Multi-parametric Numerical Simulation of Age-Specific Cancer Rates in Human Populations

by

John Kogel

Submitted to the Department of Electrical Engineering and Computer Science in Partial Fulfillment of the Requirements for the Degree of Master of Engineering in Electrical Engineering and Computer Science at the Massachusetts Institute of Technology

May 11, 2004

Author_____
Department of Electrical Engineering and Computer Science
May 11, 2004

Certified by_____
William G. Thilly
Thesis Supervisor

Accepted by_____
Arthur C. Smith

Multi-parametric Numerical Simulation of Age-Specific Cancer Rates in Human Populations

by

John Kogel

Submitted to the Department of Electrical Engineering and Computer Science

May 11, 2004

In Partial Fulfillment of the Requirements for the Degree of Master of Engineering in Electrical Engineering and Computer Science

ABSTRACT

The CancerFit computer program allows cancer researchers to analyze epidemiologic data describing the age-specific risk of cancer in terms of hypotheses about historical environmental risks, the heritability of cancer, the role of gender and the processes embedded in cancer formation. The program was based on the theories of Professor W. Thilly (MIT), Professor S. Morgenthaler (ETH) and their students. Written as a Fortran program by Prof. Morgenthaler it was transported into Java by David Hensle (MIT) who introduced a number of characteristics that enabled MIT students to perform basic parametric analyses for thesis and coursework in cancer epidemiology.

In this thesis, the CancerFit application has been extended to include new functionality that allows computation and subsequent analysis of the ratio of two age-specific cancer incidence or mortality datasets. Originally this ratio was proposed to compare the lifetime risks of children of parents with a specific form of cancer and of parents with at least one child with the same cancer; it does this task as intended. However, its use has permitted me to discover a previously unrecognized excess of colon cancer deaths in women relative to men in the 30-64 year age interval. As this is the same age interval for early breast cancer and ovarian cancer onset in women, this finding points to a more general cancer risk in pre-menopausal women than has been previously recognized.

Furthermore, the CancerFit program has been improved by permitting the cancer researcher to include historical age-specific survival rates, overall mortality rates, and reporting error rates when these are available. A user can now input data for each of these rates, which the program uses to adjust the mortality data to better approximate the age-specific rate of cancer appearance for the cohort studied. These improvements and new clinical data have permitted a clearer understanding of the age-specific risks and in the case of colorectal cancer appear to permit calculation of the critical parameters in this form of human cancer.

Thesis Supervisor: William G. Thilly
Title: Professor, Biological Engineering

Acknowledgments

I would like to thank Professor Thilly for introducing and teaching me about the world of cancer research. Besides an excellent and enthusiastic teacher, Professor Thilly has been the backbone of this thesis work since day one. Without his support, my contributions would not have been possible. I would also like to thank David Hensle, who introduced me to Professor Thilly and helped me become familiar with the CancerFit program and the code, as well as the theories behind the program. Finally I would like to thank everyone in the Thilly lab group from whom I learned a great deal and received helpful feedback on my work.

Contents

List of Figures

1 Background

Public health and census records have annually reported estimates of the population of the United States and also the number of persons recorded as dying from each of a long list of medical diagnoses. These data were organized in five year age intervals, 0-4, 5-9, ...,95-99, 100-104, since the Census Bureau undertook the recording at the national level in 1890. Most of these data have been organized and summarized (principally by Dr. Pablo Herrero-Jimenez) as a ratio, (number of deaths by specific cause)/(number of persons alive), for each age interval, t, for ten year birth cohorts, h, (1800-1809, 1810-1819,....) over the period 1890-1997 by graduate and undergraduate students in epidemiology and publicly posted at http://epidemiology.mit.edu. Up to this time this age and birth cohort specific ratio has been used as the best estimate of the age- and cohort-specific probability of dying of a specific cause. The ratio is generally referred to as the "mortality rate" for a specific cause of death.

For instance, the observed mortality rate, (OBS(h,t), (recorded 1890-1997) for cancers of the lower gastrointestinal tract, mainly colorectal cancer, in European American males (EAM) is shown below as a function of age and birth decade.

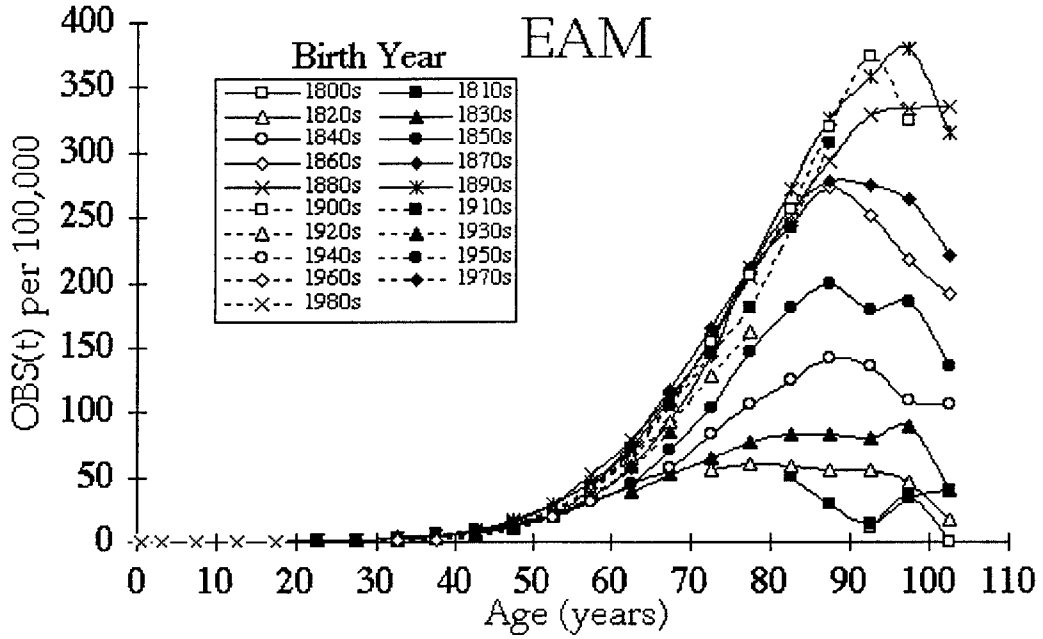


Figure 1: Observed mortality rates from colon cancer for European American Males

This form of presentation shows that the age specific death rates from colorectal cancer have increased dramatically in cohorts beginning with the 1810-19 decade reaching a maximum for middle aged males by the cohort of 1810-19 a hundred years later. Of particular interest to cancer theoreticians is that the data for all cohorts show a maximum mortality rate that appears to decline significantly from the maximum rate by the 100-104 age interval.

This decline is the basis for two separate theories about carcinogenesis. The first is that the age-specific maximum and subsequent decline is ascribable to a subpopulation that inherits an essential genetic factor required for cancer to appear in a normal life span and/or a subpopulation exposed to an environmental factor also required. The steady historical rise in age-specific mortality rates is interpreted as *prima facie* evidence that environmental factors working against a constant genetic background must have been responsible for the rising colon cancer rates in the cohorts of the 19th century. The second

is that the subpopulation at risk is limited to those persons in which a series of somatic changes have taken place in the juvenile maintenance stem cells of the organ in question before those juvenile stem cells matured into adult maintenance stem cells. This second hypothesis does not exclude the elements of the first hypothesis but opens up the possibility that rates of necessary oncogenetic changes in fetuses and juveniles are the sole determinants of lifetime risk.

Three academic collaborators, Professor Stephan Morgenthaler, Chair of Applied Mathematics, ETH, Lausanne, Switzerland, Professor Kari Hemminki, Professor of Epidemiology, Federal Cancer Research Center, University of Heidelberg, Germany and Professor William Thilly now of the Biological Engineering Division at MIT have been working together to discover what forms of mathematical analysis might provide deeper insights into the process of carcinogenesis and how it is effected by inherited and environmental factors. This collaborative effort is the only known group in the world analyzing data in the form presented in Figure 1. All other groups have restricted their analyses to ages less than 80 and therefore have not addressed the form of the data in terms of a maximum in old age.

The model they employ to create equations describing the cascade of events in an individual that leads to cancer is based on the observations by pathologists that colorectal cancer specifically and cancers in general may be described as distinct transition from normal epithelial layers to slow growing preneoplastic colonies to rapidly growing neoplastic colonies from which metastases and sequelae leading to death arise. The widely used "two-stage cancer model" introduced in its basic form by Armitage and Doll in 1957 [1] divides carcinogenesis into two rate limiting stages, initiation and promotion.

Initiation refers to the transformation of some but not necessarily all normal cells with zero adult net growth rate into a slowly growing preneoplastic or precancerous colony. Specifically, normal cells are "initiated" by one or more rare events, usually expected to be genetic events, which allows them to give rise to preneoplastic colonies. In the model, this process occurs in an unknown number of cells at risk of initiation that does not exceed N_{max}. Moreover, in order for initiation to occur, n independent initiation events occurring at a rate of R_i, R_j, ..., R_n events per cell year are postulated. For colon cancer, the field of oncogenetics has shown that n is equal to at least 2.

We have defined a variable "X", which is the probability that an individual in a population is initiated by age a_{max}. This probability is governed by the following equation [5]:

$$X(a_{max}) = (C_{init} / N_{max}) * ((\alpha-\beta)/\alpha) * \int_0^{a_{max}} a * N_a * da$$

The probability "X" is defined in terms of the parameter a_{max}. This parameter accounts for the theory that cells in an individual may only be initiated to a preneoplastic colony through a maximum age. In the case that we hypothesize that cells may only be initiated until maturity, a_{max} is about 17 for males and 15 for females. Also in the equation for "X", the parameter N_a defines the number of cells at risk at a given age "a" [9]. This parameter allows for the hypothesis that only stem cells are at risk, or that all cells may be able to be initiated. Finally, $(\alpha-\beta)/\alpha$ is the growth rate of a preneoplastic colony, discussed further below, and C_{init} is the constant of initiation, defined by the following equation [5]:

$$Cinit = 2R_iR_jN_{max}$$

Promotion is the transformation at which the preneoplastic colony – expanding at a slow growth rate– gives rise to a single "promoted" cell which is the sole precursor of the neoplastic colony that grows and evolves rapidly into a lethal tumor with or without metastases, In order for this stage to occur, m independent promotion events are postulated, each of which occurs at a rate of r_A, r_B, ..., r_M events per cell division. No genetic mutations have yet been found experimentally for promotion in humans or animals, so one must consider the case where m = 0. Furthermore, current research has yet to discover the value of m, or even any sort of bounding of m. We have chosen the value for m when modeling the promotion stage, m=0,1,2,3,... to permit researchers to understand the effect on the value of "m" on the rates of putative promotion mutations and to consider the formal possibility that m = 0. For most demonstration calculations we have been using m=1. The probability that promotion will occur for a neoplastic colony at a given age "t" after initiation at age "a", is given by the following equation [5]:

$$d\,(1 - \exp\{-r_A\,[\alpha/(\alpha-\beta)]^2 \exp \ln 2\,(\alpha-\beta)(t-a)\})/d(t-a) \text{ (for } m = 1)$$

From an understanding of the model of the process by which an individual may get cancer, we can now model the observed cancer mortality of a particular population. This model, called OBS(h,t), is given as the following equation.

$$OBS(h,t) = F*(1-SUR(h,t))(REP(h,t)(1-TOT(h,t))(1 - \exp\{-V_{OBS}(h,t)\}) / (F + (1-F) * \exp\{(-1/f) * \int((1-SUR(h,t))V_{OBS}(h,t))dt\})$$

The terms SUR(h,t), REP(h,t), and TOT(h,t) are defined by Pablo Herrero-Jimenez for his ph.D thesis work. ([9]) The term SUR(h,t) represents the relative survival rate of

those individuals diagnosed with the given cancer. The term REP(h,t) is the reporting error that may occur in counting the deaths to record the mortality rate for a specific cancer. The term TOT(h,t) is defined as the "sum of the rates of all forms of death" for a birth cohort "h" born in year "t". ([9]) Finally, this observed rate, OBS(h,t), is defined as the number of recorded deaths of people at age t and of a particular birth cohort from a specific cancer divided by the total number of people of a birth cohort alive at age "t".

The variable "$V_{OBS}(h,t)$" represents the expected number of lethal events here defined as promotional events per at-risk individual during year "t". The fraction of the population at risk for the specific type of cancer is denoted as "F". The fraction of this subpopulation that dies from this form of cancer and not of a disease sharing the same risk factors as F is defined as "f". Thus, "f" accounts for deaths resulting from competitive forms of mortality as opposed to independent forms of mortality such as automobile accidents etc.

$$V_{OBS}(h,t) = n(\alpha-\beta)/\alpha \prod R_i \sum a N_a * d(1-\exp\{-r_A[\alpha/(\alpha-\beta)]^2 \exp \ln 2 (\alpha-\beta)(t-a)\})/d(t-a)$$

for the product defined for I = 1,2,3..., the summation is defined for a = 1,2,.... a_{max} and m=1.

Unfortunately models in this form were not useful to cancer researchers whether clinical or molecular in approach. Professor Morgenthaler created a Fortran program running on Linux but this wasn't usable by researchers who were mostly Mac or PC based with few programming skills of their own. To overcome this disciplinary barrier between multi-parametric modeling and cancer research Professor Thilly began working in Spring Semester 2001 with a group of 6.872 students to transport Morgenthaler's

Fortran program into Java for which free API is available at http://java.sun.com. Initial trials of the transported program led to recognition of shortcoming of the basic model and opened up the possibility of graduate theses in this area of computational modeling.

David Hensle improved the basic Java program to the point that it could be used to model the two basic hypotheses attempting to account for a cancer rate maximum. This effort represented the primary contribution of his EECS Master of Engineering thesis in 2003. His program dubbed "CancerFit" serves as the starting point for the amendments and improvements proposed for this master of engineering thesis. CancerFit allowed a researcher to load a data specifying the recorded number of deaths for each age interval for a particular birth cohort along with the age-specific numbers of surviving persons in that cohort. These data together defined the approximate function OBS(h,t). He also incorporated the growth functions for children, male and female separately, from birth through maturity and created the means for researchers to include an estimate of the juvenile growth rate of the specific cell population modeled, e.g. epithelium of the colon. The researcher could then specify trial values of n and m deaths over ages for a population, include an estimate of the probability SUR(h,t) of surviving a particular form of death, and specific broad plausible ranges for the parameters that were included in the model of OBS(h,t) including F, f, R_i... r_A, and the growth rate of the preneoplastic colony, alpha-beta. With these ranges and specifications the application found a group of best-fit solutions defined by those solutions yielding minima for the residual sums of squares. One could then plot any of these best fit solutions against the observed data to get some intuition on how the model was working. This application was revolutionary in that nothing like it had ever been created. It was almost immediately discovered for

several cancers that the solutions very narrowly defined the best estimates of preneoplastic growth rates and that these growth rates were remarkably similar to independently determined growth rates of epithelium in the colon, esophagus and lung. This observation in turn led to the hypothesis that initiation might represent blocking juvenile stem cells from maturing into adult stem cells.

The motivation of the application was not, however, only for Professor Thilly and his colleagues to test their cancer theories, but also to allow cancer researchers around the world to explore and test their ideas. While CancerFit was an excellent initial contribution to the field of cancer research, there were and are many extensions and enhancements to both the application and the cancer model still to be completed. Inevitably in any novel undertaking there are mistakes that have been uncovered. The motivation of this thesis is to both enhance the original CancerFit application with a variety of extensions as well as to work closely with Professor Thilly to analyze, amend and improve the current cancer mortality model through the application CancerFit.

2 Ratio of Age-Specific Cancer Rates

2.1 Motivation

In the future work section of his thesis, David Hensle expresses that CancerFit could be modified to include the capability of comparing multiple data sets. Professor Thilly reiterated that interest, as it would allow for more complex and interesting analysis. Examples of such data sets that could be compared were the values of OBS(h,t) from

males and females of the cohort "h", of different cohorts, or of unique data sets developed by Professor Hemminki permitting the values of OBS(h,t) to be compared for the general Swedish population and a subpopulation of the first degree relative with any or all particular forms of cancer diagnosis. Such data exists and is discussed in "Multiple primary cancers as clues to environmental and heritable causes of cancer and mechanisms of carcinogenesis" [3] as well as in "Familial and second primary pancreatic cancers: a nationwide epidemiologic study from Sweden" [4].

This added functionality toke the form in two interrelated tasks. The first task was to allow a user to load two data sets, and then view a plot of the ratio of cancer mortality rates for these populations, as a function of age. The second task was to add an application, encapsulated by this larger application, which would allow a user to plot a ratio of the fits to the two data sets on the same graph as the ratio of the observed rates for the data sets. By doing so, a user could then set the value of the fit in the second data set and change the values for the parameters in the model to recalculate the fit for the first data set. This could give the user some intuition as to what the differences were between the two fits, by seeing what changes, specifically what value changes of which parameter, allowed the user to transform the fit ratio into the observed ratio.

The motivation behind such an application may be demonstrated by the following example. Assume we have two data sets- the data for incidence deaths from colon cancer in the general population of Swedish males at each age, and the data for the incidence deaths from colon cancer in the subpopulation of that general population of Swedish males which only includes individuals who have had a first degree relative, a parent or a child, who has cancer. We can first investigate and find the observed mortality rates due

to colon cancer at a specific age for each of these data sets, and plot them against each other accordingly. This gives a user some intuition as to the differences of the two data sets, relative to each other. It also allows a user to understand the increase in colon cancer risk that the subpopulation of first degree relatives has over the general population. The importance and implications of such a difference in risk are discussed below.

Using the CancerFit application, we can use the model to fit the observed mortality rate of the data sets. We'll call these fits OBS_1 and OBS_2. OBS_1 is the model fit for the colon cancer mortality rates for the subpopulation of individuals who have a first degree relative with cancer. (This is also referred to as "OBS primary".) OBS_2 is the model fit for the colon cancer mortality rates for the general population. (This is more typically referred to as "OBS naught'.) By taking the ratio of "OBS primary" over "OBS naught", we find the ratio of the fits to the separate data sets. We can then compare this ratio of fits to the ratio of observed data to find first some intuition about how the model is fitting the observed data. More importantly, we can begin to change the value of one of the fits by changing a value of one (or more) of the models variables for one fit while keeping each value for the models variables constant for the other fit. We then are altering the ratio of fits, and we can observe how it compares to the observed ratio as the values of different variables change.

In this example, we chose the same values for each of the variables of the "OBS primary" and "OBS naught" fits and plotted the ratio of "OBS primary" over "OBS naught" first. This ratio was simply one, as both fits were found from the model for OBS using the same values for each variable. However, when we begin changing the values of the variables of the OBS model, for example "F" or "f", we see changes in the ratio of

fits. If we plot this ratio of fits on the same plot as the ratio of observed data, and we allow a user to view how the ratio changes and becomes more like or dislike the observed ratio as the user changes the variables of the model, the user may begin to understand what the model suggests may causes the difference between the mortality rates of colon cancer for the subpopulation of individuals who have first degree relatives with colon cancer, and the general population of individuals with colon cancer. More explicitly said, a user can begin to understand what the model interprets as the reason for this additional risk, called the "familial risk", or "relative risk", for a given cancer that an individual has over another individual in the general population because the first individual has a family member with that cancer.

2.2 Loading Data Sets and Plotting Ratio Data with Estimation of Dispersion

The first task to implement this extended application within the original CancerFit application was to add functionality so that a user could plot the ratio of the observed mortality rates of two sets of data. We added a "Plot Ratio" option to the user interface to let the user plot the ratio after loading the second data set into the application. When plotting the ratio, we calculated the ratio of each population's observed mortality rates at each given age. It is important to mention that the data is typically organized so that incidences deaths over a five year range are grouped into the same age bracket. For example, the number of incidence deaths recorded for age 12.5 in a data set is actually all incidence deaths of individuals from age 10-15. Furthermore, because different data sets start with incidence deaths for different age brackets, we had to account for possible offsets between data sets. For example, some data start with data for individuals at age

2.5, while some data sets start with data at age 12.5. Thus, we need to check for all possible cases here and calculate the ratio accordingly.

Once we have handled all of the possible discrepancies with the organization of the data, the ratio is plotted in green on a pop-up graph. Also calculated is the standard deviation of the ratio in order to plot one standard deviation above and one standard deviation below the ratio on the graph. (The standard deviation points are plotted in blue.) Plotting these standard deviations is important, as it shows the user where the ratio points are well-defined, and thus reliable, and where it is not. In most cases, the standard deviation of the data for two tails of the ratio graph, the youngest ages and oldest ages are significantly larger than the standard deviation of the ratio in the middle ages. This occurs as there are not very many cases of cancer mortality for younger and older individuals of a population. Limited data tends to lead to large standard deviations, demonstrating that the data, in this case the ratio at such ages, is imprecise.

Below is a snapshot of a plotted ratio of observed mortality rates for colorectal cancer between the Swedish subpopulation of individuals with first degree relatives who have that cancer and the general Swedish population. This snapshot was taken as a .jpeg picture file with a standard screenshot application, and then transported into this document.

