ODSE) and progresses an aviator through a proposed career path. The second program interprets these data to produce an 8-year inventory estimate that is presented graphically. As such, this program is a predictive tool to calculate the number of projected qualified aviators over time and to help identify future aviator inventory shortfalls.

Data

The input data for the model are taken from the ODSE as a snapshot of the current aviator inventory. ODSE is a daily snapshot of Marine Corps personnel. The data drawn are limited to active duty aviators by selecting only certain component and strength codes and restricting the PMOSs to the 75XX occupational field (the full SQL code is included in appendix B). The user must make sure that these data fields are present and are compiled in an Excel spreadsheet in the following order:

- Social Security Number (SSN) or Proxy
- Present Grade
- Date of Rank Present Grade (DOR)
- Primary MOS (PMOS)
- PMOS Date of Attainment (PMOSDOA)
- Monitored Command Code (MCC)
- Incurred Obligated Service Code (IOS Code)
- Incurred Obligated Service Date (IOS Date)
- Pilot Designation Date (PDD)
- Aviation Service Entry Date (ASED)
- Commissioning Date (COMD)
- Aviation Continuation Pay Termination Date
- Aviation Continuation Pay Contract Month Quantity
- Aviation Continuation Pay Contract Agreement Effective Date.

Of all of the fields, the IOS Code and IOS Date are the most important to the model because they explicitly define the aviator's current

obligation. Since these data fields are relatively new and not entirely filled in for current aviators, we draw other data, such as the Pilot Designation Date, the Aviation Service Entry Date, and the Aviation Continuation Pay (ACP) fields to estimate current aviator obligations. Over time, as the IOS data fields become filled in completely, estimation of the length of the current obligation will no longer be required.

The ACP data are drawn from a remark field in ODSE, which can store multiple entries for each person. To ensure that the most recent ACP contract is captured, the remark sequence number should be used (the highest remark sequence number represents the most current data). This remark sequence number is not included in the spreadsheet of data for the model, but is used to identify the most current ACP contract in the ODSE draw (See Appendix B for the SQL program used to draw the data).

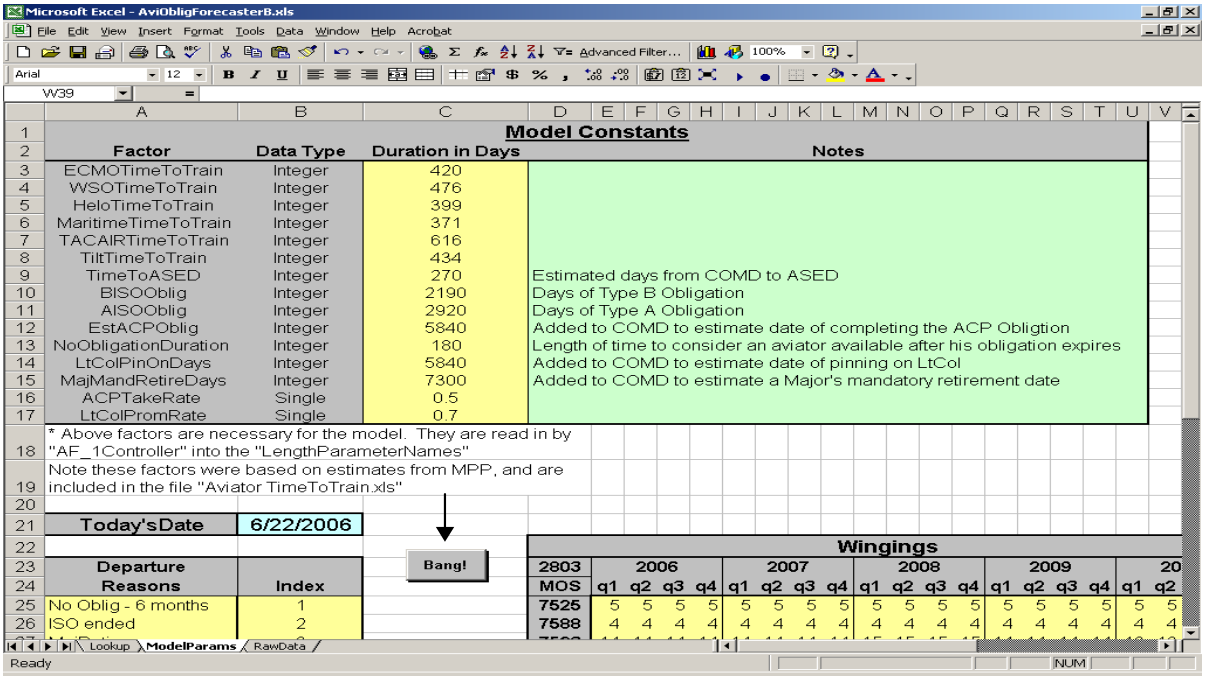
The data draw described above will provide information on all aviators currently in the database, including those with training PMOSs. The model does not use information on aviators in training, as it focuses on the qualified aviator inventory. To account for aviators that will be input into the inventory in the future, the model uses data on yearly planned wingings, broken down by aircraft.

In our draw on June 27, 2006, we pulled 4,822 aviators from ODSE. Of these, 1,205 were students, leaving us with a total of 3,617 aviators in our qualified inventory, all below the rank of lieutenant colonel.

Running the model

Once the data are drawn from ODSE, the next step is running the model. The Aviator Obligation Forecaster Spreadsheet has a button to start the “model (see the “Bang!” button in figure 2). This spreadsheet also contains the parameters used in the model calculations, which can be modified by the user to test the effects of varying the environment (see appendix A for a deeper discussion of the workings of the model).

Figure 2. Aviator Obligation Forecaster Spreadsheet



After the “Bang!” button is pressed, the model requests the location of the Excel spreadsheet containing the ODSE data (see figure 3). The user is then prompted to enter the number of runs (up to a

maximum of 8), which tells the model how many times it should run on the selected dataset (see figure 4).[4]

Figure 3. Prompt for location of ODSE data file

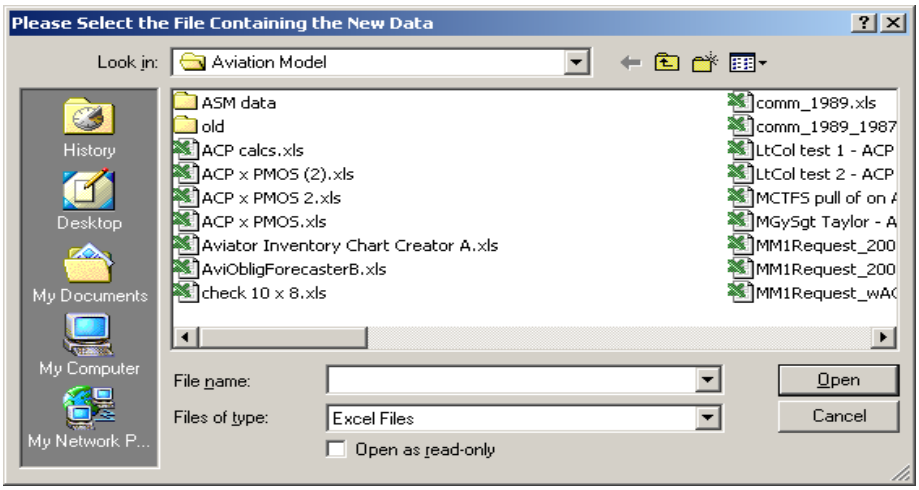
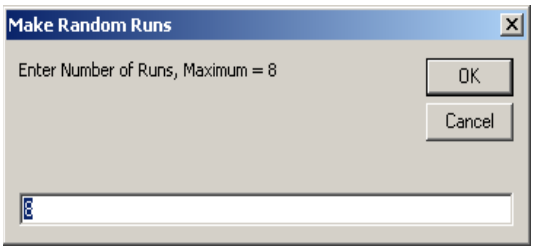


Figure 4. Prompt for number of runs for Aviator Obligation Forecaster



The model then performs the specified number of runs and creates spreadsheets for each run, titled Run1.xls to Run8.xls. These spreadsheets contain some of the original ODSE data on the aviators, as well as the calculated obligations and projected end dates. Data on

4. The model automatically calculates the maximum number of runs based on the number of wingings, the number of rows in the ODSE data, and the capacity of Microsoft Excel. The maximum will go up (down) if the wingings or ODSE data decrease (increase).

"future" aviators are also created based on the winging parameters included in the model.

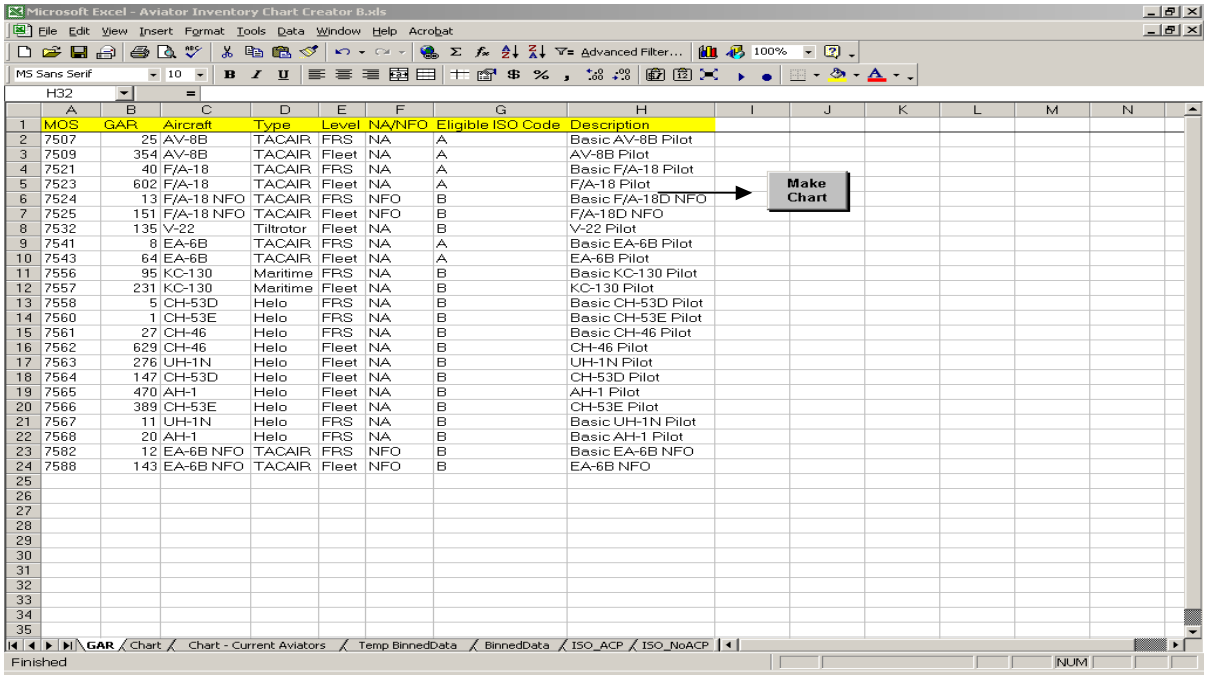
Due to the reliance on some assumptions and random number assignments, results can vary from one run to the next, even with the same data. Running the model multiple times at once allows the chart creator, described next, to average over the runs and smooth out any anomalies that arise. Therefore, we suggest running the forecaster the maximum number of times. Though this will cause the model to take more time to run and will use more disk space, both of these should be marginal increases. In our latest model run, it took just under 2 minutes for the forecaster to run, and the 8 spreadsheets created took up 22 megabytes of space.

In the second half of the model, the Aviator Inventory Chart Creator is run by pressing the "Make Chart" button (see figure 5). The user is prompted to enter the number of runs used in the forecaster, so the program knows how many files in which to find data. The chart creator pulls the data from each Run.xls file and compiles it. The program averages the numbers of aviators from each run and uses the date each aviator departs the model to create graphic depictions of the inventory of aviators for the next 8 years. Two charts are created: the current inventory of aviators (figure 6) and the total inventory, both current and future aviators (figure 7). In each of the charts, the aviator inventories are grouped by their obligation type. Table 2 lists the different obligation groups.

Table 2. Chart creator inventory categories

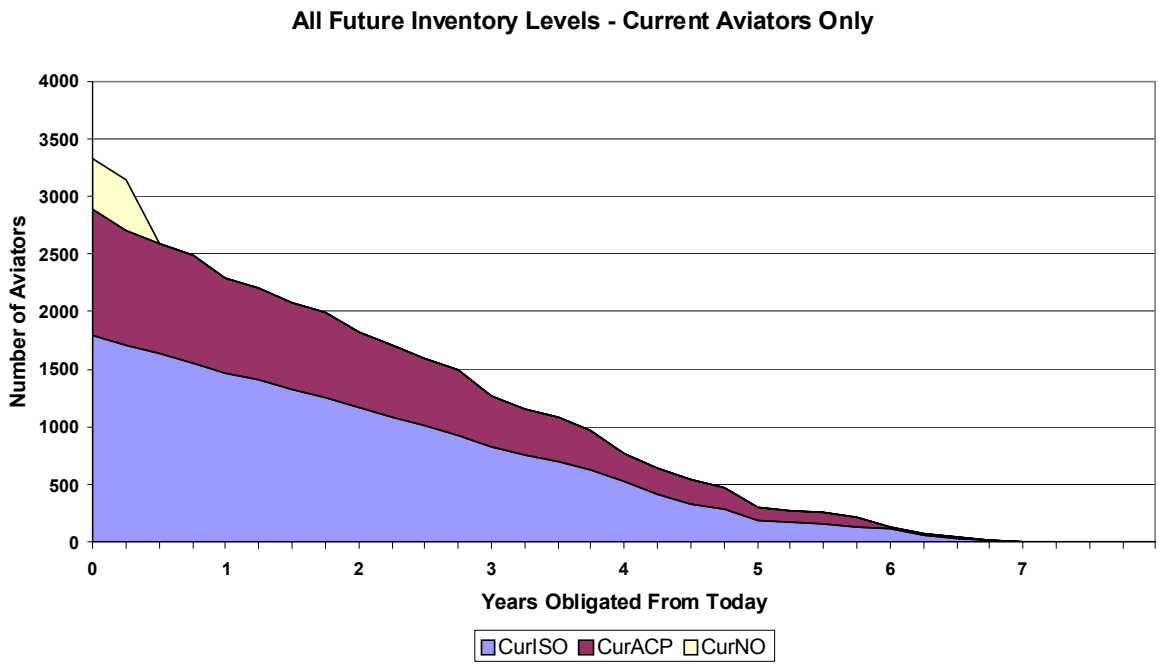
Obligation Category	Description
CurISO	Current aviators fulfilling their initial service obligation
ProjISO	"Future" aviators fulfilling their initial service obligation
CurACP	Current aviators fulfilling an obligation from accepting an ACP contract
ProjACP	"Future" aviators fulfilling projected ACP obligations.
CurNO	Current aviators with no remaining obligation
ProjNO	Future" aviators projected to have fulfilled their ISO but not projected to reobligate with ACP, thereby becoming nonobligated

Figure 5. Aviator Inventory Chart Creator Spreadsheet



MOS	GAR	Aircraft	Type	Level	NA/NFO	Eligible ISO Code	Description
7507	25	AV-8B	TACAIR	FRS	NA	A	Basic AV-8B Pilot
7509	354	AV-8B	TACAIR	Fleet	NA	A	AV-8B Pilot
7521	40	F/A-18	TACAIR	FRS	NA	A	Basic F/A-18 Pilot
7523	602	F/A-18	TACAIR	Fleet	NA	A	F/A-18 Pilot
7524	13	F/A-18 NFO	TACAIR	FRS	NFO	B	Basic F/A-18D NFO
7525	151	F/A-18 NFO	TACAIR	Fleet	NFO	B	F/A-18D NFO
7532	135	V-22	Tiltrotor	Fleet	NA	B	V-22 Pilot
7541	8	EA-6B	TACAIR	FRS	NA	A	Basic EA-6B Pilot
7543	64	EA-6B	TACAIR	Fleet	NA	A	EA-6B Pilot
7556	95	KC-130	Maritime	FRS	NA	B	Basic KC-130 Pilot
7557	231	KC-130	Maritime	Fleet	NA	B	KC-130 Pilot
7558	5	CH-53D	Helo	FRS	NA	B	Basic CH-53D Pilot
7560	1	CH-53E	Helo	FRS	NA	B	Basic CH-53E Pilot
7561	27	CH-46	Helo	FRS	NA	B	Basic CH-46 Pilot
7562	629	CH-46	Helo	Fleet	NA	B	CH-46 Pilot
7563	276	UH-1N	Helo	Fleet	NA	B	UH-1N Pilot
7564	147	CH-53D	Helo	Fleet	NA	B	CH-53D Pilot
7565	470	AH-1	Helo	Fleet	NA	B	AH-1 Pilot
7566	389	CH-53E	Helo	Fleet	NA	B	CH-53E Pilot
7567	11	UH-1N	Helo	FRS	NA	B	Basic UH-1N Pilot
7568	20	AH-1	Helo	FRS	NA	B	Basic AH-1 Pilot
7582	12	EA-6B NFO	TACAIR	FRS	NFO	B	Basic EA-6B NFO
7588	143	EA-6B NFO	TACAIR	Fleet	NFO	B	EA-6B NFO

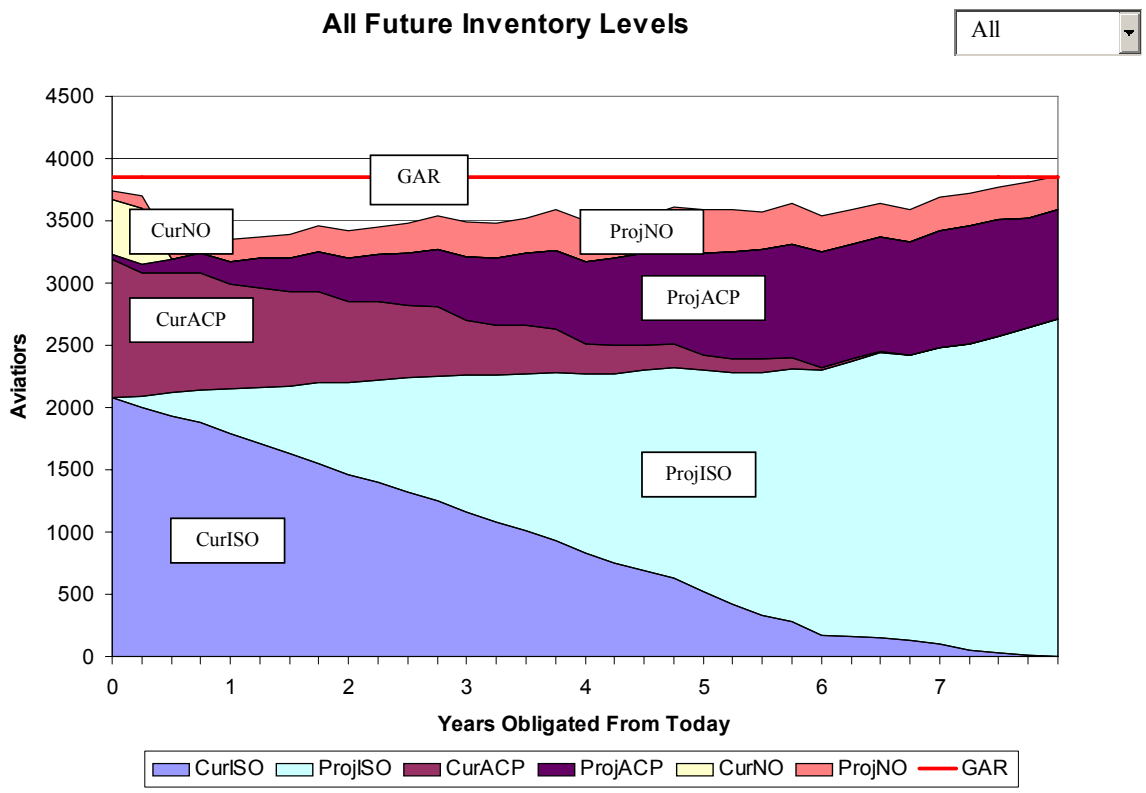
Figure 6. Inventory Chart—Current Aviators Only[a]



a. The timeline for the inventory is measured in years from today, so that 0 represents today, 1 represents a year from today, and so on.

Figure 6 shows current aviators only, grouped by their current obligation. This chart provides information on how long today's active duty aviators are obligated to remain on active duty under their current obligation. This is only one piece of the forecast, however, since aviators currently fulfilling their Initial Service Obligation (the CurISO group) can reobligate under ACP. These aviators exit this chart when their current obligation ends but show up in a new section, ProjACP, in figure 7.

Figure 7. Inventory chart—all aviators[a]



a. The timeline for the inventory is measured in years from today, so that 0 represents today, 1 represents a year from today, and so on.

Figure 7 shows the current aviator inventory but adds the estimated input from initial flight training based on the winging parameters, as well as those aviators that progress from their ISO to an ACP contract.

This chart is the full depiction of the qualified aviator inventory, providing a forecast of inventory levels out to 8 years in the future. In addition, the graph includes the current Grade Adjusted Recapitulation (GAR) requirement (from the FY06 Officer Victor GAR) as a basis of comparison for inventory levels.[5] The "All" group is a sum of the non-training aviation GARs for grades below O5.

In terms of understanding where a specific aviator is on the inventory chart, we can use a couple of examples: (1) a qualified aviator who currently has 2 years remaining on the initial service obligation and (2) a "future" aviator who will join the inventory as a helicopter pilot 1 year from today. The first aviator is currently in the inventory and is found in the CurISO group at 0. In 2 years, when the aviator completes his ISO, the model will either assign the aviator ACP, in which case he would move to the ProjACP category for the length of his contract, or he will remain without an obligation, in which case he would be in ProjNO for 6 months until exiting the model. The "future" aviator is not in today's inventory and will not appear on the chart until his ISO begins in 1 year. This aviator begins in the ProjISO category at 1. The ISO will last 6 years for a helicopter pilot, so this aviator will remain in the ProjISO group until 7 years from today, at which point he will either be assigned ACP and move to the ProjACP category or he will not re-obligate and will move to ProjNO when he leaves the model.

As previously stated, we are attempting to model the obligated aviator inventory. This forces us to make assumptions about how long aviators will remain in the service after their obligations have completed. The current assumption is that an aviator will remain for only 6 months, enough time to complete the resignation process. The idea behind this is that the Marine Corps cannot count on an unobligated officer to stay, even though many officers do stay well beyond their obligation. The inventory levels predicted by the model can therefore be viewed as conservative estimates of the overall aviator inventory.

---

5. The Marine Corps has traditionally measured shortages by comparing the onboard levels with the GAR. The GAR includes A-billet requirements as well as B-billets and P2T2 (patients, prisoners, trainees, and transients).

One issue that arises from having a set period of time an aviator stays after the obligation is complete (6 months in our example) is that all of the aviators currently in the qualified inventory that are non-obligated all leave the model at the same time. In our latest draw from which the graph is taken, there were 445 current aviators with no service obligation. These aviators, therefore, all depart the model in 6 months.

The default charts show expected inventories for all aviators. By using the dropdown box located on the total inventory chart (figure 7), specific PMOSs or groups can be displayed. When a group is selected, both of the charts are updated to show only the specified group, and the title of the charts reflects the selection. Figure 8 shows the inventory charts for a selected group—in this case, the inventory of AH-1 pilots. The full list of possible selections currently available follows:

Aircraft Grouping

- Helo
- Maritime
- TACAIR
- Tiltrotor

Aircraft

- AH-1
- AV-8B
- CH-46
- CH-53D
- CH-53E
- EA-6B
- EA-6B NFO
- F/A-18
- F/A-18 NFO
- KC-130
- UH-1N
- V-22

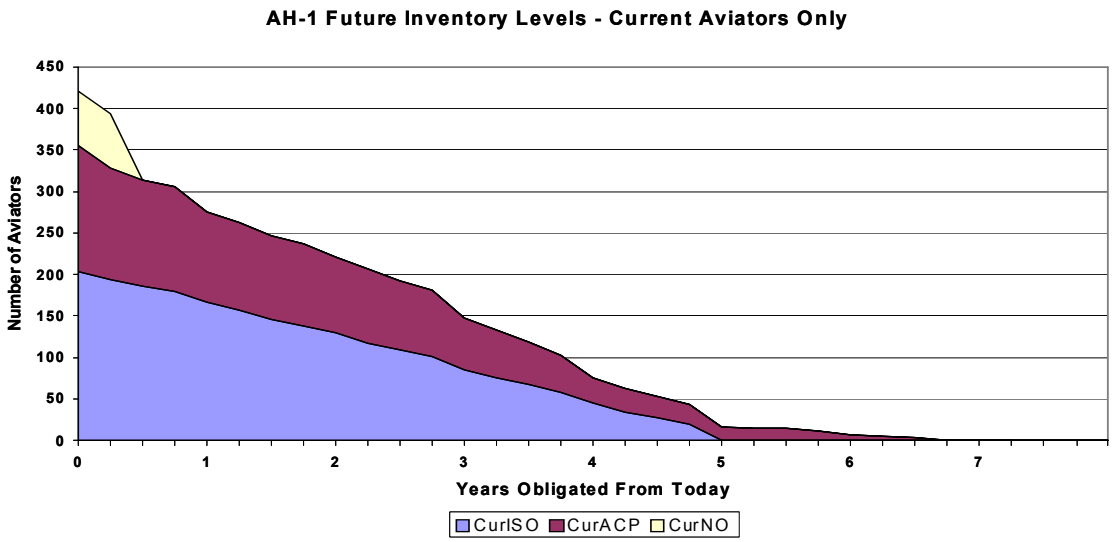
MOS

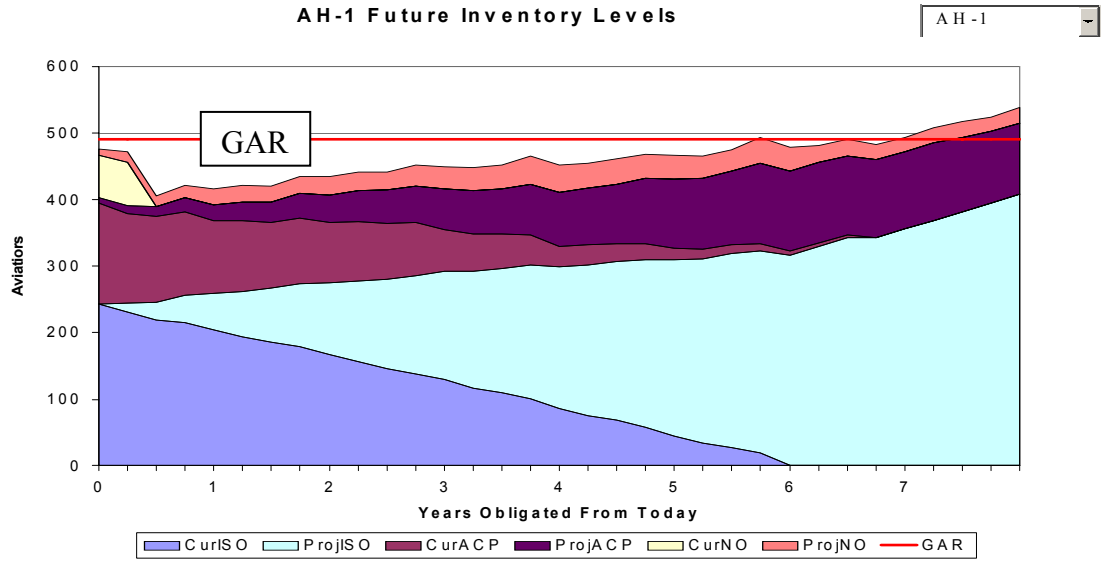
- 7509
- 7523
- 7525
- 7532
- 7543
- 7556
- 7557
- 7562
- 7563
- 7564
- 7565
- 7566
- 7588

The charts in figure 8 also show how this model can be used to analyze shortages. By comparing the AH-1 inventory to the GAR in the total inventory chart (bottom half of figure), decision-makers can see that the current onboard inventory of AH-1 pilots is not fully meeting

the requirement, but the inventory will be sufficient to meet the GAR in 6 years if conditions hold and future aviators are trained as laid out in the winging plans. Again, note that these inventory levels are estimates; however, given that the model uses some conservative assumptions, it appears likely that the AH-1 inventory will not drop below 85 percent of the GAR, which is the traditional measure of a critical shortage.

Figure 8. Figure 8. Chart for specified grouping—AH-1 pilots





Concluding comments

We believe that this model could benefit both the Aviation Planner in Officer Plans, Deputy Commandant, Manpower & Reserve Affairs (DC M&RA) and Aviation Support Manpower, Deputy Commandant, Aviation (DC AVN). The model provides a reasonable depiction of the future inventory of aviators if the current environment were to continue. The forecasting of future inventories allows the user to see apparent shortcomings well in advance, and gives planners an early warning so that they can relieve any shortages before they occur. The model's parameters, including the winging parameters, can be experimented with and changed to analyze the effects of changing the aviation environment.

Appendix A: A detailed description of the model

Current aviator inventory

The aviator model takes the current inventory of aviators and estimates the obligation, if any, of each. The IOS code provides this information; however, this data field is not completely filled in (the IOS data field was created in 2003 and was not backfilled), so we use alternate measures to determine the obligation. The model first checks to see when a person's ISO would have ended based on some assumptions and parameters included in the workbook. The algorithm for determining the ISO is shown below:

Flow for ISO creation:
Check IOS Date and IOS Code
If IOS Code = A, B, C, or D, aviator is under ISO
IOS Date will represent end of ISO

If no IOS Date, check winging date
For jet pilots, obligation is 8 years, so ISO ends 8 years after winging date
For other pilots, obligation is 6 years, so ISO ends 6 years after winging date

If no winging date, check aviation service entry date
To calculate ISO end date, add average time-to-train parameters for appropriate MOSs, and then add the 6- or 8-year obligation

Table from aviator model: (avg time to train in days)

ECMOTimeToTrain	420
WSOTimeToTrain	476
HeloTimeToTrain	399
MaritimeTimeToTrain	371
TACAIRTTimeToTrain	616
TiltTimeToTrain	434

If there is only a commissioning date
Add 270 days to commissioning date to approximate aviation service entry date
Add average time to train
Add 6- or 8-year obligation.

Next, the model determines whether the current aviator is taking ACP. The remarks section for Aviation Officer Continuation Pay (AOCP) in ODSE provides enough information to determine whether a person is currently receiving this bonus pay. If the resulting date from adding the number of contract months to the effective date falls in the future, the aviator is currently on ACP. However, if the termination date exists and occurs in the past, the contract has already been terminated and the person is no longer receiving any bonus pay under that contract.

The model also needs to chart future paths for these current aviators, including whether an aviator contracts for ACP and whether an aviator promotes to lieutenant colonel. This is done through two random number draws taken from the uniform distribution between 0 and 1. The results of these draws are compared with user-defined parameters. If the random number is less than the related parameter, a TRUE is assigned; otherwise a FALSE is assigned.

The current ACP parameter was set by using the current aviator inventory to compare personnel obligated under ACP with the eligible population. To estimate a take-rate for ACP, we used a ratio of the number of aviators currently obligated under an ACP contract to the total number of aviators who were not currently fulfilling their initial obligation, which served as a proxy for aviators eligible for ACP.[6] This resulted in an approximate take-rate of 50 percent, so the parameter was set to 0.5. This implies that 50 percent of aviators who reach eligibility for ACP will take it, and the model will assign an aviator as taking ACP if his/her random number draw was less than 0.5.

The determination of the current LtCol promotion parameter was slightly more complex, and may not be as easy to reproduce. The LtCol promotion parameter does not represent the likelihood of promoting from major to lieutenant colonel. Rather, it represents the likelihood of becoming a lieutenant colonel given that the aviator has completed UFT and received his or her wings. To calculate a reasonable estimate of this parameter, we looked at two cohorts of aviation officers – those commissioned in 1987 and those commissioned in 1989 – and looked at how many had reached the rank of lieutenant colonel in the aviation community. In both cohorts, the likelihood of attaining that rank, given that the officer was a winged aviator, was approximately 35 percent (or 0.35). However, the LtCol promotion parameter cannot simply be set to 0.35 because there is a direct tie to the ACP

6. This is not precisely the eligible population since only majors or major selects may contract for ACP under the current guidelines. Further, because we are using a snapshot of the aviator population, we have not captured those who were eligible for ACP but resigned before our snapshot, instead of re-obligating.

parameter. Given the model's assumptions about obligated and non-obligated career paths, an aviator can promote to lieutenant colonel only if he/she has re-obligated after the ISO by taking ACP.

Our data showed that about 50 percent of aviators who become eligible for ACP will take it, and that about 35 percent of all aviators who complete UFT eventually become lieutenant colonels. Therefore, to achieve the 35-percent promotion result, we set the promotion parameter to 0.7 (70 percent of the 50 percent who re-obligate with ACP would promote, resulting in 35 percent promoting to lieutenant colonel overall).

Using the random number draws and comparing them to the parameters, the model then assigns career paths based on the results of the draws. If an aviator is not assigned to take ACP, he or she will become non-obligated after the ISO is complete and will leave the model after 6 months. If the aviator does take ACP, the random number drawn for promotion to lieutenant colonel will come into play, and determine whether the aviator promotes. If the aviator does promote, he or she will depart the model as a lieutenant colonel (since we are currently not concerned with inventory levels at or above this rank), which will occur at the sixteenth year of service in our model. And if the aviator is not assigned a promotion in the model, he or she will depart the model at the mandatory retirement date for major, which is 20 years of service.

Future aviator inventory

To predict future inventory, the model needs to make assumptions on the number of aviators that will be winged in each future year. Data on planned wingings were provided for each of the next 5 years, broken down by PMOS.[7] The yearly winging data are assigned to quarters and used as model input to forecast future aviator populations by progressing them through a projected career just as current aviators were: the ISO lasts either 6 or 8 years depending on platform, and the ACP and promotion flow points work as described above. As new winging plans are formulated, the winging parameters in the model can be updated. Table 3 displays the current winging parameters as they are set up for use in the model.

7. We are grateful to the Aviation Planner (MPP-33) for providing this information.

Table 3. Winging parameters for Aviator Inventory Model

Wingings																				
2803[a]	2006				2007				2008				2009				2010			
MOS	q1	q2	q3	q4	q1	q2	q3	q4	q1	q2	q3	q4	q1	q2	q3	q4	q1	q2	q3	q4
7509	9	8	8	8	9	8	8	8	9	8	8	8	7	7	7	7	6	6	6	6
7523	14	14	14	14	14	14	14	14	15	15	15	15	14	14	14	14	13	13	13	13
7525	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
7532	4	3	4	3	4	3	4	3	4	3	4	3	7	7	7	7	7	6	6	6
7543	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
7557	8	7	8	7	8	7	8	7	8	7	8	7	8	7	8	7	8	7	8	7
7562	14	14	14	14	14	14	14	14	13	12	13	12	9	9	9	8	9	9	9	9
7563	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7
7564	3	3	3	2	3	3	3	2	3	3	3	3	3	3	3	3	3	3	3	3
7565	14	13	13	13	14	13	13	13	14	13	14	13	13	13	13	13	13	13	13	13
7566	10	10	10	10	10	10	10	10	10	9	10	9	10	9	9	9	10	9	9	9
7588	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4

a. This number represents the total number of aviators that will be created by the model run. The number of future aviators is used in calculating the maximum number of runs for the model.

The model assigns a "1st winging date" as the first day of the upcoming quarter (for example, in August it is set to September 1 of the same year). This date will be used for the first set of wingings, so in our example the first wingings would use the data from the 3rd quarter in 2006. However, the model is set up to use winging data for exactly 20 quarters, meaning there will be two quarters with no data. These quarters have been filled in using the data from the corresponding quarters of 2010, as a proxy for the unknown 2011 parameters (the first quarter of 2011 uses data from the first quarter of 2010). When updating the winging parameters, simply write over the data in the table with the new winging data. However, be sure that the first column of data is for the quarter that begins on the "1st winging date", and that any unknown data uses the corresponding quarter's data from the last year available to fill out the entire winging table.

Future aviators are created up to 8 years from the "1st winging date". Since we only have winging parameters for the next five years, aviators are winged according to the latest data available for all years beyond that point (the 2010 winging parameters are used for aviators winged 5, 6, 7, and 8 years in the future).

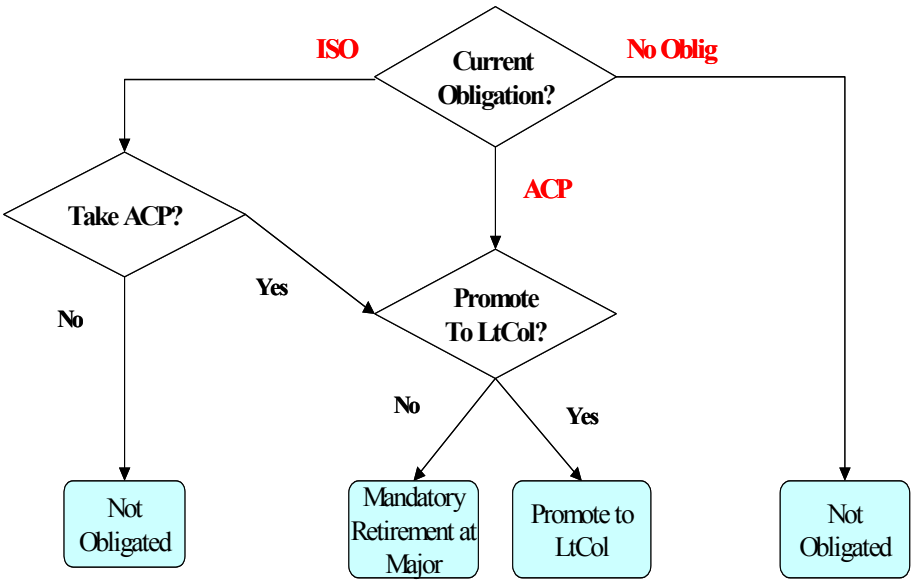
Model overview

Figure 9 provides an overview of the possible career paths of an aviator, current or future. Current aviators can begin the model on any of the three initial paths (shown in red in the figure) corresponding to the possible obligation statuses: fulfilling their ISO, contracted under ACP, or not obligated. Aviators under their ISO will have the model assign them an ACP draw, in which they will be projected to either take ACP or not. If they do not take ACP, they will no longer be obligated and will depart the model[8] after 6 months (or whatever the non-obligated parameter is set to). If they are assigned to take ACP, the model will assign them a promotion draw, where they can either promote to lieutenant colonel or not. If they promote, they will leave the model upon promotion; otherwise they will depart the model at the mandatory retirement point for majors. For current aviators who are contracted under ACP, the model provides them a promotion draw, and they will either promote and exit the model as lieutenant colonels or not promote and retire as majors. Lastly, current aviators who currently are not obligated depart the model almost immediately since the

8. Departing the model does not indicate leaving the Marine Corps. Aviators will leave the model 6 months after their obligations end, though many aviators will, in reality, remain in the service without any obligation. Exiting the model simply means that aviators are no longer in the bounds of the inventory of interest—that is, qualified obligated aviators who have not yet attained the rank of lieutenant colonel.

model applies the 6-month parameter for non-obligors and they exit the model. Future aviators all begin the model on the ISO path, and progress just as current aviators do.