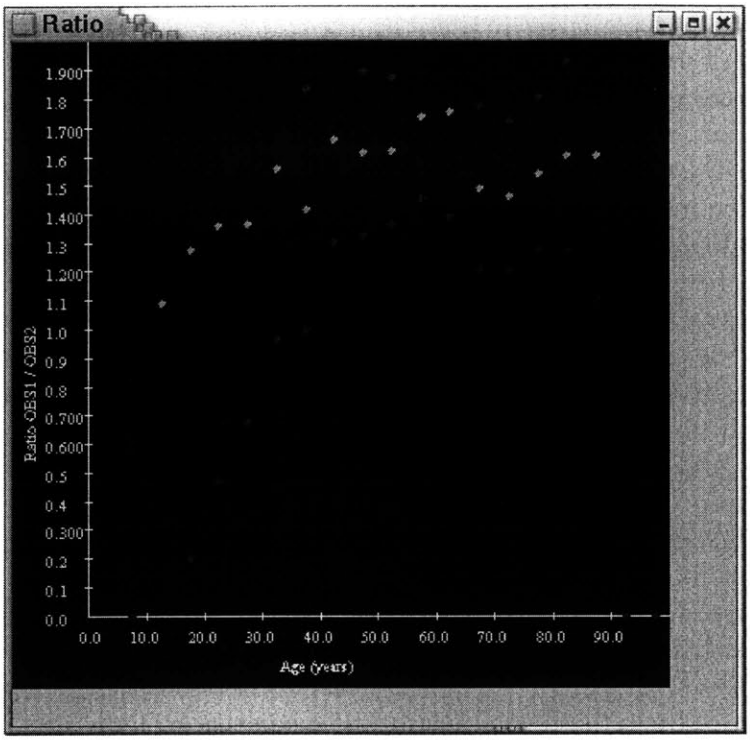
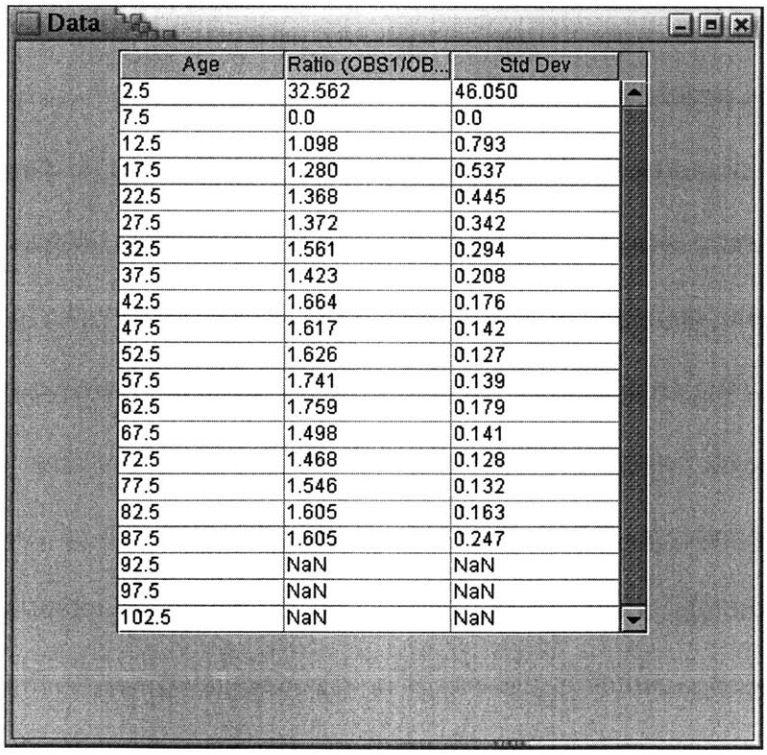


Figure 2: Ratio plot example. The middle (in green) line in the graph is the actually ratio of OBS_1/OBS_2, while the upper and lower lines (in blue) show one standard deviation above and below the ratio. These data for colorectal cancers show that the risk of colorectal cancer in first degree relatives of colorectal cancer patients is ~ 1.6 fold higher than in the general population after age 50.

The plot of this ratio immediately gives the user intuition about the differences in the data sets. First off, a user can determine where the ratio is well-defined because of the plots of the standard deviations. It is from this more reliable part of the ratio that we can then begin to analyze. We first realize that this ratio is the ratio of the percentage of each population at risk for colon cancer. This can be related to the model as the ratio of "F" values for each population. This ratio of "F" values is greater than 1, at roughly 1.5. Since we are plotting the "F" value for the population of first degree relatives over the "F" value for the general population, the user is given immediately shown the "familial risk" factor known to exist in relatives of individuals with cancer. From this discovery, a user can rationalize that this "F" ratio may imply an upper bound for "F", or the percentage at

risk, at around 1/1.5 or .667 for the general population. We derive this analysis from the fact that since there is an upper bound on "F" for any population at 1.0 (since it is a percentage), and the mortality rates for the general population seem to be about one-third less than that of the population of first degree relatives, then the "F" for the general population could be, at most, 1-1/3 or 2/3's. While these numbers are only rough estimates based on the above plot, they give some insight as to the intuition the above plot may give to users of the application.

Along with the ability to plot the ratio and one standard deviation above and below the ratio as a function of age, we also allow a user to view the ratio and the standard deviation of the ratio at each age of the data sets. To view this information, the user may select the "Show Data" button as an option after loading the second data set.



Age	Ratio (OBS1/OB...	Std Dev
2.5	32.562	46.050
7.5	0.0	0.0
12.5	1.098	0.793
17.5	1.280	0.537
22.5	1.368	0.445
27.5	1.372	0.342
32.5	1.561	0.294
37.5	1.423	0.208
42.5	1.664	0.176
47.5	1.617	0.142
52.5	1.626	0.127
57.5	1.741	0.139
62.5	1.759	0.179
67.5	1.498	0.141
72.5	1.468	0.128
77.5	1.546	0.132
82.5	1.605	0.163
87.5	1.605	0.247
92.5	NaN	NaN
97.5	NaN	NaN
102.5	NaN	NaN

Figure 3: Ratio data show example. This displays of ratio of OBS_1/OBS_2 and one standard deviation of that ratio for each age where the data for OBS_1 and OBS_2 exists.

With this option, a user may now analyze the relative difference in the data sets via a plot, or by looking at the actual ratios and standard deviations.

2.3 Graph Ratio Fit

Since a user was now able to plot the ratio of the observed mortality rates between the two data sets, the next step was to add functionality allowing a user to plot the ratio for the fits of these data sets on the same plot as the observed ratio. To include this functionality, we created a new window view for the application called "Graph Ratio Fit", where the new functionality could be encapsulated. The application that we were working from already had the ability to fit the mortality rates for an observed data set with the model discussed above. To do so, the user first inputs specific information for the type of cancer the data set describes. Then, the user may input ranges for values of each of the variables of the model over which the model will try to fit the observed mortality rates. For example, the user can specify that "f" is 1.0. When fitting data sets of colorectal cancer, this is a reasonable assumption. By setting "f" to equal 1.0, we are indicating that there are no competitive cancers such that no individual has died from the competing cancer but could have also died from colon cancer.

After the user has set the ranges and values of all of the variables, he may then use the model to fit the data. The application displays the variable values for the best fits to the observed data in the "Best Fits" pane. (As mentioned above, "best" here is determined by the residual sum of squares method.) Now, the user may select one of these fits to plot the ratio of fits for the first data set to the second data set. By selecting a fit, the user is selecting the initial values that will be used for the variables in each of the fits of the

ratio. The user may then select the "Graph Ratio Fit" pane to view the values of each variables for the fit selected, and now the user has the option to plot the ratio of the observed mortality rates for the data sets, and then plot the ratio of the fits of the data sets on the same graph. Plotting the ratio of the observed mortality rates uses the same functionality that we first added to the application. However, for this part of the application we allow for plotting the ratio of fits to the observed mortality rates in order to discover more information about the ratio of the observed mortality data.

Below is a snapshot of the ratio of fits, which is plotted in red, against the ratio (again displayed with one standard deviation above and below the ratio) of the observed mortality rates between two data sets. (The snapshot of this graph also includes the "Graph Ratio Fit" pane, to demonstrate how that window is designed.)

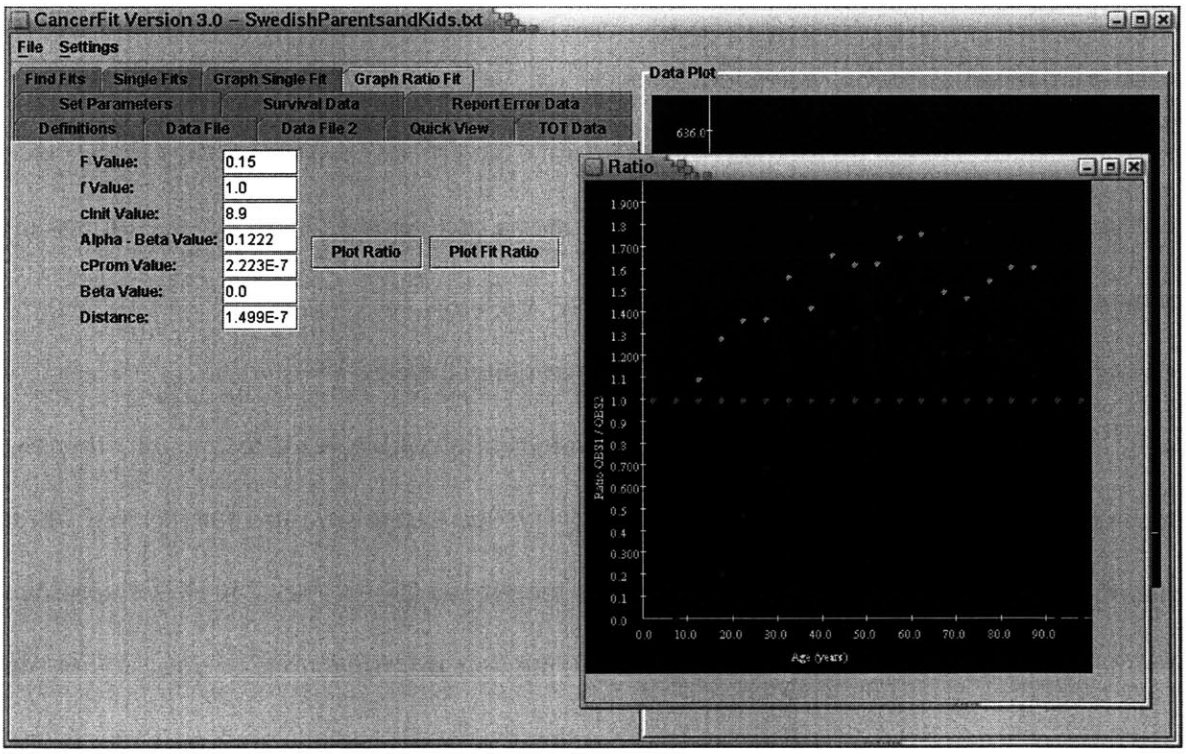


Figure 4: Here is the Graph Ratio Fit window, with the OBS_1/OBS_2 ratio plotted as before and the ratio of fits (red line) set to 1 initially.

As one may notice, the ratio displayed above is 1.0 for all ages. This occurs the first time the ratio of fits is plotted or if none of the values for the variables of the model (F, f, etc) have been altered in the "Graph Ratio Fit" window. The reasoning for this lies in how we implemented plotting the ratio of fits. Once a user selects a fit from the "Best Fits" pane and they become the initial values for each variable of the "Graph Ratio Fit" pane, those values are the initial values used to calculate the model's fits for both data sets. Since the model will give the same fit when using the same values for each variable, this ratio is initially 1.0. However, when changing a value of any of the variables, the ratio of fits and correspondingly the plot will change. This is because the first time the ratio of fits is plotted, both fits in the ratio are calculated the same but only the fit in the denominator of the ratio is set. Now, each time the user changes the values for the model and re-plots the ratio, only the fit in the numerator is recalculated. The motivation for such an application may be realized in the following example.

As before, we will use the observed mortality rates of colon cancer for the Swedish subpopulation of individuals having first degree relatives with colon cancer as our data set 1, and the general Swedish population as our data set 2. When we select a fit, and plot the ratio of the observed mortality rates between data set 1 and data set 2, we find the same results as we did earlier. When we first select the "Plot Ratio Fit" option, it plots a ratio of the fit for data set 1 to the fit for data set 2 in red as shown above. Now, when we change the value of a variable, for example, if we change the variable "F" from its initial setting of 0.15 to 0.23, and select "Plot Ratio Fit" again, the following graph is the result. (Again, the snapshot of this graph also includes the "Graph Ratio Fit" pane.)

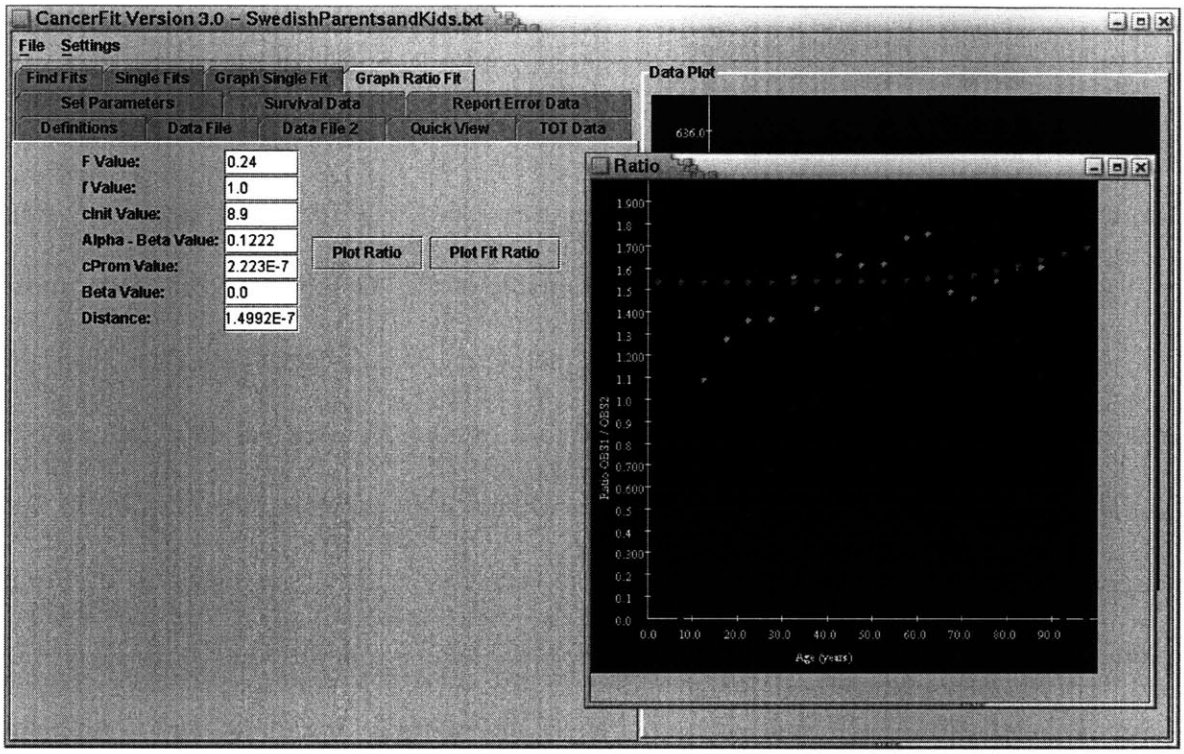


Figure 5: Here is the Graph Fit Window with the ratio of fits plot again, however now the "F" value has been changed from 0.15 to 0.24 to demonstrate possible differences in OBS_1 and OBS_2.

From this graph, we can see that changing the "F" value when calculating the fit for data set 1 shifts the ratio up closer to the ratio of the observed mortality rates. What this is allowing a user to do is to see how the ratio of the fit of data set 1 (the subpopulation of first degree relatives) to the fit of data set 2 (the general population) is changed when the value of a single variable, such as "F", is changed in the model. As mentioned before, this gives the user intuition about what the model suggests may be the differences between two data sets. In this example, by re-plotting the ratio of fits each time we change the value of "F", we are trying to discover if the variable "F" is the cause of the difference between the mortality rates of the general population and those of the subpopulation of first degree relatives. Specifically, we are attempting to determine if the difference in "F"

(the percentage of a given population at risk) explains why individuals with first degree relatives with colon cancer have this higher risk for colon cancer.

While in this example we only changed the value of "F" to see how the ratio of fits changed (and how much more or less it resembled the ratio of the observed data sets), this application allows us to change the values for any of the variables, or combination of variables, to begin to understand what factors the model implies are responsible for the "familial risk" of cancer.

3 Modifications to the CancerFit Program

The original CancerFit application developed by S. Morgenthaler, D. Hensle and others used the observed mortality rate of cancer, OBS(h,t), to fit the parameters. Pablo Herrero-Jimenez [6, 7, 8, 9], developed a more accurate version of the model of cancer occurrence probability by accounting for the historically improving therapies that increased the chance of survival given a particular cancer diagnosis. His approach was to painstakingly gather data on historical age-specific survival rates defining the parameter SUR(h,t), which he used to define the term OBS*(h,t), an estimate of mortality in a birth year cohort "h" in the abstract condition of zero chance of survival, i.e. an estimate of cancer incidence. The equation for OBS*(h,t) is shown below:

$$OBS^*(h,t) = OBS(h,t) / [REP(h,t)*(1-SUR(h,t))]$$

In his paper, Herrero-Jimenez defines R(h,t), now REP(h,t) as the reporting error that may occur in counting the deaths to record the mortality rate for a specific cancer.

(Herrero-Jimenez, 2000) REP(h,t) is the term to correct for the vagueness of mortality records especially in the historical period 1890-1935. It thus makes sense that the REP(h,t) term affects the data sets in earlier years, as diagnostic methods have become more accurate. Herrero-Jimenez defines S(h,t), now SUR(h,t) as the relative survival rate of those individuals who were diagnosed with the given cancer. ([9]) He defined the survival rate as a relative rate to account for individuals who were diagnosed with a specific cancer and died from another cause during the five year period after that diagnosis. Surviving from a given cancer is interpreted as having lived at least five years after being diagnosed.

While the SUR(h,t) and the REP(h,t) data was not used in the original version of CancerFit, Professor Thilly determined that including this data was important in creating a more accurate observed incidence curve. While the OBS(h,t) data is observed mortalities, including SUR(h,t) and REP(h,t) help better approximate the number of individuals who received a given cancer. Fitting this data is clearly more helpful than fitting only the mortality data when trying to discover how often a why individuals receive a given cancer. Below, we discuss the additions of the SUR(h,t) and REP(h,t) data to the CancerFit application.

3.1 Survival Data

In the initial version of CancerFit, a user could enter a survival rate for the loaded population data within the "Set Parameters" section of CancerFit, and the model would use this survival rate for the population at all time "t". While this was a nice simplification, the truth is that SUR(h,t) changes with "t" in present day medical

experience and has changed for many cancers as a function of "h" as improvements in cancer therapies have been discovered and applied. This need and solution is represented in the Ph.D. thesis work of Herrero-Jimenez who dug deep to define the values of SUR(h,t) for colorectal cancer since the beginning of the 20^{th} century. ([9])

Reexamining this simplification led Professor Thilly to believe that if CancerFit was going to be generally useful for cancers such as colorectal, bladder, breast and several others in which SUR(h,t) has fortunately increased in the last century it would be necessary for a user to be able to discover and then specify the survival rates over age "t" for the population being studied within the CancerFit application. In essence, it is necessary for a user to enter a data file of non-constant survival rates over age "t". We discuss the changes to the CancerFit application that made this possible below.

As seen in Herrero-Jimenez's equation for OBS*(h,t), the SUR(h,t) data is used to divide the observed data, OBS(h,t), by (1- SUR(h,t)). This division allows the model to account for those individuals who survived, as the division results in the mortality curve as if everyone in the cohort who had been diagnosed with a given cancer had died from that cancer. This tactic converts mortality data, OBS(h,t) into an estimate of incidence, OBS*(h,t), i.e. the probability of observing a particular form of cancer at age t in birth year cohort h.

3.1.1 Survival Data in the CancerFit Application

In order to include the survival data in the CancerFit application, we first needed to create a survival data file. The survival data files were created from data that Pablo Herrero-Jimenez discussed and displayed in his paper. ([9]) Next, we had to extend the

functionality of the CancerFit application to be able to load the survival data files for use. An example of those survival data files loaded into the CancerFit application is shown here:

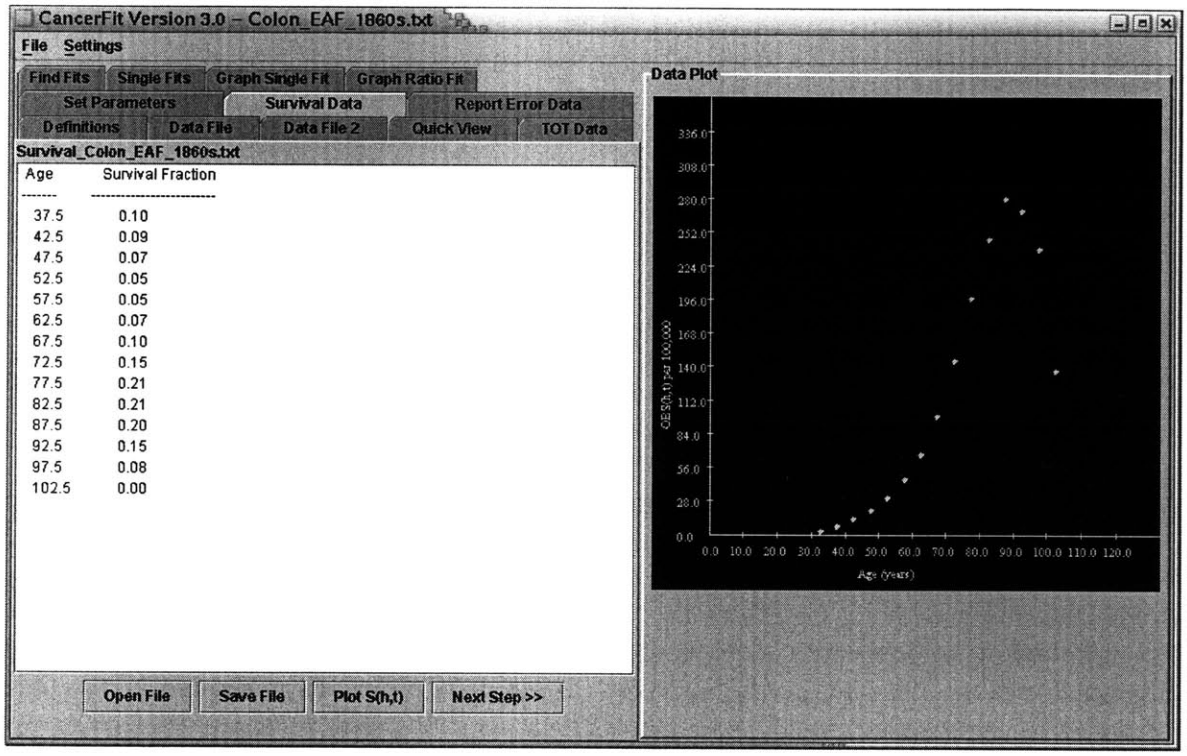


Figure 6: The survival data for European American Females (EAF) born in the 1860's who were diagnosed with colon cancer. Here is an example of the data loaded into the CancerFit program.

The functionality to allow a user to plot the loaded survival data was added as well. An example of such a plot is shown here:

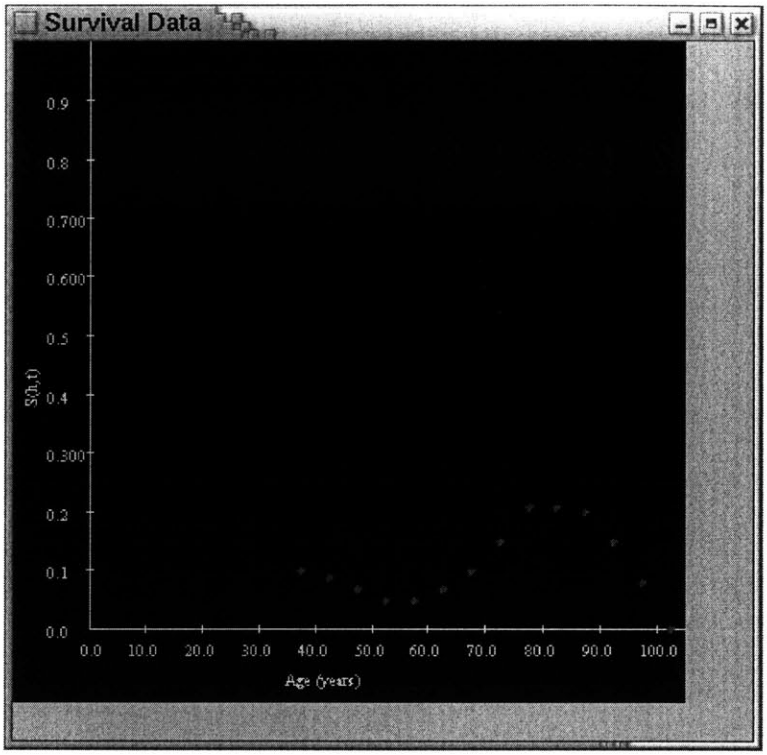


Figure 7: A view of the plot of the survival data shown loaded into the CancerFit program in Figure 6.

As mentioned above, the survival rates for some cancers may be zero, or may be constant. In such cases, it is justified and helpful to allow a user to set the survival rate to a constant. Thus, this functionality was not removed from the original CancerFit application, and is still available as an option to the user, in which case the user does not need to load a data file.

The survival data begins to affect the observed mortality curve in the middle ages of a cohort, and continues to have an affect through the end of the curve, with the greatest magnitude of affect coming from the 77.5 to 92.5. Below is an example of the observed mortality curve without the survival data plotted against the observed mortality curve with the survival data included. The yellow curve in the graph below is the OBS(h,t) after the affect of the SUR(h,t) data, whereas the red curve is the original OBS(h,t).

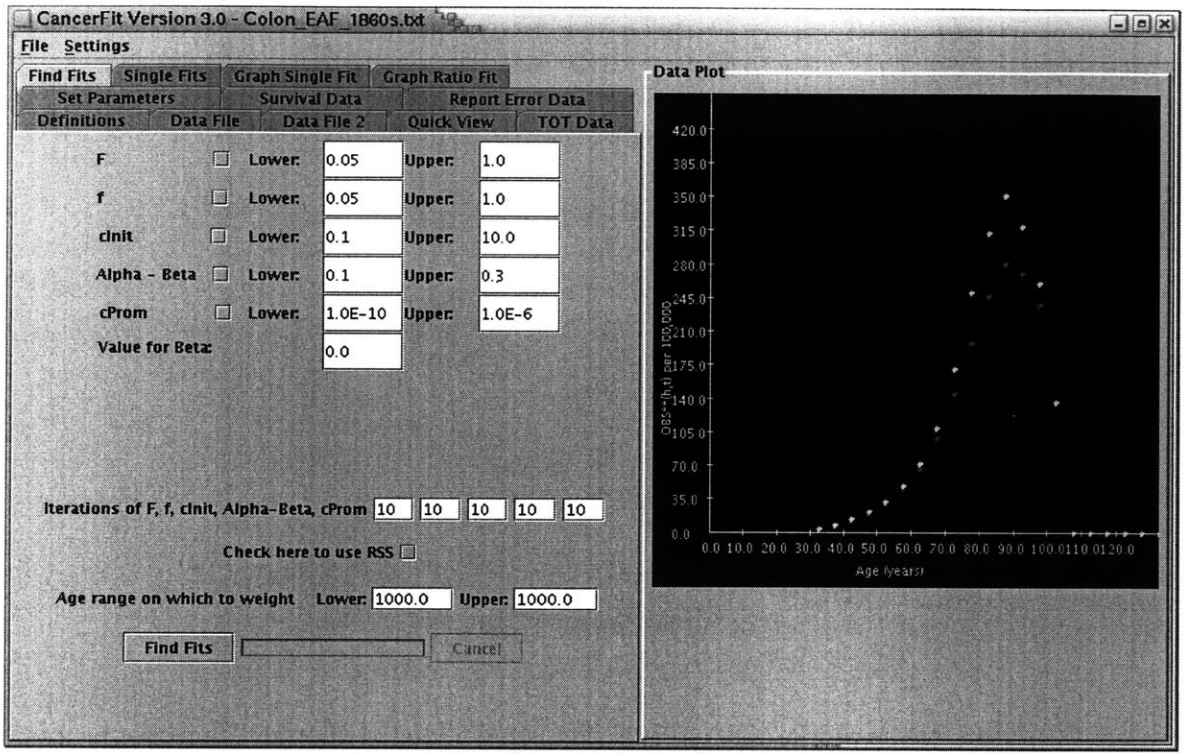


Figure 8: This graph demonstrates the effect of the survival data on the observed mortality data. The upper curve (in yellow) is the mortality data after the division of the (1 – SUR(h,t)) factor, OBS*(h,t),while the lower curve (in red) is the original data set, OBS(h,t).

3.2 Reporting Error Data

As described above, Herrero-Jimenez defined reporting error, REP(h,t), as a term to account for the vagueness or inaccuracies of mortality recording throughout history. The REP(h,t) is more exactly defined by Herrero-Jimenez as the "probability of accurately recording the cause of death". ([9]) The REP(h,t) term appears in Herrero-Jimenez's equation OBS*(h,t) (seen above) along with SUR(h,t). While we have explained how the SUR(h,t) term affects the observed mortality curve, we shall now discuss the affect of the REP(h,t) term and the motivation behind allowing a user to enter such data in the CancerFit application.

Since the REP(h,t) term accounts for the probability of recording the mortality curve correctly, before using the observed mortality curve, OBS(h,t), to fit the parameters, it seems reasonable if not necessary to re-examine the data based on the recording accuracy. As seen in the equation for OBS*(h,t) above, the REP(h,t) factor is incorporated into the model by dividing OBS(h,t) by REP(h,t). By the principles of probability, by multiplying the value we have, in this case OBS(h,t), by the probability that it is accurate, here REP(h,t), we may calculate a more accurate expected number of mortalities for each age interval.

3.2.1 Reporting Error in the CancerFit Application

The steps to include the reporting error data in the CancerFit application were identical to the steps taken to include the survival data. First we created reporting error data files for researchers to use. Again, these data files were created from data repositories that Pablo Herrero-Jimenez made when working on his PHD thesis. ([9]) Then, we extended the functionality of the CancerFit application to be able to load the reporting error data files. An example of those reporting error data files loaded into the CancerFit application is shown here:

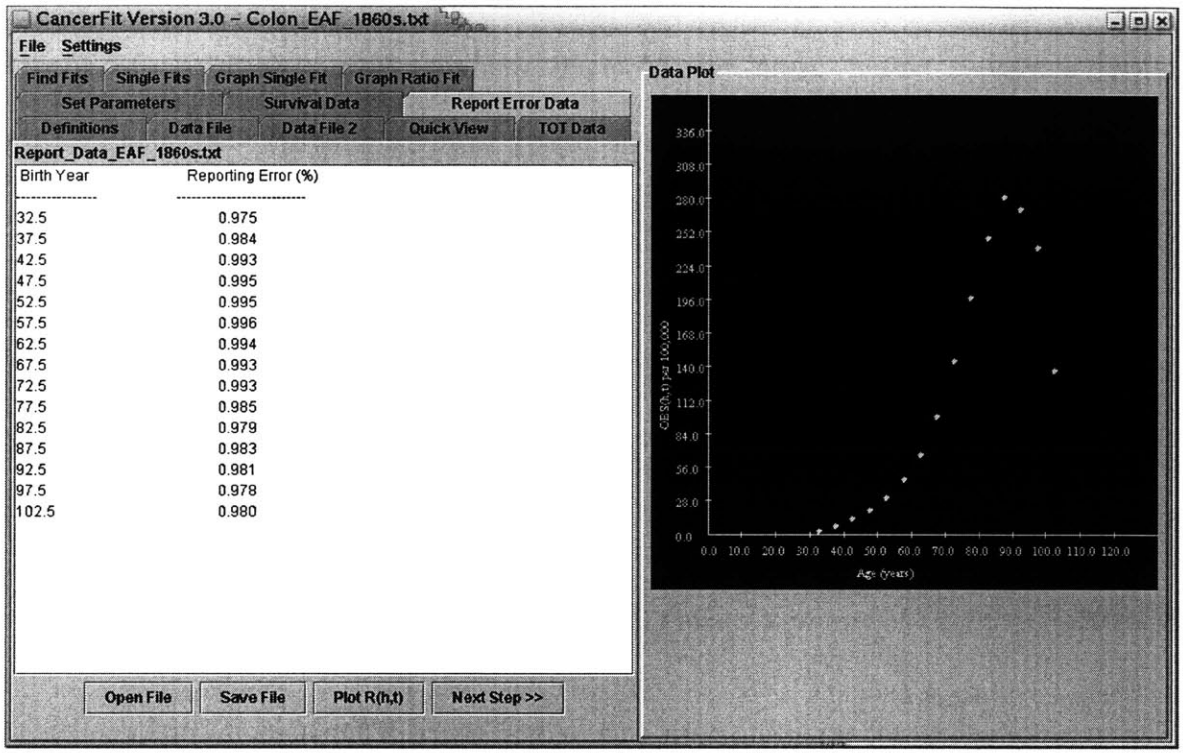


Figure 9: The reporting error data for European American Females (EAF) born in the 1860's who were diagnosed with colon cancer. Here is an example of the data loaded into the CancerFit program.

Again, a user may plot the loaded data file as shown below.

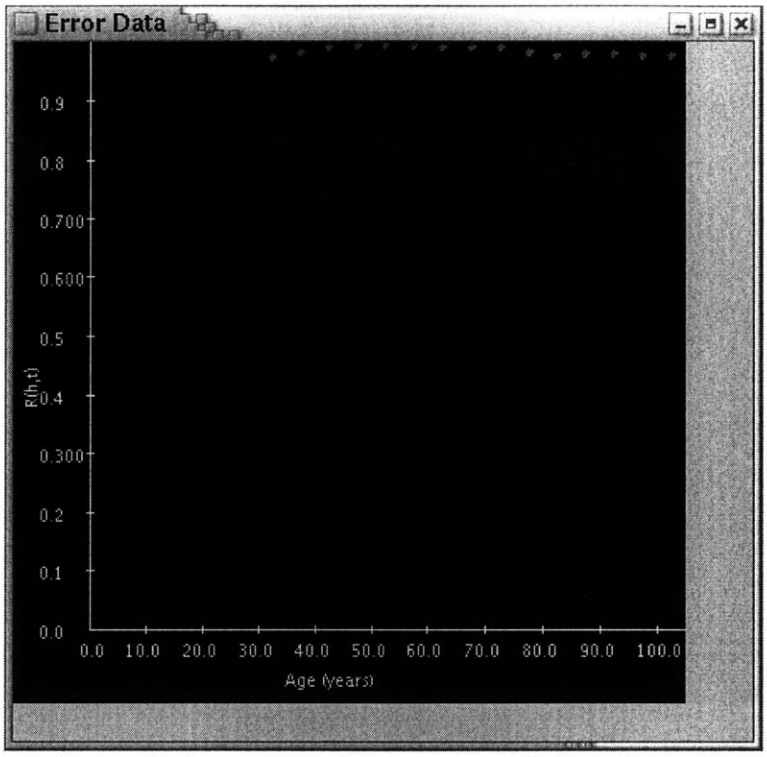


Figure 10: A view of the plot of the reporting error data shown loaded into the CancerFit program in Figure 9.

Typically for newer data sets, REP(h,t) serves only as a minor factor in better approximating the incidences of a given cancer in a population. This is because recording methods have become more accurate as medicinal practices and technology has improved throughout history. However, for older data sets, REP(h,t) may not always be negligible and may still serve to correct the data, especially for individuals at older ages. Here is an example of the observed mortality data of European females born in the 1860's without the affect of the REP(h,t) data versus that same OBS(h,t) with the REP(h,t) factor. The yellow curve below is the OBS(h,t) after the affect of the REP(h,t) data while the red curve is the original OBS(h,t) data.

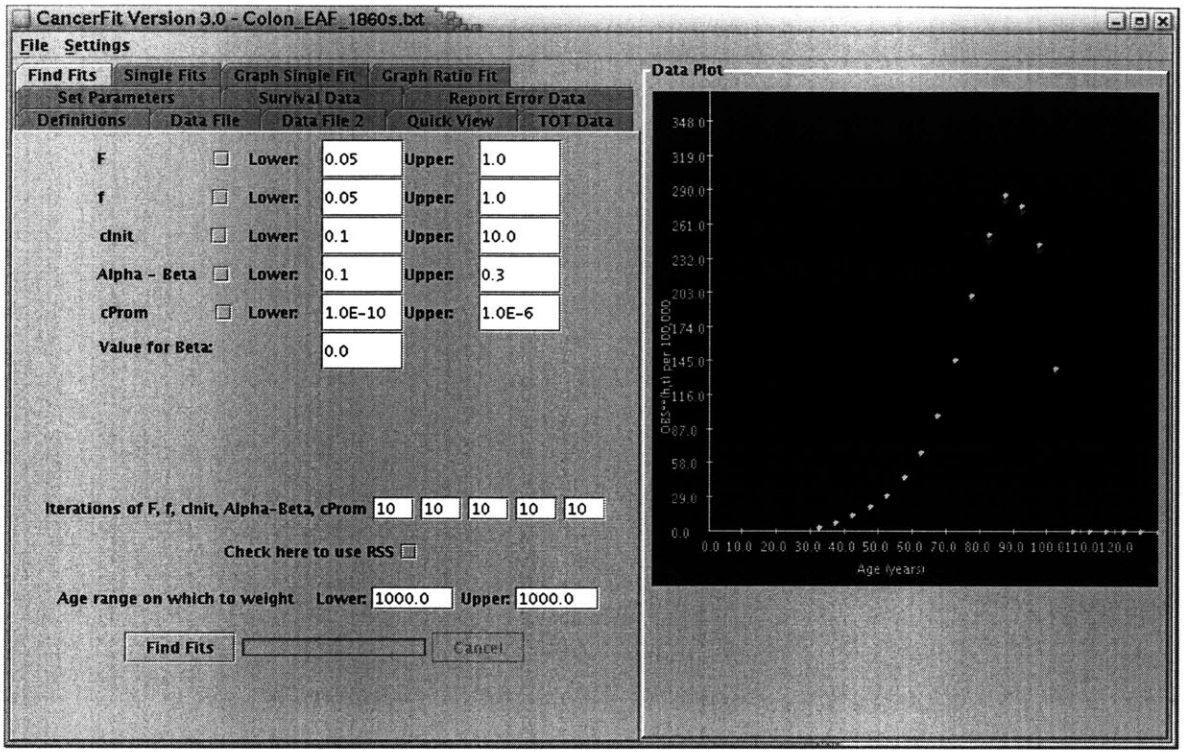


Figure 11: This graph shows the effect of the reporting error data on the observed mortality data. The upper curve (in yellow) is the mortality data after the division of the REP(h,t) factor, while the lower curve (in red) is the original data set, OBS(h,t).

As you can see, the reporting data has little effect on the observed mortality rates except for in the later ages of the cohort. Only for ages 82.5 to 97.5 is the difference noticeable. Indeed, the REP(h,t) data offers a significant correction only within data sets for older individuals who died between the 1890's and 1940's. (9) The lower values for REP(h,t) in these cases may be explained by the difficulty of accurately recording the cause of death for older individuals with naturally more possibilities for causes of death. Other than in these data, the REP(h,t) is almost one, accounting for improved accuracy in the reporting data as medical data and technology improved.

While the reporting data does not typically change the observed mortality data as much as the survival data does, we feel it is a worthy addition to the CancerFit application to give the user the option to enter REP(h,t) data. It is, of course, a necessary

correction needed by researchers exploring environmental factors that changed for the birth cohorts born after 1810-19. Regardless of the magnitude of difference, the REP(h,t) data gives a more exact observed mortality curve to fit, especially when investigating the data mentioned above.

4 Changes to the Model of Age-Specific Cancer Rates: Improved Estimation of Age-Specific Cancer Probability

4.1 Accounting for Competing Forms of Death within Each Age Interval Using Age-Specific Mortality Rates TOT(h,t)

Within the current cancer model, we consider the mortality rates of a specific cancer in a five year period. When a user enters a data set for the population and mortalities or incidences within that population, the data is grouped into five year periods, such that there is a population and mortality/incidence number every five years. To condense that mortality data into five years, the mortality data for the entire period is averaged and set as the mortality rate for the midpoint of the period. That is, all the deaths in an interval are divided by the sum of the number persons alive in the five year interval to yield a first estimate of mortality rate. The general model introduced by Herrero-Jimenez et al. [6, 7, 9] accounted for deaths in the birth cohort occurring up to the year for which the calculation applies. It did, not, however, adequately account for the increasing fraction of deaths in a single year interval in extreme old age, e.g. the age 102 years, the mid year in the interval 100-104 which has been as high as 0.3 in the data set records from 1890 to the present day. Essentially, what the data represents and what the user is testing is the

averaged cancer mortality data over five year intervals. As a practical matter it is the estimate of the mortality or incidence rate for the mid year in the five year interval.

This was both a simplification in the way the data sets were created, as well as in the number of calculations to find the best fit curve. In many ways it is a simplification that makes sense, as it condenses the mortality data and allows a user to get at the heart of the dynamics of the mortality data over time, as opposed to getting bogged down in the changes over shorter time periods.

During the course of this thesis, the averaging of mortality rates was recognized as creating this unintended bias. This arose for the age intervals above age 60, generally, in which the fraction of the population dying of all causes was not negligible. Specifically, the model should be more specific in accounting for the mortality data within the mid year interval. In its original form the number of persons estimated to be alive at the beginning of the mid year interval wert modeled as being at risk for incidence or death in that mid year interval. The possibility that a person would have died of some other cause within that single midyear interval had not been explicitly accounted. Examining the mortality data within the midyear interval is important especially in populations as they get older, because the probability of death reaches as high as 0.3 for the midyear interval for the age interval 100-104 and is significantly greater than 0.0 for all age intervals beyond 60-64.

To estimate the mortality rate in the midyear of each five year interval, we used the term TOT(h,t) for each population. TOT(h,t) is a term defined by Herrero-Jimenez as the "sum of the rates of all forms of death" for a birth cohort “h” born in year “t”. ([9]) The TOT(h,t) value was added into the equation so that the equation could be changed to

not just consider cancer among persons who survive to the end of a given interval, but to account for the fact that among persons who died in the interval would be persons who would have died of the observed disease in that same interval if they had not died of something else.

In order to correctly use the TOT(h,t) values to alter the model, we first considered the OBS*(h,t) formula from Herrero-Jimenez et al. (2000), as shown below:

$$OBS^*(h,t) = OBS(h,t) / [REP(h,t)^*(1 - SUR(h,t))]$$

Given the discussion of how the TOT(h,t) should affect the model, a new equation, OBS**(h,t), was formulated as shown below:

$$OBS^{**}(h,t) = OBS^*(h,t) / (1 - TOT(h,t))$$

Or, in terms of the observed data, OBS(h,t):

$$OBS^{**}(h,t) = OBS(h,t) / [REP(h,t) * (1 - SUR(h,t)) * (1 - TOT(h,t))]$$

By dividing the OBS(h,t) by the (1 – TOT(h,t)) term, we are able to calculate the expected number of mortalities for a given cancer that would have occurred in an interval if no one in that interval had died from a cause other than the given cancer. By dividing the OBS*(h,t) by the (1 – TOT(h,t)) term, we are able to calculate the expected number of incidences for a given cancer that would have occurred in an interval because under-reporting, REP(h,t), survival for that age and cohort, SUR(h,t), and now competing forms of death in the exact age interval year, TOT(h,t) are now accounted.

4.1.1 TOT Data in the CancerFit Program

To use the TOT(h,t) data in any study, one must first create TOT(h,t) data files that can be loaded into the CancerFit program. The TOT(h,t) data, compiled and organized by Pablo Herrero-Jimenez, is available on the web at http://epidemiology.mit.edu. ([8]) However, the data is in spreadsheet format. As such, to load this data into the CancerFit program, it must first be transported into a text file manually. These steps were taken as part of this thesis work to allow cancer researchers to begin using the TOT(h,t) data. Concurrently, the functionality of CancerFit was extended so that a TOT(h,t) data file could be loaded. Below is an example of a TOT(h,t) data file loaded into the CancerFit application.