Figure 9. Model overview



Appendix B: Computer codes for model

This appendix provides the computer codes necessary for the model. The first code is used to draw the current aviator data from the Operational Data Store Enterprise (ODSE). The second code is the Visual Basic for Applications (VBA) code used in the model's two Excel workbooks.

SQL code for ODSE draw:

```
BEGIN SQL
select      T1."SSN" 1,
            T1."PRESENT_GRADE_CODE",
            T1."PRESENT_GRADE_EFFECTIVE_DATE",
            T1."PRIMARY_MOS_CODE",
            T1."PRIMARY_MOS_ASSIGNMENT_DT",
            T1."PRESENT_MONITORED_COMMAND_CODE",
            T2."INCURRED_OBLIGATED_SERVICE_CD",
            T2."INCURRED_OBLIGATED_SERVICE_DT",
            T3."PILOT_DESIGNATION_EFFECT_DATE",
            T3."AVIATION_SERVICE_ENTRY_DATE" ,
            T1."DATE_OF_RANK_FIRST_COMMISSION",
            T1."ACTIVE_DUTY_FLAG",
            T4."AOCP_TERMINATION_DATE",
            T4."AOCP_CONTRACT_AGREE_MTH_QY",
            T4."CONTRACT_AGREEMENT_EFFECT_DATE",
 from     "ODSE"."INDIVIDUAL_MARINE" T1,
          "ODSE"."OFFICER" T2,
          "ODSE"."AVIATION_SERVICE" T3,
          "ODSE"."AVIATION_OFF_CONT_PAY_R942" T4
where   (T1."SSN" = T2."SSN") and (T1."SSN" = T3."SSN") and (T4."SSN" = T1."SSN") and
          and ((((T1."SSN" LIKE '0%') and ((T1."RECORD_STATUS_CODE" not in ('E', 'F')) and not
          (T1."RECORD_STATUS_CODE" is null) and
          (T1."COMPONENT_CODE" = '11') and
          (T1."STRENGTH_CATEGORY_CODE" not in ('G', 'I', 'J', 'O')))) and
          (T1."PRIMARY_MOS_CODE" LIKE '75%')) and
          (T1."PRESENT_GRADE_CODE" IN ('O1','O1E','O2','O2E','O3','O3E','O4')))
HAVING MAX(T4."REMARK_SEQUENCE_NUMBER_RMK_942")
order by "SSN" asc

END SQL
```

Visual Basic Code for Obligation Forecaster and Chart Creator[9]

AF_1 Controller

```
Option Explicit
Option Private Module
Option Base 1

'   Parameters are read into the arrays from
'   the ModelParams sheet and then are assigned to the
'   Public variables for use in the algorithms.

'   Arrays and vars for the length of training/obligations
Public LengthParameterNames(13) As String
Public LengthParameterValues(13) As Integer
Public ECMOTimeToTrain%
Public WSOTimeToTrain%
Public HeloTimeToTrain%
Public MaritimeTimeToTrain%
Public TACAIRTimeToTrain%
Public TiltTimeToTrain%
Public TimeToASED%
Public BISOOblig As Integer
Public AISOOblig As Integer
Public EstACPOblig%
Public NoObligDuration%
Public LtColPinOnDays%
Public MajMandRetireDays%

'   Rates
Public ACPTakeRate As Single
Public LtColPromRate As Single

Public CurrentSheetName$
Public TodaysDate As Date   'Set this to identify expired obligations
Public ISORowCounter As Integer
Public BlankObligRowNumber As Integer
Public LastRow As Integer
Public LastColumn As Integer

'Change this for number of columns in future wingings table
Public Const EXPLICIT_WINGING_QTRS As Byte = 20
Public Const NOOB As Byte = 1
Public Const ISO As Byte = 2
Public Const O4 As Byte = 3
Public Const O5 As Byte = 4
Public aryReason(4) As String
Sub CreateAviFcstProducts() 'Optional blnNew As Boolean = True)
'   Controller (main) for the model
    Dim StartTime As Double, ElapsedTime As Double
    Dim bytRuns As Byte, i As Byte, intRunsAllowed
    Dim wb As Workbook
```

9. This code was originally created by Major William B. Lambert, and then updated by Mr. Robert Shuford.

```
    StartTime = Timer
    Application.ScreenUpdating = False
    Set wb = ActiveWorkbook

    ResetToData
    'If blnNew Then NewData
    NewData
    intRunsAllowed = Int(65535 / (LastCell(ActiveSheet).Row + Range("FutWings").Value - 1))
    Call ReadInParameters

    On Error GoTo err_exit
    bytRuns = InputBox("Enter Number of Runs, Maximum = " & intRunsAllowed, "Make Random Runs",
intRunsAllowed)
    Do While bytRuns > intRunsAllowed
      bytRuns = InputBox("A Maximum of " & intRunsAllowed & " Runs Allowed", "A Little Ambitious There,
Bub")
    Loop
    On Error GoTo 0

    MakeDates
    IOSCodeG

    For i = 1 To bytRuns
        Call AddSheets
        Call InsertCleanObligHeaders
        Call PasteRawData
        Call GenerateFutureObligations
        Call FillObligData
        Call FormatSheet
        Application.DisplayAlerts = False
        Sheets("CleanOblig").Move
        ActiveWorkbook.SaveAs Filename:=ActiveWorkbook.Path & "Run" & i & ".xls"
        ActiveWorkbook.Close
        wb.Activate
    Next

    ElapsedTime = Timer - StartTime
    'If blnNew Then
    MsgBox "This routine took " & Format(ElapsedTime, "0.00") & " seconds to run."
err_exit:
    Set wb = Nothing
    Application.ScreenUpdating = True
End Sub
Private Sub ReadInParameters()
    Dim i%

'   Fill in array names and values from ModelParams sheet
    Worksheets("ModelParams").Activate
    Range("params").Activate
    For i = 1 To 13
        LengthParameterNames(i) = ActiveCell.Offset(i, 0).Value
        LengthParameterValues(i) = ActiveCell.Offset(i, 2).Value
    Next

    ACPTakeRate = Sheets("ModelParams").Cells(16, 3).Value
    LtColPromRate = Sheets("ModelParams").Cells(17, 3).Value
    TodaysDate = Sheets("ModelParams").Cells(21, 2).Value

    Range("reasons").Activate
```

```
    For i = 1 To 4
        aryReason(i) = ActiveCell.Offset(i, 0)
    Next

'   Assigned back to Public variables for ease of reading the
'   algorithms
    ECMOTimeToTrain = LengthParameterValues(1)
    WSOTimeToTrain = LengthParameterValues(2)
    HeloTimeToTrain = LengthParameterValues(3)
    MaritimeTimeToTrain = LengthParameterValues(4)
    TACAIRTimeToTrain = LengthParameterValues(5)
    TiltTimeToTrain = LengthParameterValues(6)
    TimeToASED = LengthParameterValues(7)
    BISOOblig = LengthParameterValues(8)
    AISOOblig = LengthParameterValues(9)
    EstACPOblig = LengthParameterValues(10)
    NoObligDuration = LengthParameterValues(11)
    LtColPinOnDays = LengthParameterValues(12)
    MajMandRetireDays = LengthParameterValues(13)
End Sub
Private Sub FormatSheet()
    Sheets("CleanOblig").Cells(1, 1).Activate
    ActiveCell.CurrentRegion.Select
    With Selection
        .HorizontalAlignment = xlLeft
        .EntireColumn.AutoFit
    End With
End Sub
Private Sub ResetToData()
    Application.DisplayAlerts = False
    On Error Resume Next
    Sheets("CleanOblig").Delete
    On Error GoTo 0
    Application.DisplayAlerts = True
End Sub
Private Sub GenerateFutureObligations()
'   Necessary arrays
    Dim WingingDate As Date
    Dim TempWingingDate As Date
    Dim intNum_PMOS As Integer, intWingings As Integer
    Dim WingingPMOS() As String
    Dim AviatorsPerWinging() As Integer
    Dim bytMaxYr As Byte

    Dim FutRowCounter%
    Dim AviCounter%
    Dim TempPMOS$

    Dim i%
    Dim j%
    Dim k%
    Dim n%
    Dim m%

'   Read in array data from ModelParams.
    Worksheets("ModelParams").Activate
    Range("mos_start").Activate
    intNum_PMOS = ActiveCell.CurrentRegion.Rows.Count - 3
```

```
    ReDim WingingPMOS(intNum_PMOS) As String
    ReDim AviatorsPerWinging(intNum_PMOS, EXPLICIT_WINGING_QTRS) As Integer
    For i = 1 To intNum_PMOS
        WingingPMOS(i) = """ & ActiveCell.Offset(i, 0)
        For j = 1 To EXPLICIT_WINGING_QTRS
            AviatorsPerWinging(i, j) = ActiveCell.Offset(i, j)
        Next
    Next
    WingingDate = Range("wing_dt").Value

    Worksheets("CleanOblig").Activate
    AviCounter = 1
    Call SetLastRow
    FutRowCounter = LastRow + 1

    For i = 1 To intNum_PMOS  'This is for each row, corresponding to each PMOS
        TempWingingDate = WingingDate
        TempPMOS = WingingPMOS(i)
        For j = 0 To 7      '8 years per PMOS
            bytMaxYr = Application.WorksheetFunction.Min(4, j)
            For m = 1 To 4  'Winging dates
                intWingings = AviatorsPerWinging(i, bytMaxYr * 4 + m)
                If intWingings > 0 Then
                    For k = 1 To intWingings
                        With Worksheets("CleanOblig")
                            .Cells(FutRowCounter, 1).Value = "F" & Format(AviCounter, "000000000")
                            .Cells(FutRowCounter, 2).Value = "O2"
                            .Cells(FutRowCounter, 4).Value = TempPMOS
                            .Cells(FutRowCounter, 5).Value = TempWingingDate
                            .Cells(FutRowCounter, 7).Value = "ZZZ"
                        .Cells(FutRowCounter, 8).Value = TempWingingDate + GetISOLength(Right(TempPMOS,
4))
                            .Cells(FutRowCounter, 9).Value = TempWingingDate
                            .Cells(FutRowCounter, 10).Value = TempWingingDate - TimeToASED
                            .Cells(FutRowCounter, 11).Value = TempWingingDate - (GetTTT(Right(TempPMOS,
4)) + TimeToASED)
                        End With
                        AviCounter = AviCounter + 1
                        FutRowCounter = FutRowCounter + 1
                    Next k
                    TempWingingDate = DateAdd("m", 3, TempWingingDate)
                End If
            Next m
        Next j
    Next i
End Sub
Private Function GetISOLength(tPMOS As String) As Integer
    GetISOLength = BISOOblig
    Select Case tPMOS
        Case 7507 To 7523, 7541 To 7543
            GetISOLength = AISOOblig
        Case Is >= 7597
            GetISOLength = ""
        End Select
End Function
Private Function GetTTT(tPMOS As String) As Integer
    Select Case tPMOS
        Case 7507 To 7523, 7541, 7543
            GetTTT = TACAIRTimeToTrain
        Case 7524, 7525
```

```
            GetTTT = WSOTimeToTrain
        Case 7531, 7532
            GetTTT = TiltTimeToTrain
        Case 7556, 7557
            GetTTT = MaritimeTimeToTrain
        Case 7558 To 7568
            GetTTT = HeloTimeToTrain
        Case 7582, 7588
            GetTTT = ECMOTimeToTrain
        Case 7580, Is >= 7597
            GetTTT = 0
        End Select
End Function
Private Sub MakeDates()
    Dim dtDate As Date, c As Range, lngLastrow As Long, i As Byte
    Dim aryCols As Variant

    Worksheets(3).Activate
'    Range("M:N,R:U").Select
'    Selection.Delete Shift:=xlToLeft

    lngLastrow = LastCell(ActiveSheet).Row
    aryCols = Array(3, 5, 8, 9, 10, 11, 13, 15)

    For i = LBound(aryCols, 1) To UBound(aryCols, 1)
        Range(Cells(2, aryCols(i)), Cells(lngLastrow, aryCols(i))).Select
        For Each c In Selection
            If c <> "" Then
                dtDate = c.Value
                c = dtDate
            End If
        Next
        Selection.NumberFormat = "mm/dd/yy"
    Next
End Sub
Private Sub IOSCodeG()
    Dim lngLastrow As Long, i As Long, dtEndACP As Date
    lngLastrow = LastCell(ActiveSheet).Row
    TodaysDate = Date
    For i = 2 To lngLastrow
        If Cells(i, 7) = "" And Cells(i, 13) = "" And Cells(i, 15) <> "" Then
            dtEndACP = DateAdd("m", Cells(i, 14), Cells(i, 15))
            If DateAdd("m", Cells(i, 14), Cells(i, 15)) > TodaysDate Then
                Cells(i, 7) = "G_fake"
                Cells(i, 8) = DateAdd("m", Cells(i, 14), Cells(i, 15))
            End If
        End If
    Next
End Sub
```