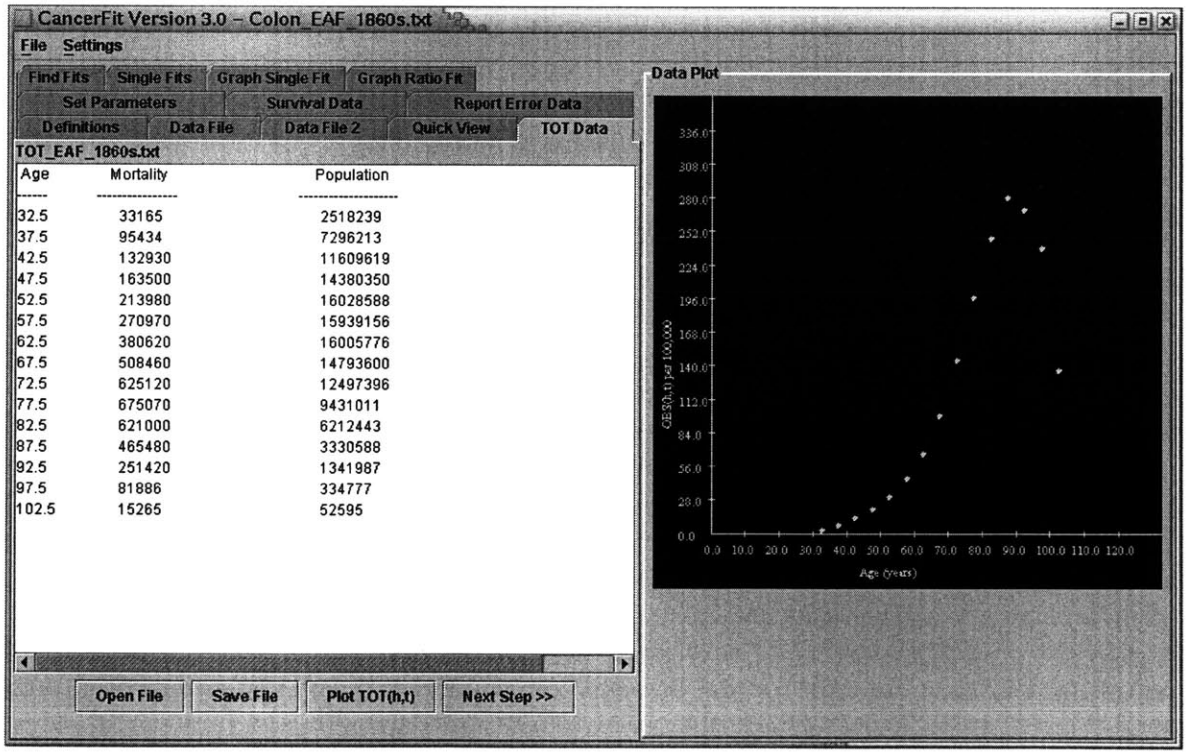


Figure 12: The TOT(h,t) data for European American Females born in the 1860's, shown loaded into the CancerFit program.

Furthermore, we allow a user to plot the TOT(h,t) data that one loads into CancerFit. An example of such a plot is shown below.

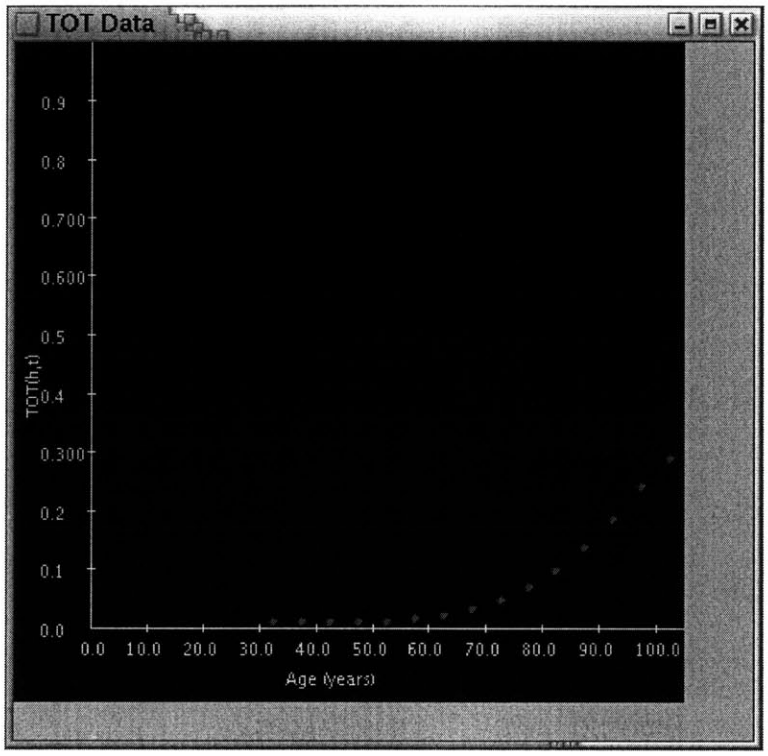


Figure 13: A view of the plot of the TOT(h,t) data loaded into the CancerFit program in Figure 12.

The addition of the TOT(h,t) factor is significant as we can display that this correction raises the mortality rates calculated for extreme old age relative to those at middle age. This correction is about 1.5 times for the most recent data for cancer deaths in the age interval 100-104 while less than 1.1 times for the age interval 55-59, for instance.

An example of the changes to OBS(h,t) that including the TOT(h,t) data alone (not including the survival or reporting error data files) is shown below. As with the SUR(h,t) and REP(h,t) examples, the yellow curve below is the OBS(h,t) curve after the affects of the TOT(h,t) data, as the red curve is the original OBS(h,t) data for comparison.

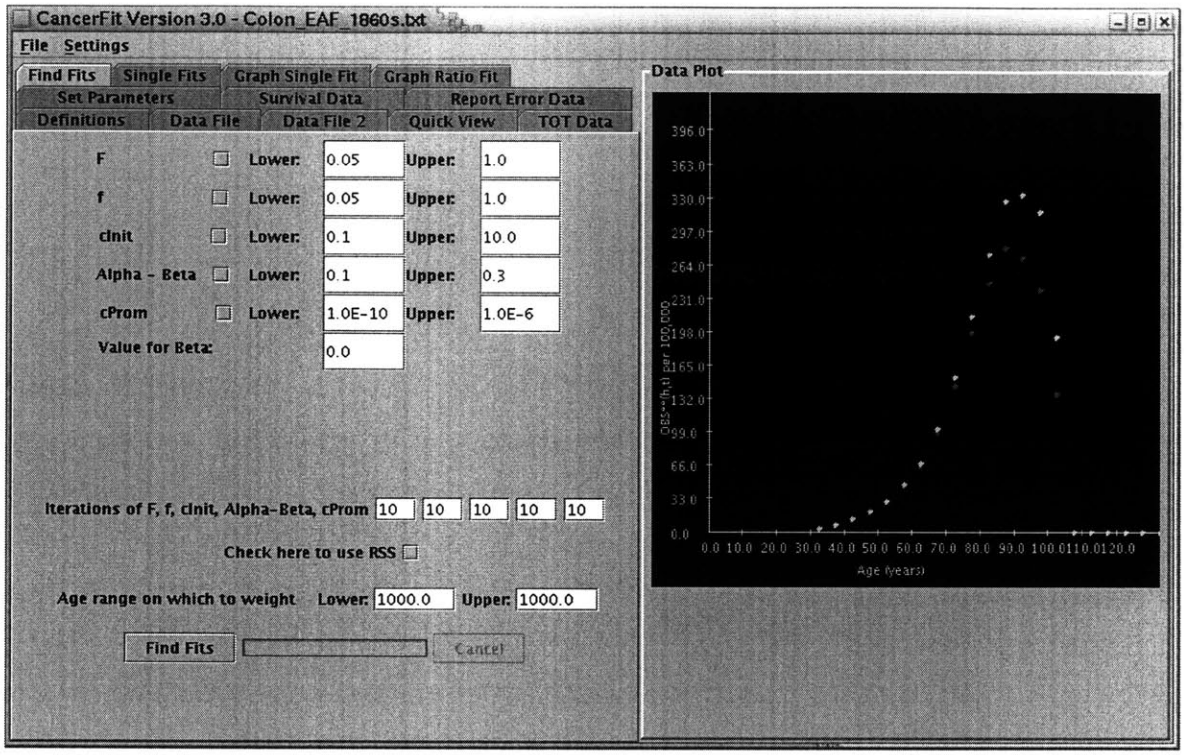


Figure 14: The effect of the TOT(h,t) data on the original observed mortality data. The upper curve (in yellow) is the mortality data after the division of the (1 – TOT(h,t)) factor, while the lower curve (in red) is the original data.

As can be seen in this plot, the TOT(h,t) data starts noticeably affecting the given cancer mortality data around age 72.5 and this affect increases as the birth cohort becomes older. As an example, for age 87.5, the regular OBS(h,t) data (per 100,000) is roughly 280, whereas the data after the including the TOT(h,t) is approximately 330, about a 18 % increase. This magnitude of increase, and magnitudes higher, can be seen for all ages of the cohort after age 85.

This example demonstrates the important of the TOT(h,t) data. The TOT(h,t) data takes into account the way the original data sets for the observed cancer mortality rates were set up. Including the division of the (1 – TOT(h,t)) term in the adjusted model helps to paint a more clear picture of the rate of incidences of the given cancer for a user of the

CancerFit program. This adjustment thus leads to more accurate fits and values of the parameters in the model.

It deserves another mention that the TOT(h,t) data better approximates the mortality data curve at ages toward the end of the curve. This is important as it is this part of the curve which led Professor Thilly to begin thinking about the possibility of a maximum age at which a person may be initiated for a given cancer. It is also the part of the curve to which many current cancer researchers pay little attention. This inclusion of the TOT(h,t) data should draw cancer researchers' attention as to what is actually happening at the end of the mortality curve and why the curve in old age is convex.

5 Results

5.1 Benefits of TOT, SUR, and REP Additions

The user now has the ability to load TOT(h,t), SUR(h,t), and REP(h,t) data into the CancerFit program. It is important to note that the user does not have to enter each of these data sets. A user may load any combination of these data files. For example, if the TOT(h.t) data is not entered by a user, the default value of TOT(h,t) for all ages is 0.0. Furthermore, if a user does not enter REP(h,t) data, the default value is 1.0. Finally, if the user does not enter SUR(h,t) data, the default value is 0.0. Therefore, a user can plot and fit the original OBS(h,t) data by simply not entering any of these data.

It is worthwhile to note for the SUR(h,t) data that while a user may now load a data file containing age-specific survival fractions for a birth cohort, the SUR(h,t) values may still be set to a constant by the user. In the "Set Parameters" pane of the CancerFit

program, a user is given the option to enter one value for SUR(h,t) or to check a box to indicate that the user would like to load a data file for SUR(h,t). This option exists for such cases as when the user is studying pancreatic cancer, for example, and the survival fraction is accepted as the constant 0. In other examples, a user may believe or want to try a constant survival fraction of 0.5. In either case, this option makes it easier for the user and adds flexibility to the CancerFit program.

When the user enters the Find Fits pane, the program automatically calculates and plots OBS**(h,t) based on data files loaded. If any of the data files do not contain information for specific ages, in other words if the data files are not complete, the default values are used for those ages. Below is an example of the OBS**(h,t) plotted when the user enters the Find Fits pane.

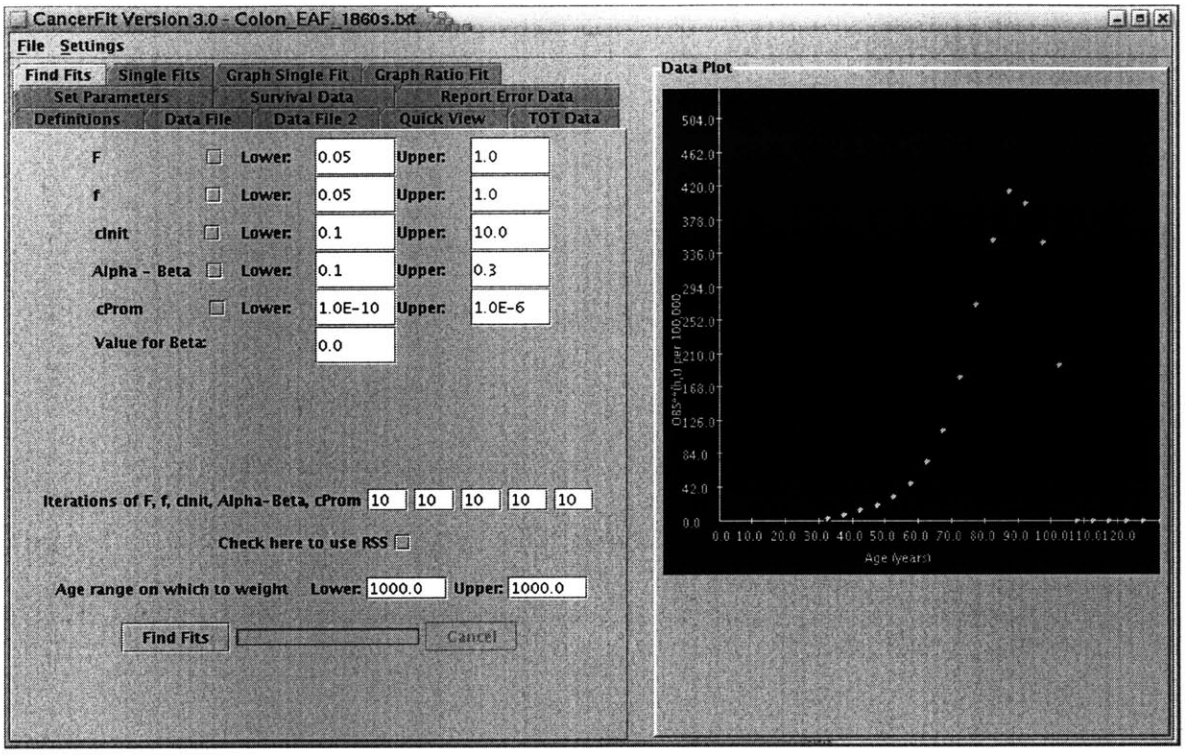


Figure 15: This graph shows the re-plotting of the OBS**(h,t) data when the user enters the Find Fits window to set up the parameter bounds for fitting. The re-plotted adjusted curve is what the CancerFit application will fit, thus is it important to allow the user to inspect this curve before setting bounds and iterations over each of the parameters.

It is appropriate to re-plot OBS**(h,t), which is the adjusted OBS(h,t), when a user goes to the Find Fits pane because it is OBS**(h,t) that the program fits with the parameter ranges the user enters. Therefore, one can examine the curve to be fit, select the parameter bounds and iterations over each parameter range appropriately, and then start the algorithm to find the best fits.

To demonstrate the complete effect when using the TOT(h,t), SUR(h,t), and REP(h,t) data together, below we show a plot of OBS**(h,t) against OBS(h,t) for the 1860's European American Females below. (OBS**(h,t) is the yellow curve, and OBS(h,t) is the red curve.)

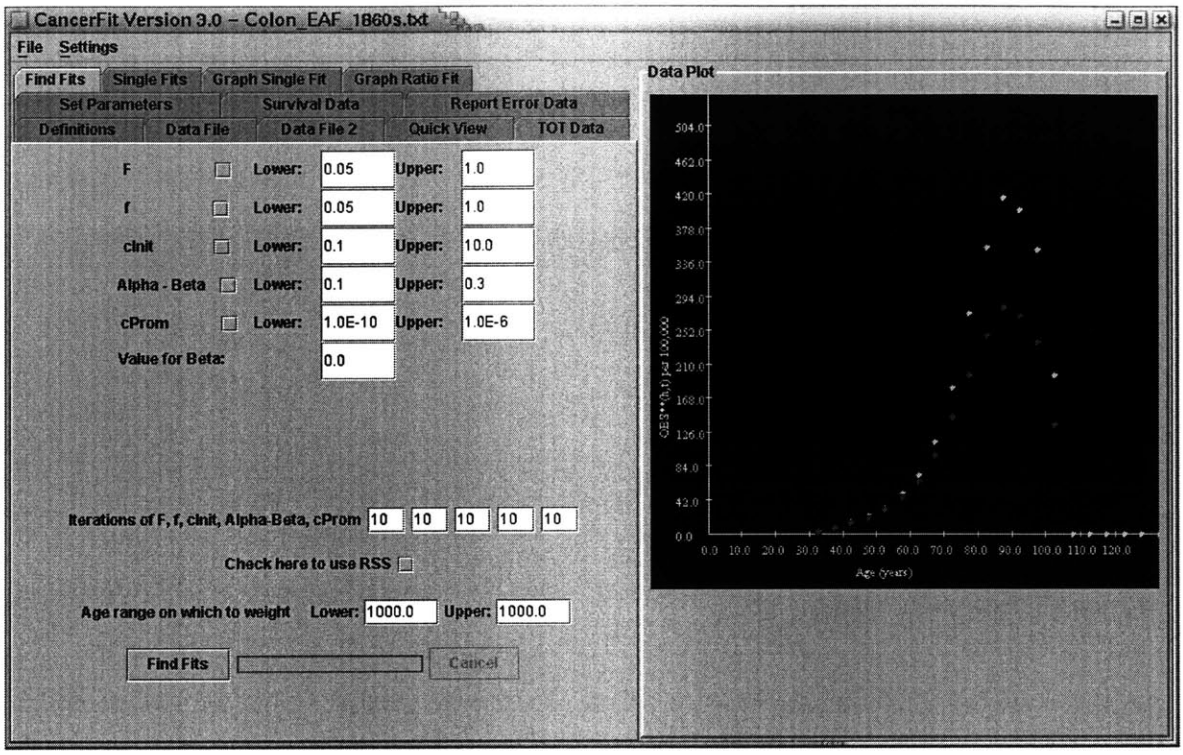


Figure 16: This plot demonstrates the overall effect of the SUR(h,t), REP(h,t), and TOT(h,t) data on the original mortality data. The upper curve (in yellow), is OBS**(h,t), which is adjusted for the SUR(h,t), REP(h,t), and TOT(h,t) data loaded into CancerFit. The lower curve (in red), is the original mortality data, OBS(h,t).

The OBS**(h,t) is an improved age-specific probability of an individual receiving the cancer in the midyear of each interval, given the individual is still alive. By using the TOT(h,t), SUR(h,t), and REP(h,t), we transform the original data to create a more accurate picture of the given cancer mortality curve. This subsequently allows for more exact fits of the parameters of the model, helping researchers better understand the reasons behind the shape of the mortality curve.

5.2 Original Findings

One of the more unexpected and interesting results of the ratio work completed for this thesis was the discovery made using the ratio functionality to compare the mortality data

of colon cancer for European females born in the 1860's against European males born in the 1860's. Below is the ratio plot of these two data sets (the ratio of females to males is plotted).

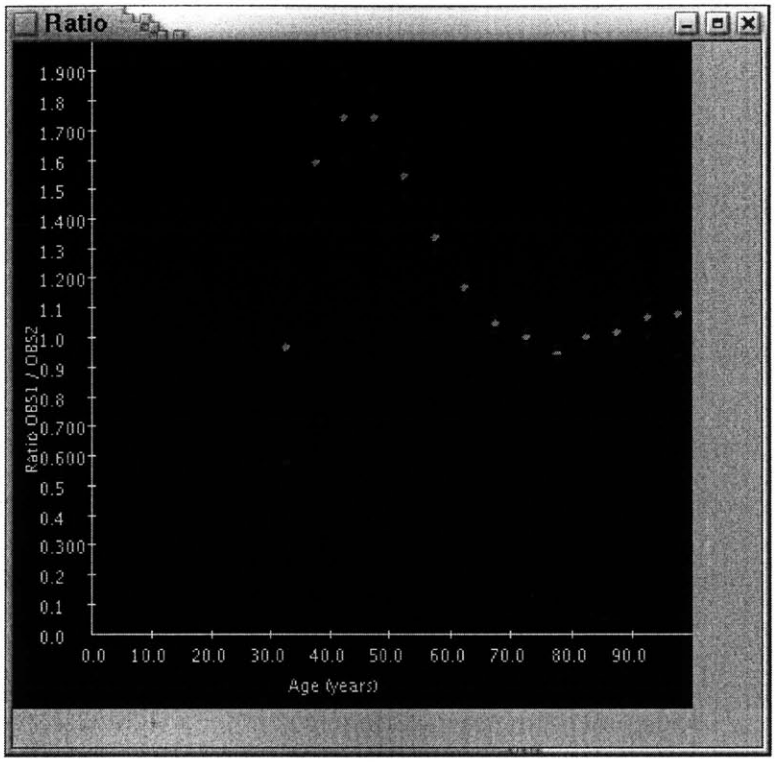


Figure 17: The ratio plot for the mortality data for the European American Females (EAF) born in the 1860's compared with the mortality data for the European American Males (EAM) born in the 1860's. Note the high relative risk of the females early in life, as indicated by a ratio of approximately 1.8 for ages 42.5 and 47.5.

As demonstrated by the plot, females born in the 1860's appear to have a much greater risk of mortality from colon cancer in the middle ages of life when compared to men. Especially in ages of 47.5 to 57.5 where the data is well-defined, the females have a risk roughly 1.8 times the risk of males (at age 47.5) to roughly 1.4 times the risk of males (at age 57.5). This type of high risk in the earlier years for cancers in females reminded Professor Thilly of the types of data he has seen in breast cancer, where females

experience high levels of mortality in the pre-menopausal and early post-menopausal age interval.

One notes in addition that in post-menopausal women and men (age 65-104) the age specific risk is not significantly different from 1.0. This influences thinking about carcinogenesis as well. The Thilly group was seriously considering the possibility that colon cancer mutation rates were high enough to create about three preneoplastic colonies in each colon. In such a case the difference in the sizes of colons in men and women would not be expected to yield a significantly different fraction of the two populations' lifetime risk. These ratio data are consistent with this new idea and suggest additional possibilities to cancer researchers.

This discovery is currently being researched in more depth. It is important to note that this same ratio (females compared to males) has already been observed for the 1890's birth cohorts as well. The 1890's female versus male mortality curve ratio shows similar form to the above plot, with slight differences in the magnitude of the ratio. While it is odd that these findings were not made earlier in cancer research, it speaks to the benefit of the functionality added in this ratio work. Not only may a researcher compare any two data sets, the comparison may be done in a quick and easy manner.