AF_2 AddSheets

```
Option Explicit
Option Private Module
Sub NewData()
   Dim i As Integer, strPathFile As String ', strFile As String
   Dim wb As Workbook
   Set wb = ActiveWorkbook

   Application.DisplayAlerts = False
   For i = wb.Sheets.Count To 3 Step -1
      Sheets(i).Delete
   Next
   Application.DisplayAlerts = True

   strPathFile = FindFile(ActiveWorkbook.Path, "Please Select the File Containing the New Data", _
         "Excel Files", "*.xls")
   If strPathFile = "" Then End
   Workbooks.Open Filename:=strPathFile 'strFile

   Application.DisplayAlerts = False
   For i = Sheets.Count To 2 Step -1
      Sheets(i).Delete
   Next

   Sheets(1).Move after:=wb.Sheets(2)
End Sub
Sub AddSheets()
'  This procedure renames the original data sheet
'  as "RawData" and adds a sheet to the end of the
'  workbook, naming it CleanOblig
   Worksheets(3).Name = "RawData"
   ActiveWorkbook.Sheets.Add after:=Worksheets(ActiveWorkbook.Sheets.Count)
   ActiveSheet.Name = "CleanOblig"
End Sub
Sub InsertCleanObligHeaders()
   Sheets("CleanOblig").Cells(1, 1).Select
   With ActiveCell
'  Column headers for pasted raw data
      .Offset(0, 0) = "SSN"
      .Offset(0, 1) = "Grade"
      .Offset(0, 2) = "DOR"
      .Offset(0, 3) = "PMOS"
      .Offset(0, 4) = "PMOSDOA"
      .Offset(0, 5) = "MCC"
      .Offset(0, 6) = "IOSCode"
      .Offset(0, 7) = "IOSDate"
      .Offset(0, 8) = "PDD"
      .Offset(0, 9) = "ASED"
      .Offset(0, 10) = "COMD"

'  Headers for data derived in AF_5FillObligFields / FillAviData
      .Offset(0, 11) = "Aircraft"
      .Offset(0, 12) = "AircraftType"
      .Offset(0, 13) = "Level"
      .Offset(0, 14) = "NA_NFO"

'  Headers for data derived in AF_5FillObligFields / FillPromotionData
      .Offset(0, 15) = "RollForPromotionToLtCol"
```

```
      .Offset(0, 16) = "PromoteToLtCol"
      .Offset(0, 17) = "EstLtColPinOnDate"
      .Offset(0, 18) = "MandatoryRetireMajor"

'   Headers for data derived in AF_5FillObligFields
      .Offset(0, 19) = "CurrentIOS"
      .Offset(0, 20) = "IOSType"
      .Offset(0, 21) = "EligISOCode"
      .Offset(0, 22) = "BasisISOExpiration"
      .Offset(0, 23) = "TheISOExpires"
      .Offset(0, 24) = "RollForACPAfterISO"
      .Offset(0, 25) = "TakeACPAfterISO"
      .Offset(0, 26) = "DateCommenceEstACP"
      .Offset(0, 27) = "TheACPExpires"
      .Offset(0, 28) = "TheNoObligExpires"
      .Offset(0, 29) = "DateDepartsModel"
      .Offset(0, 30) = "Reason"
   End With
End Sub
```

AF_3 FillSheets

```
Option Explicit
Option Private Module
Sub PasteRawData()
   SetBlankRowNumber ("CleanOblig")
'  Find the row number of last record in RawData and set that to
'  LastRow public variable
   SetLastRowNumber ("RawData")

'  Select data from RawData and copy it into CleanOblig
   Worksheets("RawData").Select
   Range(Cells(2, 1), Cells(LastRow, 12)).Select
   Selection.Copy Worksheets("CleanOblig").Cells(BlankObligRowNumber, 1)
End Sub
Function SetBlankRowNumber(SheetName As String)
   Worksheets(SheetName).Select
   Call ActivateNextBlankDown
   BlankObligRowNumber = ActiveCell.Row
End Function
Function SetLastRowNumber(SheetName As String)
   Worksheets(SheetName).Select
   Call SetLastRow
End Function
```

AF_4 FillObligFields

```
Option Explicit
Option Private Module
Private ObligRowCounter%
Private Grade$
Private DOR As Date
Private PMOS$
Private PMOSDOA As Date
Private MCC$
Private IOSCode$
Private IOSDate As Date
Private PDD As Date
Private ASED As Date
Private COMD As Date

'   Will be assigned based on PMOS
Private AC$
Private ACType$
Private Level$
Private NANFO$
Private EligISO$

Private Promote As Boolean
Private PinOnDate As Date
Private MajManRet As Date

Private IOSType$          'The type of IOS
Private ACPExp As Date      'The ACP expiration date
Private BasisISOExp$       'Basis for ISO exp date
Private ISOExp As Date      'The ISO expiration date
Private CurrentIOS$        'The current IOS, regardless of how estimated
Private NoObligExp As Date

Private TakeACP As Boolean
Private EstACPStartDate As Date
Private DateDepartModel As Date
Sub FillObligData()
'   This sub procedure calls below procedures to
'   fill in the data one row/record at a time
    Worksheets("CleanOblig").Activate
    Call SetLastRow

    Call FillAviData(LastRow)
    For ObligRowCounter = 2 To LastRow
        AC = Cells(ObligRowCounter, 12).Value
        ACType = Cells(ObligRowCounter, 13).Value
        Level = Cells(ObligRowCounter, 14).Value
        NANFO = Cells(ObligRowCounter, 15).Value
        On Error Resume Next
        EligISO = Application.WorksheetFunction.VLookup(Cells(ObligRowCounter, 4).Value, _
              Range("PMOS_Table"), 6, False)
        On Error GoTo 0
        Call AssignModuleVars
        If ACType <> "Stud" Then
            If COMD = 0 Then EstimateTheCOMD
            Call FillPromotionData
            Call FillACPData
```

```
            Call FillTheISO
            Call EstimateStartACP
            Call FillNoOblig
            Call FillDateDepartModel
            Call ClearModuleVars
        End If
    Next
End Sub
Sub AssignModuleVars()
    Grade = Cells(ObligRowCounter, 2).Value
    DOR = Cells(ObligRowCounter, 3).Value
    PMOS = Cells(ObligRowCounter, 4).Value
    PMOSDOA = Cells(ObligRowCounter, 5).Value
    MCC = Cells(ObligRowCounter, 6).Value
    IOSCode = Cells(ObligRowCounter, 7).Value
    IOSDate = Cells(ObligRowCounter, 8).Value
    PDD = Cells(ObligRowCounter, 9).Value
    ASED = Cells(ObligRowCounter, 10).Value
    COMD = Cells(ObligRowCounter, 11).Value
End Sub
Sub ClearModuleVars()
    Grade = ""
    DOR = 0#
    PMOS = ""
    PMOSDOA = 0#
    MCC = ""
    IOSCode = ""
    IOSDate = 0#
    PDD = 0#
    ASED = 0#
    COMD = 0#

    AC = ""
    ACType = ""
    Level = ""
    NANFO = ""
    EligISO = ""

    Promote = False
    PinOnDate = 0#
    MajManRet = 0#

    IOSType = ""
    ACPExp = 0#
    BasisISOExp = ""
    ISOExp = 0#
    CurrentIOS = ""
    TakeACP = False
    EstACPStartDate = 0#
End Sub
Sub FillTheISO()
'   Algorithm here to fill in the ISO dates where missing and needed
'   Change the IOSDate in the "If Then" as necessary

'   Don't look at students.  For future obligations use winging
'   plans becasue they are specific by PMOS.  If we tried to assign
'   ISOs for students we woiuld have to also assign PMOSs
    IOSType = ""
    BasisISOExp = ""
    ISOExp = 0#
```

```
    CurrentIOS = ""

    If Grade < "O5" Then
       Select Case IOSCode
          Case "F" To "H"     'Current ACP Obligation
             CurrentIOS = "ACP"
             IOSType = "ACP"
             BasisISOExp = ""
             ISOExp = 0#
             ACPExp = IOSDate
          Case "A" To "D"     'Current ISO Obligation
             CurrentIOS = "ISO"
             IOSType = "ISO"
             BasisISOExp = "ISO"
             ISOExp = IOSDate
             ACPExp = COMD + EstACPOblig
          Case "P"            'Current Tuition assistance obligation
             Call EstimateTheISO
             If ISOExp >= TodaysDate Then
                CurrentIOS = "ISO"
                IOSType = "EstISO_TA"
             Else
                CurrentIOS = "NoOblig"
                IOSType = "EstISO_Old_TA"
             End If
             '   BasisISOExp set in EstimateTheISO
             '   ISOExp set in EstimateTheISO
             ACPExp = COMD + EstACPOblig
          Case "ZZZ"          'BOGUS CODE TO IDENTIFY FUTURE AVIATORS
             CurrentIOS = "FutISO"
             IOSType = "FutISO"
             BasisISOExp = "Fut"
             ISOExp = IOSDate
             ACPExp = COMD + EstACPOblig
          Case ""             'No current obligation
             Call EstimateTheISO
             If ISOExp >= TodaysDate Then
                CurrentIOS = "ISO"
                IOSType = "EstISO"
             Else
                CurrentIOS = "NoOblig"
                IOSType = "EstISO_Old"
             End If
             '   BasisISOExp set in EstimateTheISO
             '   ISOExp set in EstimateTheISO
             ACPExp = COMD + EstACPOblig
       End Select

       Cells(ObligRowCounter, 20).Value = CurrentIOS
       Cells(ObligRowCounter, 21).Value = IOSType
       Cells(ObligRowCounter, 22).Value = EligISO
       Cells(ObligRowCounter, 23).Value = BasisISOExp
       If ISOExp <> 0# Then Cells(ObligRowCounter, 24).Value = ISOExp
       Cells(ObligRowCounter, 28).Value = ACPExp
    End If
End Sub
Sub EstimateTheISO()
'   This sub is only called if No ISO or ACP Obligation
'   date exists.
'   Estimate the ISO date, in decreasing order of confidence, from,
```

```
'   PDD, PMOS Date of Attainment (if they have an FRS PMOS),
'   ASED, and finally, commissioning date

'   Helper var
    Dim TempPreASEDTime As Date

'   ********* Estimate the ISO from PDD Block ***************************
    If PDD >= #1/1/1980# Then
        BasisISOExp = "PDD"
        '   Add 8 or 6 year obligations to PDD
        If EligISO = "A" Then
            ISOExp = PDD + AISOOblig
        ElseIf EligISO = "B" Then
            ISOExp = PDD + BISOOblig
        End If
'   ********** End ISO from PDD Block ************************

'   ********** Estimate the ISO from PMOS DOA Block ************************
    '   Estimate the ISO from PMOS Date of Attainment
    '   "FRS" is a proxy for the basic PMOSs, which should have been
    '   assigned at or close to winging - Fleet PMOS DOAs would
    '   be assigned on graduation from the FRS.
    '   All current FRS NAs should have been assigned these MOSs
    '   after the PMOSDOA field was instituted
    ElseIf Level = "FRS" And PMOSDOA >= #1/1/2003# Then
        BasisISOExp = "PMOSDOA"
        '   Add 8 or 6 year obligations to PMOSDOA
        If EligISO = "A" Then
            ISOExp = PMOSDOA + AISOOblig
'           Cells(ObligRowCounter, 31).Value = 1
        ElseIf EligISO = "B" Then
            ISOExp = PMOSDOA + BISOOblig
'           Cells(ObligRowCounter, 31).Value = 1
        End If
'   ********** End ISO from PMOS DOA Block ************************

'   ********** Estimate the ISO from ASED Block ************************
    '   Estimated ISO = ASED + time to train + ISO oblig
    '   The two factors here are the eligible obligation (A or B)
    '   and the estimated time-to-train, which depends on the type aircraft
    '   (Helo, TACAIR, Maritime, Tiltrotor, EA-6B, F/A-18)
    ElseIf ASED >= #1/1/1986# Then
        BasisISOExp = "ASED"
        If NANFO = "NA" Then
            Select Case ACType
                Case "Helo"
                    ISOExp = ASED + HeloTimeToTrain + BISOOblig
                Case "TACAIR"
                    ISOExp = ASED + TACAIRTimeToTrain + AISOOblig
                Case "Maritime"
                    ISOExp = ASED + MaritimeTimeToTrain + BISOOblig
                Case "Tiltrotor"
                    ISOExp = ASED + TiltTimeToTrain + BISOOblig
            End Select
        ElseIf NANFO = "NFO" Then
            Select Case PMOS
                Case "7524", "7525"
                    ISOExp = ASED + WSOTimeToTrain + BISOOblig
                Case "7582", "7588"
```

```
                ISOExp = ASED + ECMOTimeToTrain + BISOOblig
            End Select
        End If          '  End ASED Block
'   ********** End ISO from ASED Block ************************

'   ********** Estimate the ISO from COMD Block ************************
    '   Estimated ISO = COMD + time to ASED + time to train + ISO oblig
    '   This is calculated the same as e two factors here are the eligible obligation (A or B)
    '   and the estimated time-to-train, which depends on the type aircraft
    '   (Helo, TACAIR, Maritime, Tiltrotor, EA-6B, F/A-18)
    ElseIf COMD >= #1/1/1980# Then
        BasisISOExp = "COMD"
        TempPreASEDTime = COMD + TimeToASED
        If NANFO = "NA" Then
            Select Case ACType
                Case "Helo"
                    ISOExp = TempPreASEDTime + HeloTimeToTrain + BISOOblig
                Case "TACAIR"
                    ISOExp = TempPreASEDTime + TACAIRTimeToTrain + AISOOblig
                Case "Maritime"
                    ISOExp = TempPreASEDTime + MaritimeTimeToTrain + BISOOblig
                Case "Tiltrotor"
                    ISOExp = TempPreASEDTime + TiltTimeToTrain + BISOOblig
            End Select

        ElseIf NANFO = "NFO" Then
            Select Case PMOS
                Case "7524", "7525"
                    ISOExp = TempPreASEDTime + WSOTimeToTrain + BISOOblig
                Case "7582", "7588"
                    ISOExp = TempPreASEDTime + ECMOTimeToTrain + BISOOblig
            End Select
        End If
'   ********** End ISO from COMD Block ************************
    Else
        BasisISOExp = "UNABLE"
    End If
End Sub
Sub EstimateStartACP()
'   Generate the starting date for the ACP obligation as one day
'   after the ISO ending date.
    If Grade < "O5" And TakeACP = "True" _
        And IOSType <> "ACP" And Len(ISOExp) > 1 Then
        EstACPStartDate = ISOExp + 1
        If EstACPStartDate > 1 Then Cells(ObligRowCounter, 27).Value = EstACPStartDate
    End If
End Sub
Sub FillNoOblig()
    If CurrentIOS = "NoOblig" Then
        NoObligExp = TodaysDate + NoObligDuration
    ElseIf CurrentIOS = "ACP" Then
        If Promote Then
            NoObligExp = ACPExp + NoObligDuration
        Else
            NoObligExp = MajManRet
        End If
    ElseIf CurrentIOS = "ISO" Then
        If TakeACP Then
            NoObligExp = ACPExp + NoObligDuration
        Else
```

```
            NoObligExp = ISOExp + NoObligDuration
        End If
    ElseIf CurrentIOS = "FutISO" Then
        If TakeACP Then
            NoObligExp = ACPExp + NoObligDuration
        Else
            NoObligExp = ISOExp + NoObligDuration
        End If
    Else
        NoObligExp = 0#
    End If

    If NoObligExp <> 0# Then
        Cells(ObligRowCounter, 29).Value = NoObligExp
    End If
End Sub
Sub FillAviData(intLastRow As Integer)
    Range("L2").Formula = "=VLOOKUP($D2,PMOS_Table,COLUMN()-10,FALSE)"
    Range("L2").Select
    Selection.Copy
    Range("L2:O" & intLastRow).Select
    ActiveSheet.Paste
    Selection.Copy
    Selection.PasteSpecial Paste:=xlValues, Operation:=xlNone, SkipBlanks:= _
            False, Transpose:=False
    Application.CutCopyMode = False
    Selection.Replace What:="0", Replacement:="", LookAt:=xlWhole, _
            SearchOrder:=xlByRows, MatchCase:=False
    MissingMOS intLastRow
End Sub
Sub KillStudents() '(intLastRow As Integer)
    Range("M1").Activate
    Selection.CurrentRegion.Select
    Selection.Sort Key1:=Cells(1, 13), Order1:=xlAscending, Header:=xlYes, _
            OrderCustom:=1, MatchCase:=False, Orientation:=xlTopToBottom
    Columns("M:M").EntireColumn.Select
    Cells.Find(What:="Helo", after:=ActiveCell, LookIn:=xlValues, _
            LookAt:=xlWhole, SearchOrder:=xlByRows, SearchDirection:=xlNext, _
            MatchCase:=False).Select

End Sub
Sub FillPromotionData()
    Dim RollProm As Single

    If Grade < "O5" Then
        Randomize
        RollProm = Rnd

        If (RollProm <= LtColPromRate) Then
            Promote = True
            PinOnDate = LtColPinOnDays + COMD
        Else
            Promote = False
            MajManRet = MajMandRetireDays + COMD
        End If

        Cells(ObligRowCounter, 16).Value = Format(RollProm, "0.00")
        Cells(ObligRowCounter, 17).Value = Promote

        If PinOnDate <> 0# Then
```

```
            Cells(ObligRowCounter, 18).Value = PinOnDate
        End If

        If MajManRet <> 0# Then
            Cells(ObligRowCounter, 19).Value = MajManRet
        End If

    End If
End Sub
Sub FillACPData()
    Dim RollACP As Double

    If Grade < "O5" Then
        Randomize
        RollACP = Rnd
        TakeACP = (RollACP <= ACPTakeRate)

        Cells(ObligRowCounter, 25).Value = Format(RollACP, "0.00")
        Cells(ObligRowCounter, 26).Value = TakeACP
    End If
End Sub
Sub FillDateDepartModel()
    Dim strReason As String

    If Grade < "O5" Then
        If CurrentIOS = "NoOblig" Then
            DateDepartModel = NoObligExp
            strReason = aryReason(NOOB)
        ElseIf CurrentIOS = "ACP" Then
            If Promote Then
                DateDepartModel = PinOnDate
                strReason = aryReason(O5)
            Else
                DateDepartModel = MajManRet
                strReason = aryReason(O4)
            End If
        ElseIf CurrentIOS = "ISO" Then
            If TakeACP Then
                If Promote Then
                    DateDepartModel = PinOnDate
                    strReason = aryReason(O5)
                Else
                    DateDepartModel = MajManRet
                    strReason = aryReason(O4)
                End If
            Else
                DateDepartModel = NoObligExp
                strReason = aryReason(ISO)
            End If
        ElseIf CurrentIOS = "FutISO" Then
            If TakeACP Then
                If Promote Then
                    DateDepartModel = PinOnDate
                    strReason = aryReason(O5)
                Else
                    DateDepartModel = MajManRet
                    strReason = aryReason(O4)
                End If
            Else
                DateDepartModel = NoObligExp
```

```
            strReason = aryReason(ISO)
          End If
        End If
          Cells(ObligRowCounter, 30).Value = DateDepartModel
          Cells(ObligRowCounter, 31).Value = strReason
    End If
End Sub
Sub EstimateTheCOMD()
    Const TimeToO2 = 365 * 2
    Const TimeToO3 = 365 * 4
    Const TimeToO4 = 365 * 10
    Dim dtDOR As Date, strRank As String

    If ASED >= #1/1/1980# Then
        COMD = ASED - TimeToASED
' No ASED
    ElseIf PDD >= #1/1/1980# Then
        If NANFO = "NA" Then
            Select Case ACType
                Case "Helo"
                    COMD = PDD - HeloTimeToTrain - TimeToASED
                Case "TACAIR"
                    COMD = PDD - TACAIRTimeToTrain - TimeToASED
                Case "Maritime"
                    COMD = PDD - MaritimeTimeToTrain - TimeToASED
                Case "Tiltrotor"
                    COMD = PDD - TiltTimeToTrain - TimeToASED
                Case Else
            End Select
        ElseIf NANFO = "NFO" Then
            Select Case PMOS
                Case "7524", "7525"
                    COMD = PDD - WSOTimeToTrain - TimeToASED
                Case "7582", "7588"
                    COMD = PDD - ECMOTimeToTrain - TimeToASED
                Case Else
            End Select
        End If
' No PDD
    ElseIf IOSDate >= #1/1/1980# Then
        If NANFO = "NA" Then
            Select Case ACType
                Case "Helo"
                    If EligISO = "A" Then COMD = IOSDate - AISOOblig - HeloTimeToTrain - TimeToASED
                    If EligISO = "B" Then COMD = IOSDate - BISOOblig - HeloTimeToTrain - TimeToASED
                Case "TACAIR"
                    If EligISO = "A" Then COMD = IOSDate - AISOOblig - TACAIRTimeToTrain - TimeToASED
                    If EligISO = "B" Then COMD = IOSDate - BISOOblig - TACAIRTimeToTrain - TimeToASED
                Case "Maritime"
                    If EligISO = "A" Then COMD = IOSDate - AISOOblig - MaritimeTimeToTrain - TimeToASED
                    If EligISO = "B" Then COMD = IOSDate - BISOOblig - MaritimeTimeToTrain - TimeToASED
                Case "Tiltrotor"
                    If EligISO = "A" Then COMD = IOSDate - AISOOblig - TiltTimeToTrain - TimeToASED
                    If EligISO = "B" Then COMD = IOSDate - BISOOblig - TiltTimeToTrain - TimeToASED
                Case Else
            End Select
        ElseIf NANFO = "NFO" Then
            Select Case PMOS
                Case "7524", "7525"
```

```
                If EligISO = "A" Then COMD = IOSDate - AISOOblig - WSOTimeToTrain - TimeToASED
                If EligISO = "B" Then COMD = IOSDate - BISOOblig - WSOTimeToTrain - TimeToASED
            Case "7582", "7588"
                If EligISO = "A" Then COMD = IOSDate - AISOOblig - ECMOTimeToTrain - TimeToASED
                If EligISO = "B" Then COMD = IOSDate - BISOOblig - ECMOTimeToTrain - TimeToASED
            Case Else
        End Select
    End If
' Use Date of Rank
  Else
    dtDOR = Cells(ObligRowCounter, 3)
    If dtDOR >= #1/1/1980# Then
        Select Case Cells(ObligRowCounter, 2)
            Case "O1"
                COMD = dtDOR
            Case "O2"
                COMD = dtDOR - TimeToO2
            Case "O3"
                COMD = dtDOR - TimeToO3
            Case "O4"
                COMD = dtDOR - TimeToO4
            Case Else
        End Select
    End If
  End If
  Cells(ObligRowCounter, 11) = COMD
End Sub
Sub MissingMOS(intLastRow As Integer)
' Replaces #N/A errors in columns M through O.
' Compiles and presents a list of MOSs not in the lookup table.
  Dim c As Range, strMissing As String, strMOS As String
  Static blnNotFirstTime As Boolean

  Range("M2:O" & intLastRow).Select
  Selection.Replace What:="#N/A", Replacement:="", LookAt:=xlWhole, _
      SearchOrder:=xlByRows, MatchCase:=False
  For Each c In Range("L2:L" & intLastRow)
    With c
      If IsError(.Value) Then
        .ClearContents
        strMOS = .Offset(0, -8).Value
        If InStr(strMissing, strMOS) = 0 Then strMissing = strMissing & " " & strMOS
      End If
    End With
  Next
  If Not blnNotFirstTime And strMissing <> "" Then _
      MsgBox "The following PMOSs are not in the PMOS KEY lookup table:" & vbCrLf & strMissing
  blnNotFirstTime = True
End Sub
```