5.3 Computational Speed

With the additional functionality and improvements in the code, the CancerFit program currently allows cancer researchers to complete tests of 10^{10} iterations in roughly 48 hours. For comparison, during his ph.D thesis work, Pablo Herrero-Jimenez used Matlab to calculate fits for mortality curves. Roughly, Herrero-Jimenez could complete 10^4

iterations of calculations in a day. Furthermore, after David Hensle transported the CancerFit program from Fortran to Java, the program was able to run 5 x 108 iterations in 24 hours. It is clear that the CancerFit program has come a long way, not only in functionality, but also in computational speed; there is, however, a continuing need for improvement. We discuss this need, as well as some possible methods for solutions, in the Future Work section.

6 Future Work

The new version of the CancerFit application offers cancer researchers new functionality and improved computational speed to test quantitative hypotheses against actual human cancer experience. However, the computation time for a large number of iterations over the risk parameters when fitting the mortality data can and should be reduced. After working with the program for this thesis work, my impression is the next step should be to change the code as to produce the same results faster, possibly in the ordering of the calculations or in the way the results are computed and compared to find the best results. Alternatively, approximations to equations in the code may be exploited, thereby limiting the number of computations that must be completed and speeding up the program.

Currently the OBS(h,t) and TOT(h,t) data resides in excel files and must be manually transported into text files in order to be loaded into CancerFit. In order to make it easier for cancer researchers to test their theories regarding any data set, future work should also include permitting direct feeds of OBS(h,t) and TOT(h,t) data from the spreadsheet files into the CancerFit program for automatic parameter calculations, ratios and other logical operations across all birth year cohorts and cancer types.

Finally, there is much work to be done in comparing data sets now that the ratio functionality has been added. Specifically, analysis should seek to find related risks between different populations. Also, the ratio functionality may be used to search for cancer forms with shared risks. These studies may be done by searching for data sets with identical "F", or fraction at risk, parameters, as well as by finding cancer forms that have "f" parameters that sum to one.

A Source Code

```
/*
 * Fit.java
 */

package BioFit;

import java.awt.event.*;
import javax.swing.JMenu;
import javax.swing.JMenuItem;
import javax.swing.JMenuBar;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.KeyStroke;
import javax.swing.JOptionPane;
import javax.swing.JFileChooser;
import javax.swing.*;
import java.io.*;
import java.awt.*;
import java.io.FileWriter;
import java.io.BufferedWriter;
import javax.swing.JTabbedPane;
import java.awt.image.BufferedImage;
import java.util.Vector;
import java.lang.Math;
import java.awt.geom.AffineTransform;
import java.lang.Double;
import java.awt.geom.Ellipse2D.Float;
import javax.swing.table.AbstractTableModel;
import java.awt.Point;
import java.util.StringTokenizer;
import javax.swing.JProgressBar;
import javax.swing.event.ListSelectionListener;
import javax.swing.event.ListSelectionEvent;
import java.util.*;

public class Fit extends JFrame {

   //Global GUI variables, some of these must be constructed in order so that
   //The listeners are instantiated before the panels that use them
   MenuListener menuListener = new MenuListener();
   TabbedPaneListener tabbedPaneListener = new TabbedPaneListener();
   JTabbedPane tabbedPane = new JTabbedPane();
   DataFileView dataFileView = new DataFileView();
   DataFileView2 dataFileView2 = new DataFileView2();
   SurvivalView survivalView = new SurvivalView();
   TOTView totView = new TOTView();
   ErrorView errorView = new ErrorView();
   GraphWindow graph = new GraphWindow(400,400);
   GraphWindow graphTSquared =  new GraphWindow(400, 400);
   GraphWindow graphLogLinear = new GraphWindow(400, 400);
   GraphWindow graphLogLinear2 = new GraphWindow(400, 400);
   GraphWindow graphLogLog = new GraphWindow(400, 400);
```

```
GraphWindow graphCumObs = new GraphWindow(400, 400);
EstimateWindow estimateWindow = new EstimateWindow();
EstimateLogWindow estimateLogWindow = new EstimateLogWindow();
DataWindow dataWindow = new DataWindow();
EquationsWindow equationsWindow = new EquationsWindow();
DataWindow2 dataWindow2 = new DataWindow2();

SetParametersWindow setParametersWindow;
BestFitWindow bestFitWindow;
SelectFitWindow selectFitWindow = new SelectFitWindow();
GraphSingleFitWindow graphSingleFitWindow = new GraphSingleFitWindow();
SettingsWindow settingsWindow = new SettingsWindow();
DefinitionsWindow definitionsWindow = new DefinitionsWindow() ;
CompareRatioWindow compareRatioWindow = new CompareRatioWindow();

SwingWorker runFindFitWorker = null; // For multithreading the long math

//Global Math variables
ArrayList Ratio_global;
ArrayList Variance_global;
ArrayList Ratio_plus_global;
ArrayList Ratio_minus_global;

double population = 100000.0;
double[] pobs = new double[500];
double[] age = new double[1000];
double[] t = new double[50];
double[] obs = new double[50];
double[] cases = new double[50];
double[] cases_adj = new double[50]; // for adjusting cases to graph OBS** correctly
double[] pop = new double[50];
double[] t2 = new double[50];
double[] obs2 = new double[50];
double[] cases2 = new double[50];
double[] pop2 = new double[50];
double[] w = new double[50];
double[] survival_age = new double[50];
double[] tot = new double[50]; // ADD TOT Variable 02/11/04 JK
double[] tot_age = new double[50];
double[] tot_pop = new double[50];
double[] report_error_year = new double[50];
double[] reportError = new double[50]; // ADD R data (error) Variable 02/11/04
double[][] X = new double[50][6];
double[][] S = new double[6][6];
double[][] Sinv = new double[6][6];
double[] se = new double[6];
double[][] solutions = new double[100000][9]; //Turn this into a vector!
double[][] solutions2 = new double[100000][9];
double[] fit = new double[50];
double[] slp = new double[50];
double[] xx = new double[50];
double[] yy = new double[50];
double[] hA = new double[50];
double[] inthA = new double[50];
double[] Pobs = new double[50];
double[] intPobs = new double[50];
```

```
  double[] hfineA = new double[1000];
  double[] inthfineA = new double[1000];
  double[] survivalFraction = new double[50];

  double brth, deth, rA, rB, rC, rX;
  double c, ninit, minit;
// survival fraction default
  double survivalDefault; // initialized below to 0.0
double reportErrorDefault; // initialized below to 1.0
double totDefault; // initialized below to 0.0
double Frac, f;
  int ngrid, igrid;
  double Frl, Fru, fl, fu, cl, cu, alminbel, alminbeu, betal, betau, rAl, rAu, alpha;
  double alminb, hu,hl,ho,lu,ll,lo, stepo, stendo, stependo;
  double Fraco, fo, co, alminbeo, betao, rAo, dio, slpo, detao, disto, brtho, detho;
  double  alminbehu, alminbehl, stepu, stepl;
  double alminbeho,alminbelo;
  double slopemax, detamax, basedist;
  int windicator;
  int pindicator[] = new int[6];
  final int[] nloop = {10, 10, 10, 10, 10, 10};
  int[] nloopOriginal = nloop;

  int dataLine;
  int dataLine2;
  int dataLine3;
  int dataLineOBS;
  int dataLineSquared;
  int dataLineLogLinear;
  int dataLineLogLinear2;
  int dataLineLogLog;
  int cumObsLine ;

  int padLength = 20;  //padLength is the total length of each entry in the output file
  int nobs, nfits, nobs2;

  boolean reRan = false; //don't display the choices in the table the second time on best fits, there are to
many
  boolean runABMod = false;
  boolean breakOut = false; //Used to break out of our huge Loops if the user
                  //chooses to
  double weightsAgel = 1000.0 ;  //set to allow the Sum of Squares to use weighting from this age up
  double weightsAgeu = 1000.0 ;

  int ageIndexl = (new Long(Math.round(weightsAgel/5.0))).intValue() ;
  int ageIndexu = (new Long(Math.round(weightsAgeu/5.0))).intValue() ;

  int pobsline ;
  double[] loggedArray = new double[50] ;
  double[] logLogArray = new double[50] ;

  double ageAtMax ;
  int ageAtMaxInt ;
  double realAgeAtMax;

  DoubleNumberField sField ;
```

```
DoubleNumberField areaField ;
DoubleNumberField xSquaredField ;

double areaGlobal ;

int graphSingleFitLine = -6 ;

double maxt;
double maxcases;
double set_maxt;
boolean first_time_maxt = true;

int ageMaxForN = 150 ;
double[] N = new double[ageMaxForN] ;
double NMax  ;
int initiateableAge ;

double[][] sortedsolutions ;

WholeNumberField xstartBox ;
WholeNumberField xendBox ;
WholeNumberField ystartBox ;
WholeNumberField yendBox ;

double area1, area2, ratioOfAreas ;

static String versionNumber = "3.0" ;

boolean newEquation = false ;
boolean enterSurvivalDataFile = false;
boolean enterReportErrorDataFile = false;
boolean setupGrowthFunction = false;
boolean plotted_OBS = false;
int extension = 10 ;

double tempArea = 0.0 ;
boolean first_time = true;
/** Creates new Fit */
public Fit() {
   System.out.println("Loading....");

   //Do all the math initialization
   survivalDefault = 0.0 ;
        reportErrorDefault = 1.0;
        totDefault = 0.0;

        setupSurvivalArray(survivalDefault);
        setupReportErrorArray(reportErrorDefault);
        setupTOTArray(totDefault);

   ninit = 2;
   minit = 1;
   ngrid = 200;
   igrid = 200;
   pobsline = -12 ;
```

```
        setParametersWindow = new SetParametersWindow();

        //Set the defualt values for all the parameters
        Frl = 0.05;
        //Frl = 1.0;
        Fru = 1.0;
        fl = 0.05;
        //fl = 1.0;
        fu = 1.0;
        cl = 0.1;
        cu = 10;
        alminbel = .1;
        alminbeu = .3;
        betal = 0;
        betau = 0;
        rAl = Math.pow(10, -10);
        rAu = Math.pow(10, -6);
        alpha = 9.143 ;
        /*cl = 1.08E-4;
        cu = 1.24E-4;
        alminbel = .143;
        alminbeu = .149;
        betal = 0;
        betau = 9;
        rAl = 8.0E-4;
        rAu = 8.22E-4;
        */
        bestFitWindow = new BestFitWindow();

        //close on window exit.
        addWindowListener(new WindowAdapter() {
           public void windowClosing(WindowEvent e) {
              System.exit(0);
           }
        });

        //Put in the Menu
        JMenuBar menuBar;
        JMenu menu;
        JMenuItem menuItem;


        menuBar = new JMenuBar();
        setJMenuBar(menuBar);

        menu = new JMenu("File");
        menu.setMnemonic(KeyEvent.VK_F);
        menu.getAccessibleContext().setAccessibleDescription(
        "The file menu");
        menuBar.add(menu);

        //New, Doesn't do anything right now
        menuItem = new JMenuItem("New", new
ImageIcon("C:\\javasource\\DrivingCoach\\graphics\\New16.gif"));
        menuItem.setAccelerator(KeyStroke.getKeyStroke(
        KeyEvent.VK_N, ActionEvent.ALT_MASK));
```

```
        menuItem.addActionListener(menuListener);
        menuItem.setActionCommand("New");
        menu.add(menuItem);

        //Load
        menuItem = new JMenuItem("Open", new
ImageIcon("C:\\javasource\\DrivingCoach\\graphics\\Open16.gif"));
        menuItem.setAccelerator(KeyStroke.getKeyStroke(
        KeyEvent.VK_O, ActionEvent.ALT_MASK));

        menuItem.getAccessibleContext().setAccessibleDescription(
        "Load a Mail file.");
        menuItem.setActionCommand("Open");
        menuItem.addActionListener(menuListener);
        menu.add(menuItem);

        //Close
        menuItem = new JMenuItem("Close", KeyEvent.VK_C);
        menuItem.getAccessibleContext().setAccessibleDescription(
        "Close a Mail file");
        menuItem.setActionCommand("Close");
        menuItem.addActionListener(menuListener);
        menu.add(menuItem);

        //Exit
        menuItem = new JMenuItem("Exit", KeyEvent.VK_X);
        menuItem.setActionCommand("Exit");
        menuItem.addActionListener(menuListener);
        menu.add(menuItem);

        //The Settings Menu
        menu = new JMenu("Settings");
        menu.setMnemonic(KeyEvent.VK_S);
        menu.getAccessibleContext().setAccessibleDescription(
        "The settings menu");
        menuBar.add(menu);

        //Save  settings
        menuItem = new JMenuItem("Save Settings", KeyEvent.VK_N);
        menuItem.addActionListener(menuListener);
        menuItem.setActionCommand("Save Settings");
        menu.add(menuItem);

        //Load
        menuItem = new JMenuItem("Load Settings",  KeyEvent.VK_L);
        menuItem.setActionCommand("Load Settings");
        menuItem.addActionListener(menuListener);
        menu.add(menuItem);

        //Clear
        menuItem = new JMenuItem("Edit Settings", KeyEvent.VK_C);
        menuItem.setActionCommand("Edit Settings");
        menuItem.addActionListener(menuListener);
        menu.add(menuItem);
```

```
        this.getContentPane().setLayout(new BorderLayout());
            // here it seems that order matters in how they are displayed - JK 11/18/03
        //Build the JTabbed Pane
        tabbedPane.addTab("Definitions", null, definitionsWindow) ;
            // added 5/5/04 for Equations Pane
            tabbedPane.addTab("Equations", null, equationsWindow);
            tabbedPane.addTab("Data File", null, dataFileView, "Click to view or Modify Data in File");
        tabbedPane.addTab("Data File 2", null, dataFileView2);
            tabbedPane.addTab("Quick View", null, estimateWindow);
            tabbedPane.addTab("TOT Data", null, totView);
            //tabbedPane.addTab("Survival Data", null, survivalView);
            //tabbedPane.addTab("Report Error Data", null, errorView);
            //tabbedPane.addTab("Cum of Obs", null, graphCumObs);
        //tabbedPane.addTab("T Squared View", null, graphTSquared);
        //tabbedPane.addTab("Log Linear View", null, graphLogLinear);
        //tabbedPane.addTab("Log Log View", null, graphLogLog);
        //JLabel alphaminusbeta = new JLabel("a-b");
        //alphaminusbeta.setFont(new Font("Greek", Font.PLAIN, 12)) ;
        //tabbedPane.addTab(alphaminusbeta+"View", null, graphLogLinear2);
        //tabbedPane.addTab("Alpha-Beta View", null, graphLogLinear2);
        //tabbedPane.addTab("Quick Alpha-Beta View", null, estimateLogWindow);
        tabbedPane.addTab("Set Parameters", null, setParametersWindow);
            tabbedPane.addTab("Survival Data", null, survivalView);
            tabbedPane.addTab("Report Error Data", null, errorView);
        tabbedPane.addTab("Find Fits", null, bestFitWindow);
        tabbedPane.addTab("Single Fits", null, selectFitWindow);
        tabbedPane.addTab("Graph Single Fit", null, graphSingleFitWindow);
            tabbedPane.addTab("Graph Ratio Fit", null, compareRatioWindow);
        tabbedPane.setPreferredSize(new Dimension(500,500));
        this.getContentPane().add(tabbedPane, BorderLayout.WEST);

        //add a little popup window to the graph.
        Vector tmp = new Vector();
        tmp.add("Clear Excess Data Points");
        JPopupMenu tmp2 = createPopupMenu(tmp, menuListener);
        graph.addMouseListener(new PopupListener(tmp2));

        //Add in the Graph
        JPanel temp = new JPanel();
        temp.setBorder(BorderFactory.createTitledBorder(BorderFactory.createRaisedBevelBorder(), "Data
Plot"));
        temp.add(graph);
        temp.setPreferredSize(new Dimension(420,420));
        this.getContentPane().add(temp, BorderLayout.EAST);

    }


    /* This listens for actions in the menubar and in the buttons on the
     * tabbed panes*/

    public class MenuListener implements ActionListener {
        public void actionPerformed(ActionEvent e) {
            if(e.getActionCommand().compareTo("Exit") == 0) {
                System.exit(0);
```

```
        } else if(e.getActionCommand().compareTo("New") == 0 ||
e.getActionCommand().compareTo("Close") == 0){
            dataFileView.clearAll();
            graph.clearAll();
            graphTSquared.clearAll();
            graphLogLinear.clearAll();
            graphLogLinear2.clearAll();
            graphLogLog.clearAll();
            graphCumObs.clearAll();
            estimateWindow.clearAll();
            estimateLogWindow.clearAll();
            selectFitWindow.clearAll();
            resetVars();
        } else if(e.getActionCommand().compareTo("Open") == 0){
            JFileChooser chooser = new JFileChooser("C:\\MyJava\\SampleDir\\BioFit\\panceaf1890");
            int returnVal = chooser.showOpenDialog(Fit.this);
            if (returnVal == JFileChooser.APPROVE_OPTION) {
              //Load the file
              File file = chooser.getSelectedFile();
              Fit.this.setTitle("CancerFit Version "+ versionNumber + " - " + file.getName());
              //Send the file to the file tabbedPane.
              dataFileView.updateAndShowFile(file);
              dataFileView.plot();
            }
        } else if(e.getActionCommand().compareTo("Clear Excess Data Points") == 0){
              if(plotted_OBS = true){
                    graph.removeExcessLines(dataLineOBS); // we have already plotted OBS**, want to
remove all but that
              }
              else{
                    graph.removeExcessLines(dataLine);  // have not plotted OBS**, so keep original OBS
              }
        } else if(e.getActionCommand().compareTo("Edit Settings")  == 0) {
            settingsWindow.setLocation(Fit.this.getLocation().x + 400, Fit.this.getLocation().y + 50);
            settingsWindow.show();
        }
      }
    }


    /* This listens for Changes in the Tabbed Panes. */
    public class TabbedPaneListener implements ActionListener {
      public void actionPerformed(ActionEvent e) {
        if(e.getActionCommand().compareTo("DataFileView.Save") == 0) {
            runSaveDialog(1);
        } else if(e.getActionCommand().compareTo("DataFileView.Next") == 0) {
            dataFileView.next();
            tabbedPane.setSelectedIndex(3);
            //go to the next window
        } else if(e.getActionCommand().compareTo("DataFileView.Plot") == 0){
            dataFileView.plot();
        } else if(e.getActionCommand().compareTo("DataFileView2.Open") == 0){
            JFileChooser chooser = new JFileChooser("C:\\MyJava\\SampleDir\\BioFit\\panceaf1890");
            int returnVal = chooser.showOpenDialog(Fit.this);
            if (returnVal == JFileChooser.APPROVE_OPTION) {
              //Load the file
```

```
    File file = chooser.getSelectedFile();
    //Send the file to the file tabbedPane.
    dataFileView2.updateAndShowFile(file);
    dataFileView2.plot();
   }
} else if(e.getActionCommand().compareTo("DataFileView2.Save") == 0) {
  runSaveDialog(2);
} else if(e.getActionCommand().compareTo("DataFileView2.Plot") == 0){
  dataFileView2.setScale();

   } else if(e.getActionCommand().compareTo("DataFileView2.Data") == 0){

    dataWindow2.entryTable.updateTable(t, loggedArray);
    JFrame dataFrame = new JFrame("Data") ;
    dataFrame.getContentPane().add(dataWindow2) ;
    dataFrame.setSize(450, 450) ;
    dataFrame.setVisible(true) ;
   } else if(e.getActionCommand().compareTo("DataFileView2.Next") == 0) {
  tabbedPane.setSelectedIndex(4);
  //go to the next window
   } else if(e.getActionCommand().compareTo("SurvivalView.Save") == 0) {
    runSaveDialog(3);
   } else if(e.getActionCommand().compareTo("SurvivalView.Plot") == 0) {
    survivalView.setScale();
   } else if(e.getActionCommand().compareTo("SurvivalViewData.Plot") == 0){
    survivalView.plot();
   } else if(e.getActionCommand().compareTo("SurvivalView.Open") == 0) {
    JFileChooser chooser = new JFileChooser("C:\\MyJava\\SampleDir\\BioFit\\panceaf1890");
  int returnVal = chooser.showOpenDialog(Fit.this);
  if (returnVal == JFileChooser.APPROVE_OPTION) {
    //Load the file
    File file = chooser.getSelectedFile();
    //Send the file to the file tabbedPane.
    survivalView.updateAndShowFile(file);
    //survivalView.plot();
          }
   } else if(e.getActionCommand().compareTo("SurvivalView.Next") == 0) {
    if(setParametersWindow.reportErrorCheckBox.isSelected()){
          tabbedPane.setSelectedIndex(8); //go to reporting error data file entry pane
    }
         else {
          setupReportErrorArray(reportErrorDefault);
          bestFitWindow.plot(); // plot the appropriate OBS
          tabbedPane.setSelectedIndex(9);
         }
    //go to the next window
   } else if(e.getActionCommand().compareTo("ErrorView.Save") == 0) {
    runSaveDialog(3);
   } else if(e.getActionCommand().compareTo("ErrorView.Plot") == 0) {
    errorView.setScale();
   } else if(e.getActionCommand().compareTo("ErrorPlotData.Plot") == 0) {
    errorView.plot();
   } else if(e.getActionCommand().compareTo("ErrorView.Open") == 0) {
    JFileChooser chooser = new JFileChooser("C:\\MyJava\\SampleDir\\BioFit\\panceaf1890");
    int returnVal = chooser.showOpenDialog(Fit.this);
    if (returnVal == JFileChooser.APPROVE_OPTION) {
```

```
                    //Load the file
                    File file = chooser.getSelectedFile();
                    //Send the file to the file tabbedPane.
                    errorView.updateAndShowFile(file);
                    //errorView.plot();
                }
               } else if(e.getActionCommand().compareTo("ErrorView.Next") == 0) {
                bestFitWindow.plot(); // plot the appropriate OBS
                tabbedPane.setSelectedIndex(9);
                //go to the next window
               } else if(e.getActionCommand().compareTo("TOTView.Save") == 0) {
                runSaveDialog(3);
               } else if(e.getActionCommand().compareTo("TOTView.Plot") == 0) {
                totView.setScale();
               } else if(e.getActionCommand().compareTo("SetScaleTOT.Plot") == 0){
                totView.plot();
               } else if(e.getActionCommand().compareTo("TOTView.Open") == 0) {
                JFileChooser chooser = new JFileChooser("C:\\MyJava\\SampleDir\\BioFit\\panceaf1890");
                int returnVal = chooser.showOpenDialog(Fit.this);
                if (returnVal == JFileChooser.APPROVE_OPTION) {
                    //Load the file
                    File file = chooser.getSelectedFile();
                    //Send the file to the file tabbedPane.
                    totView.updateAndShowFile(file);
                    //totView.plot();
                }
               } else if(e.getActionCommand().compareTo("TOTView.Next") == 0) {
              tabbedPane.setSelectedIndex(6);
              //go to the next window
            } else if(e.getActionCommand().compareTo("SetScaleRatio.Plot") == 0){
              dataFileView2.plotRatio();
            } else if(e.getActionCommand().compareTo("DefinitionsWindow.Next") == 0){
              tabbedPane.setSelectedIndex(1);
                } else if(e.getActionCommand().compareTo("EquationsWindow.Next") == 0){
                 tabbedPane.setSelectedIndex(2);          // ADDED 5/5/04 for Equations Window
            } else if(e.getActionCommand().compareTo("EstimateWindow.Previous") == 0) {
              tabbedPane.setSelectedIndex(2);
            } else if(e.getActionCommand().compareTo("EstimateWindow.Next") == 0){
              tabbedPane.setSelectedIndex(5);
            } else if(e.getActionCommand().compareTo("EstimateLogWindow.Estimate") == 0) {
              estimateLogWindow.calculateMU(estimateLogWindow.entryTable.getSelIndex());
            } else if(e.getActionCommand().compareTo("EstimateLogWindow.Next") == 0) {
              tabbedPane.setSelectedIndex(10);
            } else if(e.getActionCommand().compareTo("SetParametersWindow.setupNArray") == 0) {

                if (setParametersWindow.maleCheckBox.isSelected()){
                setupNArray(Double.parseDouble(setParametersWindow.gammaBox.getText()),
Double.parseDouble(setParametersWindow.nMaxBox.getText()),
Integer.parseInt(setParametersWindow.initiateableAgeBox.getText()), true) ;
              }
              else {
                setupNArray(Double.parseDouble(setParametersWindow.gammaBox.getText()),
Double.parseDouble(setParametersWindow.nMaxBox.getText()),
Integer.parseInt(setParametersWindow.initiateableAgeBox.getText()), false) ;
              }
            } else if(e.getActionCommand().compareTo("SetParametersWindow.next") == 0) {
```

```
            if (setupGrowthFunction == false){
                Object[] options = { "OK" };
                JOptionPane.showOptionDialog(null, "Please Setup Growth Function", "Warning!",
JOptionPane.DEFAULT_OPTION, JOptionPane.WARNING_MESSAGE, null, options, options[0]);
            }
            else {// indicate to the user that they need to set the parameters by pressing button
              ninit = Double.parseDouble(setParametersWindow.nNum.getText()) ;
          minit = Double.parseDouble(setParametersWindow.mNum.getText()) ;
                if(setParametersWindow.survivalCheckBox.isSelected()){
                  tabbedPane.setSelectedIndex(7); //go to Survival Data File entry pane
                }
                else {
                  survivalDefault = Double.parseDouble(setParametersWindow.survivalNum.getText()) ;
                  setupSurvivalArray(survivalDefault);
                }
                if(setParametersWindow.reportErrorCheckBox.isSelected() &
!(setParametersWindow.survivalCheckBox.isSelected())){
                  tabbedPane.setSelectedIndex(8); //go to reporting error data file entry pane
                }
                else {
                  setupReportErrorArray(reportErrorDefault);
                }
                if(!(setParametersWindow.survivalCheckBox.isSelected()) &
!(setParametersWindow.reportErrorCheckBox.isSelected())){
                  bestFitWindow.plot(); // plot the appropriate OBS
                  tabbedPane.setSelectedIndex(9); // go to find fits entry pane
                }
            }

        } else if(e.getActionCommand().compareTo("BestFitWindow.FindFit") == 0) {
            selectFitWindow.clearAll();
            resetSolutions() ;
            solutions2 = new double[100000][9];
            bestFitWindow.runFindFitWorker();
        } else if(e.getActionCommand().compareTo("BestFitWindow.FindBestFit") == 0) {
            bestFitWindow.runBestFit();
        } else if(e.getActionCommand().compareTo("BestFitWindow.Cancel") == 0) {
            //runFindFitWorker.stop();
            breakOut = true;
            bestFitWindow.findBestFit.setEnabled(true);
            bestFitWindow.findFit.setEnabled(true);
            bestFitWindow.cancel.setEnabled(false);
        } else if(e.getActionCommand().compareTo("SelectFitWindow.reRun") == 0) {
              selectFitWindow.reRun(); //comment out this line to make the reRun just go back to original
settings
            tabbedPane.setSelectedIndex(9);
        } else if(e.getActionCommand().compareTo("SelectFitWindow.plotPobs") == 0) {
            selectFitWindow.setScale() ;
        } else if(e.getActionCommand().compareTo("SetScale.graph") == 0) {
            selectFitWindow.plotPobs() ;
        } else if(e.getActionCommand().compareTo("SelectFitWindow.save") == 0) {
            runSaveDialog2() ;
        } else if(e.getActionCommand().compareTo("SelectFitWindow.data") == 0) {
            dataWindow.entryTable.updateTable(t, loggedArray);
            JFrame dataFrame = new JFrame("Data") ;
            dataFrame.getContentPane().add(dataWindow) ;
```

```
            dataFrame.setSize(450, 450) ;
            dataFrame.setVisible(true) ;
          } else if(e.getActionCommand().compareTo("GraphSingleFitWindow.plot") == 0) {
            graphSingleFitWindow.plot() ;
          } else if(e.getActionCommand().compareTo("GraphSingleFitWindow.unplot") == 0) {
            graphSingleFitWindow.unplot() ;
          } else if(e.getActionCommand().compareTo("GraphSingleFitWindow.plotPobs") == 0) {
            graphSingleFitWindow.plotPobs() ;
              } else if(e.getActionCommand().compareTo("CompareRatioWindow.setScale") == 0) {
               compareRatioWindow.setScale();
          } else if(e.getActionCommand().compareTo("CompareRatioWindow.plotRatio") == 0) {
               compareRatioWindow.plotRatio() ;
              } else if(e.getActionCommand().compareTo("CompareRatioWindow.plotFit") == 0) {
               compareRatioWindow.plotFit() ;
              } else if(e.getActionCommand().compareTo("CompareRatioWindow.unplot") == 0) {
               compareRatioWindow.unplot() ;
              }
      }
    }

  // Listeners for the popup Menu. */
    private class PopupMenuListener implements ActionListener {
      public void actionPerformed(ActionEvent e) {
        JMenuItem item = (JMenuItem)e.getSource();
        String text = item.getText();

        if(text.compareTo("Re-run Good Fits with this line") == 0) {
          selectFitWindow.reRun();
          //bestFitWindow.runFindFitWorker();
          //tabbedPane.setSelectedIndex(6);
        }
        //not a menu item, just return
        return;
      }
    }

    /** Listener to popup a meun in the table. */
    class PopupListener extends MouseAdapter {
      //A handle to the window to popup
      JPopupMenu popup = null;
      PopupListener(JPopupMenu menu) {
        popup = menu;
      }
      public void mousePressed(MouseEvent e) {
        maybeShowPopup(e);
      }
      public void mouseReleased(MouseEvent e) {
        maybeShowPopup(e);
      }
      private void maybeShowPopup(MouseEvent e) {
        if (e.isPopupTrigger()) {
          popup.show(e.getComponent(),
          e.getX(), e.getY());
        }
      }
    }
```

```
/** Displays defintions*/
public class DefinitionsWindow extends JPanel {
   JScrollPane scrollPane = null;
   JTextArea dataView = new JTextArea();
   //should be changed to be relative somehow
   //File defFile = new File("/Users/dahens/Desktop/BioFit/Definitions.txt") ;
   File defFile = new File("BioFit/Definitions.txt") ;

   DefinitionsWindow(){

      //Initialize the Panel, we need to put the dataView text area into
      //the scroll Pane.
      setLayout(new BorderLayout());
      dataView.setEditable(false) ;
      scrollPane = new JScrollPane(dataView);
      scrollPane.setPreferredSize(new Dimension(400,500));
      add(scrollPane, BorderLayout.CENTER);
      //load the definitions file
      loadFile(defFile) ;

      //Create the Bottom Panel with Buttons in it
      JPanel tempPanel = new JPanel();
      JButton tempButton = new JButton("Next Step >>");
      tempButton.addActionListener(tabbedPaneListener);
      tempButton.setActionCommand("DefinitionsWindow.Next");
      tempPanel.add(tempButton);
      tempPanel.setPreferredSize(new Dimension(400, 50));

      add(tempPanel, BorderLayout.SOUTH);
   }

   /*void loadFile(File file) {
      try{