AF_RangeUtilities

```
Option Explicit
Option Private Module
Sub SetLastRow()
   Dim r As Integer
   Dim Flag As Boolean

   r = ActiveSheet.UsedRange.Rows.Count
   Flag = True
   Do
      If Not (IsEmpty(Cells(r, 1))) Then Flag = False
      r = r - 1
   Loop While Flag
   LastRow = r + 1
'  MsgBox "LastRow is " & LastRow
End Sub
Sub SetLastColumn()
   Dim c As Integer
   Dim Flag As Boolean

   c = ActiveSheet.UsedRange.Columns.Count
   Flag = True
   Do
      If Not (IsEmpty(Cells(c, 1))) Then Flag = False
      c = c - 1
   Loop While Flag
   LastColumn = c + 1
'  MsgBox "LastColumn is " & LastColumn
End Sub
Sub SelectActiveArea()
   Range(Range("A1"), ActiveCell.SpecialCells(xlLastCell)).Select
End Sub
Sub NumberLastRowAndColumn()
'  Doesn't work when you can't eliminate the extra rows/cells
   Call SelectActiveArea
   LastColumn = Selection.Columns.Count
   LastRow = Selection.Rows.Count
   MsgBox "LastRow is  " & LastRow
End Sub
Function RenameActiveSheet(SheetName As String)
   ActiveSheet.Name = SheetName
End Function
Sub ActivateNextBlankDown()
   ActiveSheet.Cells(1, 1).Select
   Do While Not IsEmpty(ActiveCell)
      ActiveCell.Offset(1, 0).Select
   Loop
End Sub
Sub SelectActiveColumn()
   Dim TopCell As Range
   Dim BottomCell As Range

   If IsEmpty(ActiveCell) Then Exit Sub
'  ignore error if activecell is in Row 1
   On Error Resume Next
   If IsEmpty(ActiveCell.Offset(-1, 0)) Then Set TopCell = ActiveCell Else Set TopCell =
ActiveCell.End(xlUp)
   If IsEmpty(ActiveCell.Offset(1, 0)) Then Set BottomCell = ActiveCell Else Set BottomCell =
```

```
ActiveCell.End(xlDown)
   On Error GoTo 0
   Range(TopCell, BottomCell).Select
End Sub
Function SetBlankRowNumber(SheetName As String)
   Worksheets(SheetName).Select
   Call ActivateNextBlankDown
   BlankObligRowNumber = ActiveCell.Row
End Function
Function SetLastRowNumber(SheetName As String)
   Worksheets(SheetName).Select
   Call SetLastRow
End Function
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(Row, Col)
End Function
```

AF_TextUtilities

```
Option Explicit
Option Private Module
Function ISLIKE(text As String, pattern As String) As Boolean
'   Returns TRUE if the first argument is like the second
    If text Like pattern Then ISLIKE = True Else ISLIKE = False
End Function
Function ShowDataType()
    MsgBox "This Cell's data type is " & TypeName(ActiveCell.Value)
End Function
Function GetType()
    TypeName (ActiveCell.Value)
End Function
Sub ClearBadCells()
    Dim c As Range
    On Error GoTo Errors
    For Each c In Selection
        If c.Value = "" Then c.ClearContents
    Next
Errors:
    Set c = Nothing
    On Error GoTo 0
End Sub
Function ExtractElement(Txt, n, Separator) As String
'   Returns the nth element of a text string, where the
'   elements are separated by a specified separator character
    Dim Txt1 As String, TempElement As String
    Dim ElementCount As Integer, i As Integer

    Txt1 = Txt
'   If space separator, remove excess spaces
    If Separator = Chr(32) Then Txt1 = Application.Trim(Txt1)

'   Add a separator to the end of the string
    If Right(Txt1, Len(Txt1)) <> Separator Then _
        Txt1 = Txt1 & Separator

'   Initialize
    ElementCount = 0
    TempElement = ""

'   Extract each element
    For i = 1 To Len(Txt1)
        If Mid(Txt1, i, 1) = Separator Then
            ElementCount = ElementCount + 1
            If ElementCount = n Then
'               Found it, so exit
                ExtractElement = TempElement
                Exit Function
            Else
                TempElement = ""
            End If
        Else
            TempElement = TempElement & Mid(Txt1, i, 1)
        End If
    Next i
    ExtractElement = ""
End Function
```

```
Function IntToString(n As Integer) As String
   IntToString = "" & n
End Function
Sub NumberToString()
'  Convert Numeric values to Strings

'   Call ShowDataType
'   MsgBox "IsNumeric = " & IsNumeric(ActiveCell.Value)
   If IsNumeric(ActiveCell.Value) Then
      ActiveCell.Value = "" & ActiveCell.Value
'      Call ShowDataType
   End If
End Sub
Sub StringToTime()
'   MsgBox "The current cell is of type " & TypeName(ActiveCell.Value)
   If Not IsNumeric(ActiveCell.Value) Then
      ActiveCell.Value = TimeValue(ActiveCell.Value)
   End If
'   MsgBox "After StringToTime cell is of type " & TypeName(ActiveCell.Value)
End Sub
Sub StringToDate()
'   MsgBox "The current cell is of type " & TypeName(ActiveCell.Value)
   If Not IsNumeric(ActiveCell.Value) Then
      ActiveCell.Value = DateValue(ActiveCell.Value)
   End If
'   MsgBox "After StringToTime cell is of type " & TypeName(ActiveCell.Value)
End Sub
Sub ReplaceLineFeed()
   ActiveCell.Replace What:=Chr(10), Replacement:="   ", LookAt:=xlPart, _
      SearchOrder:=xlByRows, MatchCase:=False
End Sub
Sub ReplaceParenStart()
   ActiveCell.Replace What:="(", Replacement:=" (", LookAt:=xlPart, _
      SearchOrder:=xlByRows, MatchCase:=False
End Sub
```

File_Utilities

```
Option Explicit
Option Private Module

' API Function
Private Declare Function GetOpenFileName Lib "comdlg32.dll" Alias _
   "GetOpenFileNameA" (pOpenfilename As OPENFILENAME) As Boolean

' Constants
Private Const ALLFILES = "All Files"

' Data types
Private Type MSA_OPENFILENAME
' Filter string used for the Open dialog filters.
' Use MSA_CreateFilterString() to create this.
' Default = All Files, *.*
   strFilter As String
' Initial Filter to display.
' Default = 1.
   lngFilterIndex As Long
' Initial directory for the dialog to open in.
' Default = Current working directory.
   strInitialDir As String
' Initial file name to populate the dialog with.
' Default = "".
   strInitialFile As String
   strDialogTitle As String
' Default extension to append to file if user didn't specify one.
' Default = System Values (Open File, Save File).
   strDefaultExtension As String
' Flags (see constant list) to be used.
' Default = no flags.
   lngFlags As Long
' Full path of file picked.  When the File Open dialog box is presented,
' if the user picks a nonexistent file, only the text in the "File Name" box is returned.
   strFullPathReturned As String
' File name of file picked.
   strFileNameReturned As String
' Offset in full path (strFullPathReturned) where the file name (strFileNameReturned) begins.
   intFileOffset As Integer
' Offset in full path (strFullPathReturned) where the file extension begins.
   intFileExtension As Integer
End Type
Private Type OPENFILENAME
   lStructSize As Long
   hWndOwner As Long
   hInstance As Long
   lpstrFilter As String
   lpstrCustomFilter As Long
   nMaxCustrFilter As Long
   nFilterIndex As Long
   lpstrFile As String
   nMaxFile As Long
   lpstrFileTitle As String
   nMaxFileTitle As Long
   lpstrInitialDir As String
   lpstrTitle As String
```

```
    Flags As Long
    nFileOffset As Integer
    nFileExtension As Integer
    lpstrDefExt As String
    lCustrData As Long
    lpfnHook As Long
    lpTemplateName As Long
End Type
Public Function FindFile(SearchPath As String, Title As String, FilterName As String, Filter As String) As
String
'Syntax: FindFile(SearchPath = Initial Path to set dialog to,
'         Title = Title of the dialog box,
'         Filtername - name for type of files to be located (E.G. "Excel Files"),
'         Filter - Wildcard Patern for Files (E.G. *.xls))
'   Returns the full path to File.
    Dim msaof As MSA_OPENFILENAME

    ' Set options for the dialog box.
    msaof.strDialogTitle = Title
    msaof.strInitialDir = SearchPath
    msaof.strFilter = MSA_CreateFilterString(FilterName, Filter)

    ' Call the Open dialog routine.
    MSA_GetOpenFileName msaof

    ' Return the path and file name.
    FindFile = Trim(msaof.strFullPathReturned)
End Function
Private Function MSA_CreateFilterString(ParamArray varFilt() As Variant) As String
' Creates a filter string from the passed in arguments.
' Returns "" if no arguments are passed.
' Expects an even number of arguments (filter name, extension),
' but if an odd number is passed in, it appends "*.*".
    Dim strFilter As String
    Dim intRet As Integer, intNum As Integer

    intNum = UBound(varFilt)
    If (intNum <> -1) Then
        For intRet = 0 To intNum
            strFilter = strFilter & varFilt(intRet) & vbNullChar
        Next
        If intNum Mod 2 = 0 Then
            strFilter = strFilter & "*.*" & vbNullChar
        End If
        strFilter = strFilter & vbNullChar
    Else
        strFilter = ""
    End If

    MSA_CreateFilterString = strFilter
End Function
Private Function MSA_GetOpenFileName(msaof As MSA_OPENFILENAME) As Integer
' Opens the Open dialog.
    Dim of As OPENFILENAME
    Dim intRet As Integer

    MSAOF_to_OF msaof, of
    intRet = GetOpenFileName(of)
    If intRet Then
```

```
        OF_to_MSAOF of, msaof
    End If
    MSA_GetOpenFileName = intRet
End Function
Private Sub MSAOF_to_OF(msaof As MSA_OPENFILENAME, of As OPENFILENAME)
' This sub converts from the Microsoft Access structure to the Win32 structure.
    Dim strFile As String * 512
' Initialize some parts of the structure.
    'of.hWndOwner = Application.hWndAccessApp
    of.hInstance = 0
    of.lpstrCustomFilter = 0
    of.nMaxCustrFilter = 0
    of.lpfnHook = 0
    of.lpTemplateName = 0
    of.lCustrData = 0

    If msaof.strFilter = "" Then
        of.lpstrFilter = MSA_CreateFilterString(ALLFILES)
    Else
        of.lpstrFilter = msaof.strFilter
    End If
    of.nFilterIndex = msaof.lngFilterIndex

    of.lpstrFile = msaof.strInitialFile _
        & String(512 - Len(msaof.strInitialFile), 0)
    of.nMaxFile = 511
    of.lpstrFileTitle = String(512, 0)
    of.nMaxFileTitle = 511
    of.lpstrTitle = msaof.strDialogTitle
    of.lpstrInitialDir = msaof.strInitialDir
    of.lpstrDefExt = msaof.strDefaultExtension
    of.Flags = msaof.lngFlags
    of.lStructSize = Len(of)
End Sub
Private Sub OF_to_MSAOF(of As OPENFILENAME, msaof As MSA_OPENFILENAME)
' This sub converts from the Win32 structure to the Microsoft Access structure.
    msaof.strFullPathReturned = Left(of.lpstrFile, InStr(of.lpstrFile, vbNullChar) - 1)
    msaof.strFileNameReturned = of.lpstrFileTitle
    msaof.intFileOffset = of.nFileOffset
    msaof.intFileExtension = of.nFileExtension
End Sub
```

RndmRun

```
Option Explicit
Sub MakeRuns()
   Application.ScreenUpdating = False
   Dim i As Byte, bytRuns As Byte

   On Error Resume Next
   Application.DisplayAlerts = False
   Sheets("CleanOblig").Delete
   Application.DisplayAlerts = True
   On Error GoTo err_exit
   bytRuns = InputBox("Enter Number of Runs", "Make Random Runs")
   On Error GoTo 0
   Application.DisplayAlerts = False
   For i = 1 To bytRuns
      CreateAviFcstProducts False
      Sheets("CleanOblig").Move
      ActiveWorkbook.SaveAs Filename:=ActiveWorkbook.Path & "Run" & i & ".xls"
      ActiveWorkbook.Close
   Next
err_exit:
   Application.DisplayAlerts = True
   Application.ScreenUpdating = True
End Sub
```