         //Load the File
         BufferedReader reader = new BufferedReader(new FileReader(file));
         String line = reader.readLine();
         String fileText = new String("");

         //We are assuming that the file is in proper format.
         //Now make a huge string of one file and display it.

         line = reader.readLine();

         while(line != null) {
            fileText += "\n";
            fileText += line;
            line = reader.readLine();
         }
         fileText += "\n";

         //Display the file
         dataView.setText(fileText);
         dataView.setCaretPosition(0);

      }catch (Exception e1) {
```

```
            e1.printStackTrace();
          }
        }*/


    void loadFile(File file){
          String fileText = new String("") ;
          fileText += "\n" ;
fileText += "TERMS";
fileText += "\n" ;
fileText += "Preneoplastic Colony:  Pre-cancerous colony that has a net growth rate of alpha-beta";
fileText += "\n" ;
fileText += "Neoplastic Colony:     Cancerous colony that leads to death";
fileText += "\n" ;
fileText += "Initiation:          Creation of first surviving preneoplastic cell";
fileText += "\n" ;
fileText += "Promotion:           Creation of first surviving neoplastic cell";
fileText += "\n" ;
fileText += "\n" ;

fileText += "POPULATION PARAMETERS";
fileText += "\n" ;
fileText += "G:   Fraction of the population that is genetically at risk";
fileText += "\n" ;
fileText += "E:   Fraction of the population that is environmentally at risk";
fileText += "\n" ;
fileText += "F:   Fraction of the population at lifetime risk (F = Sum(Gi x Ei)) ";
fileText += "\n" ;
fileText += "f:   Factor accounting for forms of mortality with shared risk";
fileText += "\n" ;
fileText += "h:   Calendar year of birth" ;
fileText += "\n" ;
fileText += "t:   Age (years) at death" ;
fileText += "\n" ;
fileText += "OBS(h,t):   OBServed mortality rate for a specific cancer";
fileText += "\n";
fileText += "TOT(h,t):   TOTal recorded mortality rate";
fileText += "\n";
fileText += "SUR(h,t):   SURvival fraction for specific cancer";
fileText += "\n";
fileText += "REP(h,t):   REPorted fraction of deaths with interpretable causes";
fileText += "\n" ;
fileText += "\n" ;

fileText += "PARAMETERS";
fileText += "\n" ;
fileText += "a[max]:         The age through which initiation is able to occur";
fileText += "\n" ;
fileText += "Na:             The number of cells at risk of initiation at age a";
fileText += "\n" ;
fileText += "Nmax:           The maximum number of cells at risk of initiation";
fileText += "\n" ;
fileText += "cInit:          Constant of initiation";
fileText += "\n" ;
fileText += "Alpha:       Rate of cell division for cells at risk of promotion in preneoplastic colonies";
fileText += "\n" ;
```

```
fileText += "Beta:          Rate of cell death for cells at risk of promotion in preneoplastic colonies";
fileText += "\n" ;
fileText += "Alpha-Beta:   Net growth rate of cells at risk of promotion in preneoplastic colonies  (alpha-
beta = μ)";
fileText += "\n" ;
fileText += "cProm:        Constant of promotion";
fileText += "\n" ;
fileText += "n:            Number of required initiation events (n = 1, 2, 3, ...)";
fileText += "\n" ;
fileText += "m:            Number of required promotion events (m = 0, 1, 2, ...)" ;
fileText += "\n" ;

        dataView.setText(fileText);
        dataView.setCaretPosition(0);


    }
  }

  /** Displays equations*/
  public class EquationsWindow extends JPanel {
    JScrollPane scrollPane = null;
    JTextArea dataView = new JTextArea();
    File defFile = new File("BioFit/Equations.txt") ;

    EquationsWindow(){

      //Initialize the Panel, we need to put the dataView text area into
      //the scroll Pane.
      setLayout(new BorderLayout());
      dataView.setEditable(false) ;
      scrollPane = new JScrollPane(dataView);
      scrollPane.setPreferredSize(new Dimension(400,500));
      add(scrollPane, BorderLayout.CENTER);
      //load the definitions file
      loadFile(defFile) ;

      //Create the Bottom Panel with Buttons in it
      JPanel tempPanel = new JPanel();
      JButton tempButton = new JButton("Next Step >>");
      tempButton.addActionListener(tabbedPaneListener);
      tempButton.setActionCommand("EquationsWindow.Next");
      tempPanel.add(tempButton);
      tempPanel.setPreferredSize(new Dimension(400, 50));

      add(tempPanel, BorderLayout.SOUTH);
    }

  void loadFile(File file){
      String fileText = new String("") ;
      fileText += "\n" ;
fileText += "EQUATIONS";
fileText += "\n" ;
fileText += "\n";
fileText += "\n";
fileText += "                                        F(h,t) * Pobs(h,t) ";
```

```
fileText += "\n" ;
fileText += "OBS(h,t) =     ---------------------------------------------------------------------------------------- ";
fileText += "\n" ;
fileText += "                    F(h,t) + (1 - F(h,t)) * e^(1/( f(h,t) * S Pobs(h,t)*(1- SUR(h,t)) dt))";
fileText += "\n" ;
fileText += "\n" ;
fileText += "                              OBS(h,t)";
fileText += "\n" ;
fileText += "OBS*(h,t) =    ---------------------------------- " ;
fileText += "\n" ;
fileText += "                    REP(h,t) * (1 - SUR(h,t))";

fileText += "\n" ;
fileText += "\n" ;
fileText += "                                        OBS(h,t)" ;
fileText += "\n" ;
fileText += "OBS**(h,t) =   ---------------------------------------------------- " ;
fileText += "\n" ;
fileText += "                    REP(h,t) * (1 - SUR(h,t)) * (1 - TOT(h,t)) " ;
fileText += "\n" ;
fileText += "\n" ;
fileText += "Pobs(h,t) =  1 - e^Vobs(h,t)";
fileText += "\n" ;
fileText += "\n" ;
fileText += "Vobs(h,t) =  Cinit * (alpha - beta)/alpha * S a * Na * { d(1 - e^(-Cprom* (alpha / (alpha -
beta))^2 *2^((alpha - beta)*(t - a)))) / d (t- a)} da   (n = 2, m = 1)";
fileText += "\n" ;
fileText += "\n" ;
fileText += "Cinit =  2*Ri*Rj*Nmax   (n = 2) ";
fileText += "\n" ;
fileText += "\n" ;
fileText += "Cprom =  S a * Na * { d(1 - e^(-rA* (alpha / (alpha - beta))^2 *2^((alpha - beta)*(t - a) + 1))) /
d (t- a)} da    (m = 1)";
fileText += "\n" ;
fileText += "\n" ;
fileText += "X =  2*Ri*Rj*((alpha-beta)/alpha) * S a * Na da";
fileText += "\n" ;
fileText += "\n" ;

        dataView.setText(fileText);
        dataView.setCaretPosition(0);


    }
  }

  /** A simple viewer and modifier for a data file.  A file must be in the
   ** following format, [age    deaths    population] */
  public class DataFileView extends JPanel {
    JScrollPane scrollPane = null;
    JTextArea dataView = new JTextArea();
    JLabel filename = new JLabel("File Name");
    int numlines = 0;

    DataFileView(){
```

```
    //Initialize the Panel, we need to put the dataView text area into
    //the scroll Pane.
    setLayout(new BorderLayout());
    scrollPane = new JScrollPane(dataView);
    scrollPane.setPreferredSize(new Dimension(400,500));
    add(filename, BorderLayout.NORTH);
    add(scrollPane, BorderLayout.CENTER);
    //Create the Bottom Panel with Buttons in it
    JPanel tempPanel = new JPanel();

    JButton tempButton = new JButton("Open File");
    tempButton.setActionCommand("Open");
    tempButton.addActionListener(menuListener);
    tempPanel.add(tempButton);

    tempButton = new JButton("Save File");
    tempButton.setActionCommand("DataFileView.Save");
    tempButton.addActionListener(tabbedPaneListener);
    tempPanel.add(tempButton);

    tempButton = new JButton("Plot");
    tempButton.setActionCommand("DataFileView.Plot");
    tempButton.addActionListener(tabbedPaneListener);
    tempPanel.add(tempButton);


    tempButton = new JButton("Next Step >>");
    tempButton.addActionListener(tabbedPaneListener);
    tempButton.setActionCommand("DataFileView.Next");
    tempPanel.add(tempButton);
    tempPanel.setPreferredSize(new Dimension(400, 50));

    add(tempPanel, BorderLayout.SOUTH);
}

void clearAll() {
    numlines = 0;
    dataView.setText("");
    graph.clearData();
}

void updateAndShowFile(File file) {
    try{
        //Load the File into the window and update the next tabbed pane with
        //the Slope and alpha-beta files.
        numlines = 1;
        filename.setText(file.getName());

        BufferedReader reader = new BufferedReader(new FileReader(file));
        String line = reader.readLine();
        String fileText = new String("");

        //We are assuming that the file is in proper format.
        //Now make a huge string of one file and display it.

        fileText = " Age          Incidence          Population\n" +
```

```
            "------      ----------          ----------\n" + line;
      line = reader.readLine();

      while(line != null) {
         numlines++;
         fileText += "\n";
         fileText += line;
         line = reader.readLine();
      }
      fileText += "\n"; //we need this to find whitespace after the
      //last number.

      //Display the file and parse the string so we can use the
      //values.
      dataView.setText(fileText);
      dataView.setCaretPosition(0);

   }catch (Exception e1) {
      e1.printStackTrace();
   }
}
//Creates array t, cases, and pop.  sets nobs, and plots data.  Allows
//For next step of choosing the slope.
public void plot() {

   //Parse the dataview to get the numbers out.
   StringBuffer stringBuffer = new StringBuffer(dataView.getText());
   nobs = 0;

   for(int i =0; i < numlines; i++){
      t[i] = getDouble(stringBuffer);
      cases[i] = getDouble(stringBuffer);
      pop[i] = getDouble(stringBuffer);
            cases_adj[i] = cases[i]; // in order to initialize the cases[] for OBS** correctly
      nobs++;
   }

   maxcases = 0;
   maxt = 0;

   /****OBS*****/
   //we know t and cases are the same length arrays, so we can
   //this in one loop.
   double normalizedIncidents ;
   for (int i =0; i<t.length; i++) {
      normalizedIncidents = cases[i]*(100000.00/pop[i]) ;
      if(maxcases < normalizedIncidents) {
         maxcases = normalizedIncidents;
         ageAtMax = t[i] ;
         ageAtMaxInt = i ;
      }
            if(maxt < t[i]) {
              maxt = t[i];
      }
   }
     // need to set maxt so that we can maintain it and still calc Area- 4/23/04
```

```
            if(first_time_maxt == true){
              set_maxt = maxt;
              first_time_maxt = false;
            }
          //System.out.println("maxt: " + maxt);
          double thisAge = t[numlines-1] ; //- nned this to calc Area correctly later
          for (int j= numlines; j <50; j++){
            thisAge = thisAge + 5.0;
            t[j] = thisAge;
          }

          //Initialize and plot the graph
          graph.setStart(0,0);
          graph.setXLabel("Age (years)");
              // really plotting OBS(h,t) since no survival data - fixed labels on graph
          graph.setYLabel("OBS(h,t) per 100,000");
          int yInterval = Math.round(Math.round(maxcases/10)) ;
          graph.setEnd(set_maxt + 30, maxcases + yInterval*3);
          BufferedImage img = new BufferedImage(5,5, BufferedImage.TYPE_INT_RGB);
          Graphics2D g2 = img.createGraphics();
          g2.setColor(Color.yellow);
          g2.fill(new java.awt.geom.Ellipse2D.Float(0,0, 5, 5));
          dataLine = graph.addLine(img);  //Set the Global dataLine
          graph.setOffset(45,45);
          graph.setInterval(10,yInterval);
          graph.calibrate();
          graph.drawAxis();

          for(int i =0; i<t.length; i++) {
            if((t[i] == 0) || ((cases[i] ==0) && (i > 14))){
              break;
            } else {
              //normalize incidents per 100,000
              normalizedIncidents = cases[i]*(100000.00/pop[i]) ;
              graph.addPoint(dataLine, t[i], normalizedIncidents);
            }
          }


  // BELOW DOES THE RATIO AND VARIANCE WORK
/** A simple viewer and modifier for a data file.  A file must be in the
   ** following format, [age    deaths    population] */
   public class DataFileView2 extends JPanel {
     JScrollPane scrollPane = null;
     JTextArea dataView = new JTextArea();
     JLabel filename = new JLabel("File Name");
     int numlines = 0;
     WholeNumberField xstartbox ;
     WholeNumberField xendbox ;
     WholeNumberField ystartbox ;
     WholeNumberField yendbox ;
    // Ratio and Variance are now global variables
     ArrayList Ratio = new ArrayList();
         ArrayList Ratio_plus_dev = new ArrayList();
         ArrayList Ratio_minus_dev = new ArrayList();
     ArrayList Variance = new ArrayList();
```

```
    double normalizedIncidents1, normalizedIncidents2 ;
    double minRatio = 10000.0 ;
    double maxRatio = 0.0 ;


    DataFileView2(){

      //Initialize the Panel, we need to put the dataView text area into
      //the scroll Pane.
      setLayout(new BorderLayout());
      scrollPane = new JScrollPane(dataView);
      scrollPane.setPreferredSize(new Dimension(400,500));
      add(filename, BorderLayout.NORTH);
      add(scrollPane, BorderLayout.CENTER);
      //Create the Bottom Panel with Buttons in it
      JPanel tempPanel = new JPanel();

      JButton tempButton = new JButton("Open File");
      tempButton.setActionCommand("DataFileView2.Open");
      tempButton.addActionListener(tabbedPaneListener);
      tempPanel.add(tempButton);

      tempButton = new JButton("Save File");
      tempButton.setActionCommand("DataFileView2.Save");
      tempButton.addActionListener(tabbedPaneListener);
      tempPanel.add(tempButton);

      tempButton = new JButton("Plot Ratio");
      tempButton.setActionCommand("DataFileView2.Plot");
      tempButton.addActionListener(tabbedPaneListener);
      tempPanel.add(tempButton);


          // ADDED THIS SHOW DATA BUTTON
          tempButton = new JButton("Show Ratio Data");
          tempButton.setActionCommand("DataFileView2.Data");
          tempButton.addActionListener(tabbedPaneListener);
          tempPanel.add(tempButton);

          tempButton = new JButton("Next Step >>");
      tempButton.addActionListener(tabbedPaneListener);
      tempButton.setActionCommand("DataFileView2.Next");
      tempPanel.add(tempButton);
      tempPanel.setPreferredSize(new Dimension(400, 85)); // this is changed to fix the view of the
buttons

          add(tempPanel, BorderLayout.SOUTH);
    }

    void clearAll() {
      numlines = 0;
      dataView.setText("");
    }

    void updateAndShowFile(File file) {
```

```
 try{
   //Load the File into the window and update the next tabbed pane with
   //the Slope and alpha-beta files.


   numlines = 1;
   filename.setText(file.getName());

   BufferedReader reader = new BufferedReader(new FileReader(file));
   String line = reader.readLine();
   String fileText = new String("");

   //We are assuming that the file is in proper format.
   //Now make a huge string of one file and display it.

fileText = " Age          Incidence          Population\n" +
        "------     ----------        ----------\n" + line;
line = reader.readLine();

while(line != null) {
   numlines++;
   fileText += "\n";
   fileText += line;
   line = reader.readLine();
}
fileText += "\n"; //we need this to find whitespace after the
//last number.

//Display the file and parse the string so we can use the
//values.
dataView.setText(fileText);
dataView.setCaretPosition(0);

          StringBuffer stringBuffer = new StringBuffer(dataView.getText());
          nobs2 = 0;

          for(int i =0; i < numlines; i++){
            t2[i] = getDouble(stringBuffer);
            cases2[i] = getDouble(stringBuffer);
            pop2[i] = getDouble(stringBuffer);
            nobs2++;
          }

          // discover the offset from one data set to the next
          boolean t_greater = false;
          if (t[0] >= t2[0]){
            t_greater = true;
          }

          // FIND OFFSET
      int offset;
          if(t_greater){
            for(offset = 0; offset < t2.length; offset++){
              if(t[0] == t2[offset]){
                break;
              }
```

```
                    }
                   }
                   else{
                    for(offset = 0; offset < t.length; offset++){
                     if(t[offset] == t2[0]){
                      break;
                     }
                    }
                   }

                   int temp_len = Math.min(t.length, t2.length);
                  //Calculate Ratio, Variances for Display
                   for(int i =0; i<temp_len-offset-1; i++) { // t1.length-offset
                    if(t[i] == 0)
                     break;
          else {
                    if(t_greater == true){
                     normalizedIncidents1 = cases[i]*(100000.00/pop[i]) ;
                     normalizedIncidents2 = cases2[i+offset]*(100000.00/pop2[i+offset]) ;

                     Ratio.add(i, new Double(normalizedIncidents1/normalizedIncidents2));
                     Variance.add(i, new
Double((cases[i]*Math.pow(pop2[i+offset],2))/((Math.pow(pop[i],2))*(Math.pow(cases2[i+offset],2))) +
((Math.pow(cases[i],2))*(Math.pow(pop2[i+offset],2)))/((Math.pow(pop[i],3))*(Math.pow(cases2[i+offset]
,2))) +
((Math.pow(cases[i],2))*(Math.pow(pop2[i+offset],2)))/(Math.pow(pop[i],2)*(Math.pow(cases2[i+offset],
3))) + (Math.pow(cases[i],2)*pop2[i+offset])/(Math.pow(pop[i],2)*Math.pow(cases2[i+offset],2))));
                     Ratio_plus_dev.add(i, new Double(((Double) Ratio.get(i)).doubleValue() +
2*Math.sqrt(((Double) Variance.get(i)).doubleValue())));
                     Ratio_minus_dev.add(i, new Double(((Double) Ratio.get(i)).doubleValue() -
2*Math.sqrt(((Double) Variance.get(i)).doubleValue())));


                     if ((t[i] > 40) && (t[i] < 90)){
                      double temp = ((Double) Ratio.get(i)).doubleValue();
                      minRatio =  Math.min(minRatio, temp) ;
                      if (minRatio == temp)
                            maxRatio = minRatio ;
                      maxRatio =  Math.max(maxRatio, temp) ;
                     }
                    }
                    else{
                     normalizedIncidents1 = cases[i+offset]*(100000.00/pop[i+offset]) ;
                     normalizedIncidents2 = cases2[i]*(100000.00/pop2[i]) ;

                     Ratio.add(i, new Double(normalizedIncidents1/normalizedIncidents2));
                     Variance.add(i, new
Double((cases[i+offset]*Math.pow(pop2[i],2))/((Math.pow(pop[i+offset],2))*(Math.pow(cases2[i],2))) +
((Math.pow(cases[i+offset],2))*(Math.pow(pop2[i+offset],2)))/((Math.pow(pop[i+offset],3))*(Math.pow(c
ases2[i],2))) +
((Math.pow(cases[i+offset],2))*(Math.pow(pop2[i],2)))/((Math.pow(pop[i+offset],2))*(Math.pow(cases2[i]
,3))) + (Math.pow(cases[i+offset],2)*pop2[i])/(Math.pow(pop[i+offset],2)*Math.pow(cases2[i],2))));
                     // We will make two lines around the ratio curve- one above it by 1 std dev
                     // one below if by 1 std dev- if want to change to 2, multiply by 2 instead
                     // of 1
                     Ratio_plus_dev.add(i, new Double(((Double) Ratio.get(i)).doubleValue() +
```

```
1*Math.sqrt(((Double) Variance.get(i)).doubleValue()));
                    Ratio_minus_dev.add(i, new Double(((Double) Ratio.get(i)).doubleValue() -
1*Math.sqrt(((Double) Variance.get(i)).doubleValue()));


                    if ((t[i] > 40) && (t[i] < 90)){
                      double temp = ((Double) Ratio.get(i)).doubleValue();
                      minRatio =  Math.min(minRatio, temp) ;
                      if (minRatio == temp)
                            maxRatio = minRatio ;
                      maxRatio =  Math.max(maxRatio, temp) ;

                     }
                    }
                   }
                   }
            Ratio_global = Ratio;
            Variance_global = Variance;
            Ratio_plus_global = Ratio_plus_dev;
            Ratio_minus_global = Ratio_minus_dev;



        }catch (Exception e1) {
          e1.printStackTrace();
        }

         }



      //Creates array t, cases, and pop.  sets nobs, and plots data.  Allows
      //For next step of choosing the slope.
      public void plot() {

        //Parse the dataview to get the numbers out.

           // JUST COMMENTED THIS BELOW
           /*  StringBuffer stringBuffer = new StringBuffer(dataView.getText());
        nobs2 = 0;

        for(int i =0; i < numlines; i++){
          t2[i] = getDouble(stringBuffer);
          cases2[i] = getDouble(stringBuffer);
          pop2[i] = getDouble(stringBuffer);
          nobs2++;
        }
           */
        /****OBS2*****/
        //we know t and cases are the same length arrays, so we can
        //this in one loop.
        double normalizedIncidents ;
        //double thisAge = t2[numlines-1] ; - commented out 4/21/04
        for (int j= numlines; j <50; j++){
        //    thisAge = thisAge + 5.0;
        //    t2[j] = thisAge;
```

```
        //Initialize and plot the graph
        BufferedImage img = new BufferedImage(5,5, BufferedImage.TYPE_INT_RGB);
        Graphics2D g2 = img.createGraphics();
        g2.setColor(Color.red);
        g2.fill(new java.awt.geom.Ellipse2D.Float(0,0, 5, 5));
        int lineNum = graph.addLine(img);  //Set the Global dataLine

        for(int i =0; i<t2.length; i++) {
          if(t2[i] == 0)
            break;
          else {
            //normalize incidents per 100,000
            normalizedIncidents = cases2[i]*(100000.00/pop2[i]) ;
            graph.addPoint(lineNum, t2[i], normalizedIncidents);
          }
        }
           }
         }


public void setScale(){
        xstartBox = new WholeNumberField(0, 5) ;
        xendBox = new WholeNumberField(105, 5) ;
        ystartBox = new WholeNumberField(0, 5) ;
        yendBox = new WholeNumberField(2, 5) ;
        JButton graph = new JButton("Create Graph") ;

        JFrame setScaleFrame = new JFrame("Set Scale") ;
        JPanel temp = new JPanel() ;
        temp.setLayout(new GridLayout(8, 3));
        temp.add(new JLabel("X    ")) ;
        temp.add(createEastWestPanel(new JLabel("Start:  "), xstartBox));
        temp.add(createEastWestPanel(new JLabel("End:  "), xendBox));
        temp.add(new JLabel("Y    ")) ;
        temp.add(createEastWestPanel(new JLabel("Start:  "), ystartBox));
        temp.add(createEastWestPanel(new JLabel("End:  "), yendBox));
        //setScaleFrame.getContentPane().add(temp) ;

        //temp = new JPanel() ;
        graph.setActionCommand("SetScaleRatio.Plot");
        graph.addActionListener(tabbedPaneListener);
        temp.add(graph) ;

        setScaleFrame.getContentPane().add(temp) ;
        setScaleFrame.setSize(200, 200) ;
        setScaleFrame.setVisible(true) ;
}

 // Shows data including the Variances
 //     public void showData(){

 //     }


 //Creates a new window that displays a graph of Pobs
```

```
public void plotRatio(){

//create a popup frame
JFrame ratioFrame = new JFrame("Ratio") ;
GraphWindow ratioGraph = new GraphWindow(400, 400) ;

//Setup the GraphWindow
ratioGraph.setStart(xstartBox.getValue(), ystartBox.getValue());
ratioGraph.setXLabel("Age (years)");
ratioGraph.setYLabel("Ratio OBS1 / OBS2");
ratioGraph.setEnd(xendBox.getValue(), yendBox.getValue());
BufferedImage img = new BufferedImage(5,5, BufferedImage.TYPE_INT_RGB);
Graphics2D g2 = img.createGraphics();
g2.setColor(Color.green);
g2.fill(new java.awt.geom.Ellipse2D.Float(0,0, 5, 5));
int lineNum = ratioGraph.addLine(img);
ratioGraph.setOffset(45,45);
ratioGraph.setInterval(Math.round(Math.round(xendBox.getValue()/10)), 0.1);
ratioGraph.calibrate();
ratioGraph.drawAxis();


    // ADDED THESE TWO LINENUMS
img = new BufferedImage(5,5, BufferedImage.TYPE_INT_RGB);
g2 = img.createGraphics();
g2.setColor(Color.blue);
g2.fill(new java.awt.geom.Ellipse2D.Float(0,0, 5, 5));
int lineNum2 = ratioGraph.addLine(img);

    img = new BufferedImage(5,5, BufferedImage.TYPE_INT_RGB);
g2 = img.createGraphics();
g2.setColor(Color.blue);
g2.fill(new java.awt.geom.Ellipse2D.Float(0,0, 5, 5));
int lineNum3 = ratioGraph.addLine(img);

    boolean t_greater = false;
    if (t[0] >= t2[0]){
     t_greater = true;
    }

    int offset;
     if(t_greater){
       for(offset = 0; offset < t2.length; offset++){
         if(t[0] == t2[offset]){
               break;
         }
       }
     }
     else{
       for(offset = 0; offset < t.length; offset++){
         if(t[offset] == t2[0]){
               break;
         }
       }
     }
```

```
            int temp_len = Math.min(t.length, t2.length);

            for(int i =0; i<temp_len-offset-1; i++) { //t.length
           if(t[i] == 0)
            break;
           else {

            ratioGraph.addPoint(lineNum, t[i], ((Double) Ratio.get(i)).doubleValue());
            ratioGraph.addPoint(lineNum2, t[i], ((Double) Ratio_plus_dev.get(i)).doubleValue());
            ratioGraph.addPoint(lineNum3, t[i], ((Double) Ratio_minus_dev.get(i)).doubleValue());


           }
            }

      }
   }

/** A simple viewer and modifier for a survival data file.  A survival file must be in the
    ** following format, [age     survivalFraction] */
   public class SurvivalView extends JPanel {
      JScrollPane scrollPane = null;
      JTextArea dataView = new JTextArea();
      JLabel filename = new JLabel("File Name");
      int numlines = 0;
      WholeNumberField xstartbox ;
      WholeNumberField xendbox ;
      WholeNumberField ystartbox ;
      WholeNumberField yendbox ;
      double normalizedIncidents1, normalizedIncidents2 ;
      double minRatio = 10000.0 ;
      double maxRatio = 0.0 ;


      SurvivalView(){

         //Initialize the Panel, we need to put the dataView text area into
         //the scroll Pane.
         setLayout(new BorderLayout());
         scrollPane = new JScrollPane(dataView);
         scrollPane.setPreferredSize(new Dimension(400,500));
         add(filename, BorderLayout.NORTH);
         add(scrollPane, BorderLayout.CENTER);
         //Create the Bottom Panel with Buttons in it
         JPanel tempPanel = new JPanel();

         JButton tempButton = new JButton("Open File");
         tempButton.setActionCommand("SurvivalView.Open");
         tempButton.addActionListener(tabbedPaneListener);
         tempPanel.add(tempButton);

         tempButton = new JButton("Save File");
         tempButton.setActionCommand("SurvivalView.Save");
         tempButton.addActionListener(tabbedPaneListener);
         tempPanel.add(tempButton);
```

```
    tempButton = new JButton("Plot S(h,t)");
    tempButton.setActionCommand("SurvivalView.Plot");
    tempButton.addActionListener(tabbedPaneListener);
    tempPanel.add(tempButton);

        tempButton = new JButton("Next Step >>");
    tempButton.addActionListener(tabbedPaneListener);
    tempButton.setActionCommand("SurvivalView.Next");
    tempPanel.add(tempButton);
    tempPanel.setPreferredSize(new Dimension(400, 50));


        add(tempPanel, BorderLayout.SOUTH);
}

void clearAll() {
   numlines = 0;
   dataView.setText("");
}

void updateAndShowFile(File file) {

       try{
         //Load the File into the window and update the next tabbed pane with
         //the Slope and alpha-beta files.


         numlines = 1;
         filename.setText(file.getName());

         BufferedReader reader = new BufferedReader(new FileReader(file));
         String line = reader.readLine();
         String fileText = new String("");

         //We are assuming that the file is in proper format.
         //Now make a huge string of one file and display it.

      fileText = "  Age          Survival Fraction\n" +
            " -------      -------------------------\n" + line;
      line = reader.readLine();

      while(line != null) {
         numlines++;
         fileText += "\n";
         fileText += line;
         line = reader.readLine();
      }
      fileText += "\n"; //we need this to find whitespace after the
      //last number.

      //Display the file and parse the string so we can use the
      //values.
      dataView.setText(fileText);
      dataView.setCaretPosition(0);
```

```
            StringBuffer stringBuffer = new StringBuffer(dataView.getText());

            // Changes to the code below
            for(int i =0; i < numlines; i++){
             survival_age[i] = getDouble(stringBuffer);
             survivalFraction[i] = getDouble(stringBuffer);
             // need to fix offset problem here
             for(int ii = 0 ; ii < t.length; ii++){