CC_1 Controller

```
Option Explicit
Option Base 1

'Public DataWithLabels As Range
'Public AllObligData As Range

Private DataSheets(7) As String
Private CriteriaFields(7, 3) As Integer  '3 fields per sheet
Private CriteriaValues(7, 3) As String   '3 values per sheet
Private DaysRangeNames(7, 3) As String   'Name of day range
Private DaysRangeStartCell(3) As String  'Start cell of Day range

Sub CreateCharts()
   Application.ScreenUpdating = False
   Dim i As Byte, bytRuns As Byte, wb As Workbook
   Set wb = ActiveWorkbook
   On Error Resume Next
   Application.DisplayAlerts = False
   Worksheets("CleanOblig").Delete
   On Error GoTo 0

   bytRuns = InputBox("Enter Number of Runs", "Random Runs")
   Workbooks.Open Filename:=ActiveWorkbook.Path & "\Run1.xls"
   Sheets("CleanOblig").Move Before:=wb.Sheets(1)
   Cells(LastCell(Worksheets("CleanOblig")).Row + 1, 1).Select
   For i = 2 To bytRuns
      Workbooks.Open Filename:=ActiveWorkbook.Path & "\Run" & i & ".xls"
      Range("A2", LastCell(Worksheets("CleanOblig"))).Select
      Selection.Copy
      wb.Activate
      Cells(LastCell(Worksheets("CleanOblig")).Row + 1, 1).Select
      ActiveSheet.Paste
      Workbooks("Run" & i & ".xls").Close
   Next

   Range("Runs").Value = bytRuns
   Call NameDataRange
   Call FillArrays
   Call ClearDataSheets
   Call FillSheets
   Call NameDayRanges
   ChartTitles
   Worksheets("CleanOblig").Delete
   Set wb = Nothing
   Application.DisplayAlerts = True
   Application.ScreenUpdating = True
End Sub
Sub NameDataRange()
   Dim n%
   Dim i%
   Dim TempLastRow%
   Dim TempName As String

'  Delete names
   n = ActiveWorkbook.Names.Count
   For i = n To 1 Step -1
```

```
        TempName = ActiveWorkbook.Names(i).Name
        If (TempName = "DataWithLabels") Then
            ActiveWorkbook.Names(i).Delete
        End If
    Next i

'   Rename data range
    Worksheets("CleanOblig").Activate
    Columns("AE:IV").EntireColumn.Delete
    Worksheets("CleanOblig").Range(Cells(1, 1), _
            LastCell(Worksheets("CleanOblig"))).Name = "DataWithLabels"
End Sub

Sub FillArrays()

    Dim i%
    Dim j%

'   Fill the DataSheets array from the Parameters sheet
    Worksheets("Parameters").Activate
    Range("B2").Select
    For i = 1 To 7
        DataSheets(i) = ActiveCell.Offset(i, 0).Value
    Next

'   Fill in range names and start cells from Parameters sheet
    Worksheets("Parameters").Activate
    Range("B12").Select
    For i = 1 To 7
        For j = 1 To 3
            DaysRangeNames(i, j) = ActiveCell.Offset(i, j).Value
            DaysRangeStartCell(j) = ActiveCell.Offset(11, j).Value
        Next
    Next

'   Fill in filter criteria fields from Parameters sheet
    Worksheets("Parameters").Activate
    Range("B27").Select
    For i = 1 To 7
        For j = 1 To 3
            CriteriaFields(i, j) = ActiveCell.Offset(i, j).Value
        Next
    Next

'   Fill in filter criteria values from Parameters sheet
    Worksheets("Parameters").Activate
    Range("B37").Select
    For i = 1 To 7
        For j = 1 To 3
            CriteriaValues(i, j) = ActiveCell.Offset(i, j).Value
        Next
    Next
End Sub

Sub ClearDataSheets()
'   Clear the previous data but retain formulas in the top row
'   of columns AE - AG

    Dim MySheet As Worksheet
```

```
    Dim i%
    Dim iLastRow%

    For i = 1 To 7
        Set MySheet = Worksheets(DataSheets(i))
        iLastRow = LastCell(MySheet).Row

        If iLastRow >= 2 Then MySheet.Range("A2", "AD" & iLastRow).ClearContents
        '   Retain the formulas in top row of calculated fields
        If iLastRow >= 3 Then MySheet.Range("AE3", "AG" & iLastRow).ClearContents
    Next
End Sub

Sub FillSheets()
    Dim MySheet As Worksheet
    Dim SourceRange As Range
    Dim FillRange As Range
    Dim TempLastRow%
    Dim i%

'   Select a sheet
    For i = 1 To 7
        Set MySheet = Worksheets(DataSheets(i))

        Dim f1%
        Dim f2%
        Dim f3%
        Dim v1$
        Dim v2$
        Dim v3$

'   Set criteria
        f1 = CriteriaFields(i, 1)
        v1 = CriteriaValues(i, 1)
        f2 = CriteriaFields(i, 2)
        v2 = CriteriaValues(i, 2)
        f3 = CriteriaFields(i, 3)
        v3 = CriteriaValues(i, 3)

'   Autofilter and copy data
        Range("DataWithLabels").AutoFilter Field:=f1, Criteria1:=v1
        Range("DataWithLabels").AutoFilter Field:=f2, Criteria1:=v2
        Range("DataWithLabels").AutoFilter Field:=f3, Criteria1:=v3
        Range("DataWithLabels").Copy Destination:=MySheet.Range("A1")
        Worksheets("CleanOblig").ShowAllData

'   Autofill the formulas of adjacent cells
        TempLastRow = LastCell(MySheet).Row
        Set SourceRange = MySheet.Range("AE2", "AG2")
        Set FillRange = MySheet.Range("AE2", "AG" & TempLastRow)
        If TempLastRow > 2 Then SourceRange.AutoFill Destination:=FillRange

'   Format Columns
        MySheet.Activate
        Range("A1").CurrentRegion.Select
        With Selection
            .HorizontalAlignment = xlLeft
            .EntireColumn.AutoFit
        End With
```

```
    Next
Worksheets("CleanOblig").AutoFilterMode = False
End Sub
Sub NameDayRanges()

    Dim TempRN$
    Dim TempSC$
    Dim i%
    Dim j%

    For i = 1 To 7
        Worksheets(DataSheets(i)).Activate
        For j = 1 To 3
            TempRN = DaysRangeNames(i, j)
            TempSC = DaysRangeStartCell(j)
            If TempRN <> "EMPTY" Then
                With Range(TempSC)
                    Range(.Cells(1, 1), .End(xlDown)).Name = TempRN
                End With
            End If
        Next
    Next

End Sub
Sub ChartTitles()
    Dim strMOS As String
    On Error GoTo err_exit
    strMOS = Range("MOS").Value

    Sheets("Chart For Slide Current").Select
    ActiveChart.ChartTitle.Text = strMOS & " Future Inventory Levels - Current Aviators Only"
    ActiveChart.Deselect

    Sheets("Chart For Slide").Select
    ActiveChart.ChartTitle.Text = strMOS & " Future Inventory Levels"
    ActiveChart.Deselect

    Worksheets("Charted").Activate
    ActiveSheet.ChartObjects("Chart 1").Chart.ChartTitle.Text = strMOS & " Future Inventory Levels"
err_exit:
    On Error GoTo 0
End Sub
```

RangeUtilities

```
Option Explicit

Sub SetLastRow()
   Dim r As Integer
   Dim Flag As Boolean

   r = ActiveSheet.UsedRange.Rows.Count
   Flag = True
   Do
      If Not (IsEmpty(Cells(r, 1))) Then Flag = False
      r = r - 1
   Loop While Flag
   LastRow = r + 1
'   MsgBox "LastRow is " & LastRow
End Sub
Function LastRow(SheetName As String) As Integer

   Dim r As Integer
   Dim Flag As Boolean

   Worksheets(SheetName).Activate
   r = ActiveSheet.UsedRange.Rows.Count
   Flag = True
   Do
      If Not (IsEmpty(Cells(r, 1))) Then Flag = False
      r = r - 1
   Loop While Flag
   LastRow = r + 1
'   MsgBox "LastRow is " & LastRow

End Function
Sub SetLastColumn()
   Dim c As Integer
   Dim Flag As Boolean

   c = ActiveSheet.UsedRange.Columns.Count
   Flag = True
   Do
      If Not (IsEmpty(Cells(1, c))) Then Flag = False
      c = c - 1
   Loop While Flag
   LastColumn = c + 1
'   MsgBox "LastColumn is " & LastColumn
End Sub


Sub SelectActiveArea()
   Range(Range("A1"), ActiveCell.SpecialCells(xlLastCell)).Select
End Sub

Sub NumberLastRowAndColumn()
'  Doesn't work when you can't eliminate the extra rows/cells

   Call SelectActiveArea
   LastColumn = Selection.Columns.Count
   LastRow = Selection.Rows.Count
```

```
   MsgBox "LastRow is  " & LastRow

End Sub

Function RenameActiveSheet(SheetName As String)
   ActiveSheet.Name = SheetName
End Function

Sub DeleteEmptyRowsWithinData()
   Dim lrow As Integer
   Dim r As Integer

   lrow = ActiveSheet.UsedRange.Rows.Count
   Application.ScreenUpdating = False
   For r = lrow To 1 Step -1
      If Application.WorksheetFunction.CountA(Rows(r)) = 0 _
         Then Rows(r).Delete
   Next r
End Sub

Sub ActivateNextBlankDown()
   ActiveSheet.Cells(1, 1).Select
   Do While Not IsEmpty(ActiveCell)
      ActiveCell.Offset(1, 0).Select
   Loop
End Sub

Sub SelectActiveColumn()
   Dim TopCell As Range
   Dim BottomCell As Range

   If IsEmpty(ActiveCell) Then Exit Sub
'  ignore error if activecell is in Row 1
   On Error Resume Next
   If IsEmpty(ActiveCell.Offset(-1, 0)) Then Set TopCell = ActiveCell Else Set TopCell =
ActiveCell.End(xlUp)
   If IsEmpty(ActiveCell.Offset(1, 0)) Then Set BottomCell = ActiveCell Else Set BottomCell =
ActiveCell.End(xlDown)
   Range(TopCell, BottomCell).Select

End Sub
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(Row, Col)
End Function
Function NextWingingDt()
    NextWingingDt = DateAdd("q", 1, Now)
    NextWingingDt = DateSerial(Year(NextWingingDt), Int(Month(NextWingingDt) / 3) * 3, 1)
End Function
```

RndmRun

```
Option Explicit
'Option Private Module
Public Sub RunRandom()
   Application.ScreenUpdating = False
   Dim c As Range, ws As Worksheet, wb As Workbook
   Dim i As Byte, bytRuns As Byte, strGroup As String, lngLast As Long
   Set ws = ActiveWorkbook.Worksheets("Check")
   Set wb = ActiveWorkbook

   bytRuns = InputBox("Enter Number of Runs", "Random Runs")
   On Error Resume Next
   wb.Names("All").Delete
   On Error GoTo 0
   wb.Names.Add Name:="All", RefersToLocal:= _

"=BinnedData!A3:A35,BinnedData!C3:D35,BinnedData!F3:H35,BinnedData!J3:L35,BinnedData!N3:O3
5,BinnedData!Q3:Q35,BinnedData!S3:U35,BinnedData!W3:AB35"

   For i = 1 To bytRuns
      Workbooks.Open Filename:=ActiveWorkbook.Path & "\Run" & i & ".xls"
      Sheets("CleanOblig").Move Before:=wb.Sheets(1)
      CreateCharts
         For Each c In Range("MOSs")
            strGroup = "" & c.Value
            Range("MOS").Value = strGroup
            lngLast = LastRow("Check")
            ws.Cells(lngLast + 1, 22) = strGroup
            ws.Cells(lngLast + 1, 23) = i

            Range("All").Copy
            ws.Cells(lngLast + 1, 1).Select
            Selection.PasteSpecial Paste:=xlValues

            ws.Range(Cells(lngLast + 1, 22), Cells(lngLast + 1, 23)).Select
            Selection.Copy
            ws.Range(Cells(lngLast + 1, 22), Cells(lngLast + 33, 23)).Select
            ActiveSheet.Paste
      Next
      Application.DisplayAlerts = False
      wb.Worksheets("CleanOblig").Delete
      Application.DisplayAlerts = True
   Next
   Application.CutCopyMode = False
   Set ws = Nothing
   Set wb = Nothing
   Application.ScreenUpdating = True
End Sub
Public Sub RunRandom2()
   Application.ScreenUpdating = False
   Dim c As Range, ws As Worksheet
   Dim bytRuns As Byte, strGroup As String, lngLast As Long
   Set ws = ActiveWorkbook.Worksheets("Check")

   On Error Resume Next
   ActiveWorkbook.Names("All").Delete
   On Error GoTo 0
   ActiveWorkbook.Names.Add Name:="All", RefersToLocal:= _
```

```
"=BinnedData!A3:A35,BinnedData!C3:D35,BinnedData!F3:H35,BinnedData!J3:L35,BinnedData!N3:O3
5,BinnedData!Q3:Q35,BinnedData!S3:U35,BinnedData!W3:AB35"

    For Each c In Range("MOSs")
        strGroup = "'" & c.Value
        Range("MOS").Value = strGroup
        lngLast = LastRow("Check")
        ws.Cells(lngLast + 1, 22) = strGroup

        Range("All").Copy
        ws.Cells(lngLast + 1, 1).Select
        Selection.PasteSpecial Paste:=xlValues

        Cells(lngLast + 1, 22).Select
        Selection.Copy
        ws.Range(Cells(lngLast + 1, 22), Cells(lngLast + 33, 22)).Select
        ActiveSheet.Paste
    Next

    Application.CutCopyMode = False
    Set ws = Nothing
    Application.ScreenUpdating = True
```

This Workbook

```
Private Sub Workbook_Open()
    Range("wing_dt").Value = NextWingingDt
End Sub
```

Sheet4 (Model Params)

```
Private Sub cmdBang_Click()
    CreateAviFcstProducts
End Sub
```

List of figures

CNA