              if(survival_age[i] == t[ii])
               cases_adj[ii] = cases_adj[ii]/(1-survivalFraction[i]); // adjust for survival data
             }
            }


  }catch (Exception e1) {
    e1.printStackTrace();
  }

   }


//Creates array t, cases, and pop.  sets nobs, and plots data.  Allows
//For next step of choosing the slope.
public void plot() {

    JFrame survivalFrame = new JFrame("Survival Data") ;
    GraphWindow survivalGraph = new GraphWindow(400, 400) ;
    survivalGraph.setStart(xstartBox.getValue(), ystartBox.getValue());
    survivalGraph.setXLabel("Age (years)");
    survivalGraph.setYLabel("S(h,t)");
    survivalGraph.setEnd(xendBox.getValue(), yendBox.getValue());

    BufferedImage img = new BufferedImage(5,5, BufferedImage.TYPE_INT_RGB);
    Graphics2D g2 = img.createGraphics();
    g2.setColor(Color.red);
    g2.fill(new java.awt.geom.Ellipse2D.Float(0,0, 5, 5));
    int lineNum = survivalGraph.addLine(img);  //Set the Global dataLine
    survivalGraph.setOffset(45,45);
    survivalGraph.setInterval(Math.round(Math.round(xendBox.getValue()/10)), 0.1);
    survivalGraph.calibrate();
    survivalGraph.drawAxis();
    for(int i =0; i<survival_age.length; i++) {
     if(survival_age[i] == 0)
       break;
     else {

       survivalGraph.addPoint(lineNum, survival_age[i], survivalFraction[i]);
     }
    }
    survivalFrame.getContentPane().add(survivalGraph);
    survivalFrame.setSize(450,450);
    survivalFrame.setVisible(true);

   }
```

```
    public void setScale(){
        xstartBox = new WholeNumberField(0, 5) ;
        xendBox = new WholeNumberField(105, 5) ;
        ystartBox = new WholeNumberField(0, 5) ;
        yendBox = new WholeNumberField(1, 5) ;
        JButton graph = new JButton("Create Graph") ;

        JFrame setScaleFrame = new JFrame("Set Scale") ;
        JPanel temp = new JPanel() ;
        temp.setLayout(new GridLayout(8, 3));
        temp.add(new JLabel("X    ")) ;
        temp.add(createEastWestPanel(new JLabel("Start:  "), xstartBox));
        temp.add(createEastWestPanel(new JLabel("End:  "), xendBox));
        temp.add(new JLabel("Y    ")) ;
        temp.add(createEastWestPanel(new JLabel("Start:  "), ystartBox));
        temp.add(createEastWestPanel(new JLabel("End:  "), yendBox));
        //setScaleFrame.getContentPane().add(temp) ;

        //temp = new JPanel() ;
        graph.setActionCommand("SurvivalViewData.Plot");
        graph.addActionListener(tabbedPaneListener);
        temp.add(graph) ;

        setScaleFrame.getContentPane().add(temp) ;
        setScaleFrame.setSize(200, 200) ;
        setScaleFrame.setVisible(true) ;
    }


    }


/** A simple viewer and modifier for a survival data file.  A survival file must be in the
    ** following format, [age     survivalFraction] */
    public class ErrorView extends JPanel {
      JScrollPane scrollPane = null;
      JTextArea dataView = new JTextArea();
      JLabel filename = new JLabel("File Name");
      int numlines = 0;
      WholeNumberField xstartbox ;
      WholeNumberField xendbox ;
      WholeNumberField ystartbox ;
      WholeNumberField yendbox ;
      double normalizedIncidents1, normalizedIncidents2 ;
      double minRatio = 10000.0 ;
      double maxRatio = 0.0 ;


      ErrorView(){

        //Initialize the Panel, we need to put the dataView text area into
        //the scroll Pane.
        setLayout(new BorderLayout());
        scrollPane = new JScrollPane(dataView);
        scrollPane.setPreferredSize(new Dimension(400,500));
```

```
    add(filename, BorderLayout.NORTH);
    add(scrollPane, BorderLayout.CENTER);
    //Create the Bottom Panel with Buttons in it
    JPanel tempPanel = new JPanel();

    JButton tempButton = new JButton("Open File");
    tempButton.setActionCommand("ErrorView.Open");
    tempButton.addActionListener(tabbedPaneListener);
    tempPanel.add(tempButton);

    tempButton = new JButton("Save File");
    tempButton.setActionCommand("ErrorView.Save");
    tempButton.addActionListener(tabbedPaneListener);
    tempPanel.add(tempButton);

    tempButton = new JButton("Plot R(h,t)");
    tempButton.setActionCommand("ErrorView.Plot");
    tempButton.addActionListener(tabbedPaneListener);
    tempPanel.add(tempButton);

        tempButton = new JButton("Next Step >>");
    tempButton.addActionListener(tabbedPaneListener);
    tempButton.setActionCommand("ErrorView.Next");
    tempPanel.add(tempButton);
    tempPanel.setPreferredSize(new Dimension(400, 50));


        add(tempPanel, BorderLayout.SOUTH);
}

void clearAll() {
    numlines = 0;
    dataView.setText("");
}

void updateAndShowFile(File file) {

        try{
          //Load the File into the window and update the next tabbed pane with
          //the Slope and alpha-beta files.


          numlines = 1;
          filename.setText(file.getName());

          BufferedReader reader = new BufferedReader(new FileReader(file));
          String line = reader.readLine();
          String fileText = new String("");

          //We are assuming that the file is in proper format.
          //Now make a huge string of one file and display it.

      fileText = " Birth Year                Reporting Error (%)\n" +
              "----------------          ----------------------\n" + line;
      line = reader.readLine();
```

```
    while(line != null) {
      numlines++;
      fileText += "\n";
      fileText += line;
      line = reader.readLine();
    }
    fileText += "\n"; //we need this to find whitespace after the
    //last number.

    //Display the file and parse the string so we can use the
    //values.
    dataView.setText(fileText);
    dataView.setCaretPosition(0);

            StringBuffer stringBuffer = new StringBuffer(dataView.getText());

            // Changes to the code below
            for(int i =0; i < numlines; i++){
             report_error_year[i] = getDouble(stringBuffer);
             reportError[i] = getDouble(stringBuffer);
             // need to fix the offset problem: here is solution
              for(int ii = 0 ; ii < t.length; ii++){
               if(report_error_year[i] == t[ii])
                 cases_adj[ii] = cases_adj[ii]/reportError[i]; // adjust for Reporting data
              }
            }


  }catch (Exception e1) {
    e1.printStackTrace();
  }

   }



//Creates array t, cases, and pop.  sets nobs, and plots data.  Allows
//For next step of choosing the slope.
public void plot() {

  /****OBS2*****/
  //we know t and cases are the same length arrays, so we can
  //this in one loop.
  //double normalizedIncidents ;
  //double thisAge = t2[numlines-1] ;
  //for (int j= numlines; j <50; j++){
  //   thisAge = thisAge + 5.0;
  //   t2[j] = thisAge;

  //Initialize and plot the graph

    JFrame errorFrame = new JFrame("Error Data") ;
    GraphWindow errorGraph = new GraphWindow(400, 400) ;
    errorGraph.setStart(xstartBox.getValue(), ystartBox.getValue());
    errorGraph.setXLabel("Age (years)");
    errorGraph.setYLabel("R(h,t)");
```

```
            errorGraph.setEnd(xendBox.getValue(), yendBox.getValue());
            BufferedImage img = new BufferedImage(5,5, BufferedImage.TYPE_INT_RGB);
            Graphics2D g2 = img.createGraphics();
            g2.setColor(Color.red);
            g2.fill(new java.awt.geom.Ellipse2D.Float(0,0, 5, 5));
            int lineNum = errorGraph.addLine(img);  //Set the Global dataLine
            errorGraph.setOffset(45,45);
            errorGraph.setInterval(Math.round(Math.round(xendBox.getValue()/10)), 0.1);
            errorGraph.calibrate();
            errorGraph.drawAxis();
            for(int i =0; i<report_error_year.length; i++) {
              if(report_error_year[i] == 0)
                break;
              else {
                errorGraph.addPoint(lineNum, report_error_year[i], reportError[i]);
              }
            }
            errorFrame.getContentPane().add(errorGraph);
            errorFrame.setSize(450,450);
            errorFrame.setVisible(true);

          }


    public void setScale(){
          xstartBox = new WholeNumberField(0, 5) ;
          xendBox = new WholeNumberField(105, 5) ;
          ystartBox = new WholeNumberField(0, 5) ;
          yendBox = new WholeNumberField(1, 5) ;
          JButton graph = new JButton("Create Graph") ;

          JFrame setScaleFrame = new JFrame("Set Scale") ;
          JPanel temp = new JPanel() ;
          temp.setLayout(new GridLayout(8, 3));
          temp.add(new JLabel("X    ")) ;
          temp.add(createEastWestPanel(new JLabel("Start:  "), xstartBox));
          temp.add(createEastWestPanel(new JLabel("End:  "), xendBox));
          temp.add(new JLabel("Y    ")) ;
          temp.add(createEastWestPanel(new JLabel("Start:  "), ystartBox));
          temp.add(createEastWestPanel(new JLabel("End:  "), yendBox));
          //setScaleFrame.getContentPane().add(temp) ;

          //temp = new JPanel() ;
          graph.setActionCommand("ErrorPlotData.Plot");
          graph.addActionListener(tabbedPaneListener);
          temp.add(graph) ;

          setScaleFrame.getContentPane().add(temp) ;
          setScaleFrame.setSize(200, 200) ;
          setScaleFrame.setVisible(true) ;
    }
    }

/** A simple viewer and modifier for a survival data file.  A survival file must be in the
   ** following format, [age     survivalFraction] */
    public class TOTView extends JPanel {
```

```
JScrollPane scrollPane = null;
JTextArea dataView = new JTextArea();
JLabel filename = new JLabel("File Name");
int numlines = 0;
WholeNumberField xstartbox ;
WholeNumberField xendbox ;
WholeNumberField ystartbox ;
WholeNumberField yendbox ;
double normalizedIncidents1, normalizedIncidents2 ;
double minRatio = 10000.0 ;
double maxRatio = 0.0 ;


TOTView(){

   //Initialize the Panel, we need to put the dataView text area into
   //the scroll Pane.
   setLayout(new BorderLayout());
   scrollPane = new JScrollPane(dataView);
   scrollPane.setPreferredSize(new Dimension(400,500));
   add(filename, BorderLayout.NORTH);
   add(scrollPane, BorderLayout.CENTER);
   //Create the Bottom Panel with Buttons in it
   JPanel tempPanel = new JPanel();

   JButton tempButton = new JButton("Open File");
   tempButton.setActionCommand("TOTView.Open");
   tempButton.addActionListener(tabbedPaneListener);
   tempPanel.add(tempButton);

   tempButton = new JButton("Save File");
   tempButton.setActionCommand("TOTView.Save");
   tempButton.addActionListener(tabbedPaneListener);
   tempPanel.add(tempButton);

   tempButton = new JButton("Plot TOT(h,t)");
   tempButton.setActionCommand("TOTView.Plot");
   tempButton.addActionListener(tabbedPaneListener);
   tempPanel.add(tempButton);

       tempButton = new JButton("Next Step >>");
   tempButton.addActionListener(tabbedPaneListener);
   tempButton.setActionCommand("TOTView.Next");
   tempPanel.add(tempButton);
   tempPanel.setPreferredSize(new Dimension(400, 50));


       add(tempPanel, BorderLayout.SOUTH);
}

void clearAll() {
   numlines = 0;
   dataView.setText("");
}

void updateAndShowFile(File file) {
```

```
    try{
      //Load the File into the window and update the next tabbed pane with
      //the Slope and alpha-beta files.


      numlines = 1;
      filename.setText(file.getName());

      BufferedReader reader = new BufferedReader(new FileReader(file));
      String line = reader.readLine();
      String fileText = new String("");

      //We are assuming that the file is in proper format.
      //Now make a huge string of one file and display it.

    fileText = " Age          Mortality                    Population  \n" +
           "------      ----------                  ----------\n" + line;
    line = reader.readLine();

    while(line != null) {
      numlines++;
      fileText += "\n";
      fileText += line;
      line = reader.readLine();
    }
    fileText += "\n"; //we need this to find whitespace after the
    //last number.

    //Display the file and parse the string so we can use the
    //values.
    dataView.setText(fileText);
    dataView.setCaretPosition(0);

            StringBuffer stringBuffer = new StringBuffer(dataView.getText());

            // Changes to the code below
            for(int i =0; i < numlines; i++){
              tot_age[i] = getDouble(stringBuffer);
              tot[i] = getDouble(stringBuffer);
              tot_pop[i] = getDouble(stringBuffer);
              // now set up what tot should actually be for OBS* calculation
              tot[i] = tot[i]/tot_pop[i];
              // fixed offset problem with this for loop...
              for(int ii = 0 ; ii < t.length; ii++){
                if(tot_age[i] == t[ii])
                  cases_adj[ii] = cases_adj[ii]/(1-tot[i]); // adjust cases for TOT data
              }

            }

}catch (Exception e1) {
  e1.printStackTrace();
}

  }
```

```
//Creates array t, cases, and pop.  sets nobs, and plots data.  Allows
//For next step of choosing the slope.
public void plot() {
  //Initialize and plot the graph

      JFrame totFrame = new JFrame("TOT Data") ;
      GraphWindow totGraph = new GraphWindow(400, 400) ;
      totGraph.setStart(xstartBox.getValue(), ystartBox.getValue());
      totGraph.setXLabel("Age (years)");
      totGraph.setYLabel("TOT(h,t)");
      totGraph.setEnd(xendBox.getValue(), yendBox.getValue());
      BufferedImage img = new BufferedImage(5,5, BufferedImage.TYPE_INT_RGB);
 Graphics2D g2 = img.createGraphics();
      g2.setColor(Color.red);
      g2.fill(new java.awt.geom.Ellipse2D.Float(0,0, 5, 5));
      int lineNum = totGraph.addLine(img);  //Set the Global dataLine
      totGraph.setOffset(45,45);
      totGraph.setInterval(Math.round(Math.round(xendBox.getValue()/10)), 0.1);
      totGraph.calibrate();
      totGraph.drawAxis();
   for(int i =0; i<tot_age.length; i++) {
     if(tot_age[i] == 0)
       break;
     else {
      totGraph.addPoint(lineNum, tot_age[i], tot[i]);
     }

      }
      totFrame.getContentPane().add(totGraph);
      totFrame.setSize(450,450);
      totFrame.setVisible(true);
      //}
    }


public void setScale(){
    xstartBox = new WholeNumberField(0, 5) ;
    xendBox = new WholeNumberField(105, 5) ;
    ystartBox = new WholeNumberField(0, 5) ;
    yendBox = new WholeNumberField(1, 5) ;
    JButton graph = new JButton("Create Graph") ;

    JFrame setScaleFrame = new JFrame("Set Scale") ;
    JPanel temp = new JPanel() ;
    temp.setLayout(new GridLayout(8, 3));
    temp.add(new JLabel("X    ")) ;
    temp.add(createEastWestPanel(new JLabel("Start:  "), xstartBox));
    temp.add(createEastWestPanel(new JLabel("End:  "), xendBox));
    temp.add(new JLabel("Y    ")) ;
    temp.add(createEastWestPanel(new JLabel("Start:  "), ystartBox));
    temp.add(createEastWestPanel(new JLabel("End:  "), yendBox));

    graph.setActionCommand("SetScaleTOT.Plot");
```

```
        graph.addActionListener(tabbedPaneListener);
        temp.add(graph) ;
        setScaleFrame.getContentPane().add(temp) ;
        setScaleFrame.setSize(200, 200) ;
        setScaleFrame.setVisible(true) ;
    }



    }


    /** Allows the user to see the numbers behind the fits */

    class DataWindow extends JPanel {
      JPanel directions = new JPanel();
      JTable entryView = null;
      EntryTable entryTable = new EntryTable();

      DataWindow() {
            entryView = new JTable(entryTable);
            entryView.setPreferredScrollableViewportSize(new Dimension(300, 335));
            JScrollPane scrollPne = new JScrollPane(entryView);
            this.add(scrollPne);
      }

       //Clears the table and the other data
       void clearAll() {
          entryTable.rowCount = 0;
          entryTable.data = null;
       }


       //A to show the various values and allow the user to pick two
       public class EntryTable extends AbstractTableModel {
          Vector columnNames = new Vector();
          int rowCount = 0;
          Object[][] data;

          public EntryTable() {
             columnNames.add("Age");
             columnNames.add("Value");
          }

          public void updateTable(double[] age, double[] values) {
             data = new Object[nobs+extension][2];
             int i = 0 ;
             //double area = 0.0 ;
             while(i < nobs+extension){
                data[i][0] = String.valueOf(t[i]);
                data[i][1] = truncString(String.valueOf(Pobs[i] * Frac / (Frac + ((double)1 - Frac) *
Math.exp((1-survivalFraction[i])*intPobs[i] / f))), 4);
                //area = area + Double.parseDouble((String)data[i][1]) ;
                i++;
             }
```

```
        //System.out.println("area is:  " + area);
        rowCount = i;
        fireTableDataChanged();


    }

    //Returns the first two selected entries.
    public Point getSelIndex() {
        int tmp1 = -1;
        int tmp2 = -1;

        for(int i = 0; i < nobs; i++){
            if (data[i][2].equals(new Boolean(true))){
                if(tmp1 == -1) {
                    tmp1 = i;
                } else {
                    tmp2 = i;
                    break;
                }
            }
        }
        return new Point(tmp1, tmp2);
    }


    public int getColumnCount() {
        return columnNames.size();
    }

    public int getRowCount() {
        return rowCount;
    }

    public String getColumnName(int col) {
        return (String)columnNames.get(col);
    }

    public Object getValueAt(int row, int col) {
        return data[row][col];
    }

/*
 * JTable uses this method to determine the default renderer/
 * editor for each cell.  If we didn't implement this method,
 * then the last column would contain text ("true"/"false"),
 * rather than a check box.
 */
    public Class getColumnClass(int c) {
      return getValueAt(0, c).getClass();
    }



    public void setValueAt(Object value, int row, int col) {
        data[row][col] = (Boolean)value;
```

```
            Boolean val = (Boolean)value;

            fireTableCellUpdated(row, col);
        }

        //Only the booleans are editable
        public boolean isCellEditable(int row, int col) {
            if(col == 2) {
                return true;
            } else
                return false;
        }
    }
}


// Variances, etc
// ADDED 5/1/03
class DataWindow2 extends JPanel {
    JPanel directions = new JPanel();
    EntryTable entryTable = new EntryTable();
    JTable entryView = null;


    DataWindow2() {
        entryView = new JTable(entryTable);
        entryView.setPreferredScrollableViewportSize(new Dimension(300, 335));
        JScrollPane scrollPne = new JScrollPane(entryView);
        this.add(scrollPne);
    }

     //Clears the table and the other data
     void clearAll() {
        entryTable.rowCount = 0;
        entryTable.data = null;
     }


// ADDED THIS FOR DATAVIEWFILE2 SHOW
    public class EntryTable extends AbstractTableModel {
        Vector columnNames = new Vector();
        int rowCount = 0;
        Object[][] data;

        public EntryTable() {
            columnNames.add("Age");
            columnNames.add("Ratio (OBS1/OBS2)");
                    columnNames.add("Std Dev");
        }

        public void updateTable(double[] age, double[] values) {
            data = new Object[nobs+extension][3];
            int i = 0 ;
            //double area = 0.0 ;
            while(i < nobs+extension && t[i] < 106){
                data[i][0] = String.valueOf(t[i]);
```

```
            data[i][1] = truncString(String.valueOf(((Double)Ratio_global.get(i)).doubleValue()), 3);
                    data[i][2] =
truncString(String.valueOf(Math.sqrt(((Double)Variance_global.get(i)).doubleValue())), 3);
            i++;
          }
          //System.out.println("area is:  " + area);
          rowCount = i;
          fireTableDataChanged();

        }

        //Returns the first two selected entries.
        public Point getSelIndex() {
          int tmp1 = -1;
          int tmp2 = -1;

          for(int i = 0; i < nobs; i++){
            if (data[i][2].equals(new Boolean(true))){
              if(tmp1 == -1) {
                tmp1 = i;
              } else {
                tmp2 = i;
                break;
              }
            }
          }
          return new Point(tmp1, tmp2);
        }


        public int getColumnCount() {
          return columnNames.size();
        }

        public int getRowCount() {
          return rowCount;
        }

        public String getColumnName(int col) {
          return (String)columnNames.get(col);
        }

        public Object getValueAt(int row, int col) {
          return data[row][col];
        }

       /*
        * JTable uses this method to determine the default renderer/
        * editor for each cell.  If we didn't implement this method,
        * then the last column would contain text ("true"/"false"),
        * rather than a check box.
        */
        public Class getColumnClass(int c) {
          return getValueAt(0, c).getClass();
        }
```

```
        public void setValueAt(Object value, int row, int col) {
          data[row][col] = (Boolean)value;
          Boolean val = (Boolean)value;

          fireTableCellUpdated(row, col);
        }

        //Only the booleans are editable
        public boolean isCellEditable(int row, int col) {
          if(col == 2) {
            return true;
          } else
            return false;
        }
      }
    }


    /** This is the second step in the data processing process.  Takes in the data
     * from the first window, and allows the user to select two point to use to
     * calculate alpha - beta. */

    class EstimateWindow extends JPanel {
      JPanel directions = new JPanel();
      JTable entryView = null;
      EntryTable entryTable = new EntryTable();
      JButton next = new JButton("Next >>");
      JButton estimate = new JButton("Calculate Estimate");
      JButton previous = new JButton("<< Previous");
      DoubleNumberField aminusb = new DoubleNumberField(0, 5);
      DoubleNumberField maxDeta = new DoubleNumberField(2, 5);
      DoubleNumberField maxSlope = new DoubleNumberField(0, 5);

      int checkcount = 0;

      EstimateWindow() {

        this.setLayout(new BoxLayout(this, BoxLayout.Y_AXIS));
        JLabel label4 = new JLabel("Here are the current bounds on the parameters.");
        JLabel label5 = new JLabel("If you would like to change the bounds, enter them in the appropriate
columns.");
        JLabel label6 = new JLabel("If you would like to hold a parameter constant,");
        JLabel label7 = new JLabel("change the value in the first column, and check the corresponding
box.");
        JLabel label51 = new JLabel("If you wish to examine the case where alpha - beta varies with
time,");
        JLabel label52 = new JLabel("you must check the boxes next to Mod A - B and Age.");
        directions.setLayout(new BoxLayout(directions, BoxLayout.Y_AXIS));
        directions.add(label5);
        directions.add(label6);
        directions.add(label7);
        directions.add(label51);
        directions.add(label52);
```

```
    //have to use the tmp jpanel because I could not get the layout to
    //properly display the directions otherwise.

    JPanel tmp = new JPanel();
    tmp.add(directions);
    //this.add(tmp);
    entryView = new JTable(entryTable);
    entryView.setPreferredScrollableViewportSize(new Dimension(300, 335));
    JScrollPane scrollPne = new JScrollPane(entryView);
    this.add(scrollPne);

    //Add in the text fields that display the values
    JPanel temp = new JPanel();
    temp.setLayout(new GridLayout(1,3));
    //temp.add(createEastWestPanel(new JLabel("Max Slope: "), maxSlope));
    //temp.add(createEastWestPanel(new JLabel("Max Delta: "), maxDeta));
    //temp.add(createEastWestPanel(new JLabel("A - B"), aminusb));
    this.add(temp);

    temp = new JPanel();
    next.addActionListener(tabbedPaneListener);
    previous.addActionListener(tabbedPaneListener);
    //estimate.addActionListener(tabbedPaneListener);


    next.setActionCommand("EstimateWindow.Next");
    previous.setActionCommand("EstimateWindow.Previous");
    //estimate.setActionCommand("EstimateWindow.Estimate");
    temp.add(previous);
    //temp.add(estimate);
    temp.add(next);
    this.add(temp);
    estimate.setEnabled(false);
}

//Clears the table and the other data
void clearAll() {
    maxSlope.setText("");
    maxDeta.setText("");
    entryTable.rowCount = 0;
    entryTable.data = null;
}

/** Updates the Slope field in the estimateWindow using the values
 * passed in where pnt.getX is hte first age and pnt.getY is the
 * second age */
void updateSlope(Point pnt) {

    double val = getAgeIntervals((int)pnt.getX(),(int)pnt.getY(), t, obs);
    aminusb.setText(truncString(String.valueOf(val), 4));
}

//A to show the various values and allow the user to pick two
public class EntryTable extends AbstractTableModel {
    Vector columnNames = new Vector();
    int rowCount = 0;
```

```
Object[][] data;

public EntryTable() {
   columnNames.add("Starting Age");
   columnNames.add("Ending Age");
   columnNames.add("Alpha - Beta");
   //columnNames.add("Slope");
   //columnNames.add("Delta");
   columnNames.add("Select");
}

public void updateTable(double[] slp, double[] slope, double[] deta) {
   data = new Object[nobs][6];
   for(int i = 0; i < nobs; i++){
      data[i][0] = String.valueOf(t[i]);
      data[i][1] = String.valueOf(t[i+1]);
      data[i][2] = truncString(String.valueOf((loggedArray[i + 1] - loggedArray[i]) / 5), 3) ;
      //data[i][2] = truncString(String.valueOf(slp[i] / Math.log(2.0)), 3);
      data[i][4] = truncString(String.valueOf(slope[i] * Math.pow(10, 5)), 4);
      data[i][5] = truncString(String.valueOf(deta[i]), 2);
      data[i][3] = new Boolean(false);
   }
   rowCount = nobs - 1;
   fireTableDataChanged();


}

//Returns the first two selected entries.
public Point getSelIndex() {
   int tmp1 = 0;
   int tmp2 = 0;

   for(int i = 0; i < nobs; i++){
      if (data[i][3].equals(new Boolean(true))){
         if(tmp1 == 0) {
            tmp1 = i;
         } else {
            tmp2 = i;
            break;
         }
      }
   }
   return new Point(tmp1, tmp2);
}


public int getColumnCount() {
   return columnNames.size();
}

public int getRowCount() {
   return rowCount;
}

public String getColumnName(int col) {
```

```
        return (String)columnNames.get(col);
      }

      public Object getValueAt(int row, int col) {
        return data[row][col];
      }

    /*
     * JTable uses this method to determine the default renderer/
     * editor for each cell.  If we didn't implement this method,
     * then the last column would contain text ("true"/"false"),
     * rather than a check box.
     */
      public Class getColumnClass(int c) {
        return getValueAt(0, c).getClass();
      }



      public void setValueAt(Object value, int row, int col) {
        data[row][col] = (Boolean)value;
        Boolean val = (Boolean)value;
        if(val.equals(new Boolean(false))){
          checkcount--;
        } else if(val.equals(new Boolean(true))){
          checkcount++;
        }

        //Test to see that there are only two selected and enable
        //the user to move foward
        if(checkcount == 2) {
          estimate.setEnabled(true);
        } else {
          estimate.setEnabled(false);
        }

        fireTableCellUpdated(row, col);
      }

      //Only the booleans are editable
      public boolean isCellEditable(int row, int col) {
        if(col ==3) {
          return true;
        } else
          return false;
      }

    }


}

/** Allows the user to change the parameters that the equations use
 * to calculate teh best fits. */
class SetParametersWindow extends JPanel{
   int textBoxSize = 6;
```

```
    JLabel growthLabel1 = new JLabel("Input the following three parameter") ;
    JLabel growthLabel2 = new JLabel("to uniquely define the growth of cells");
    JLabel growthLabel3 = new JLabel("at risk as a function of age") ;
    DoubleNumberField gammaBox = new DoubleNumberField(1.0, textBoxSize);
    DoubleNumberField nMaxBox = new DoubleNumberField(8.5*Math.pow(10,10), textBoxSize);
    WholeNumberField initiateableAgeBox = new WholeNumberField(17, textBoxSize);
   // to change Initiable Age to be a half number, could change above line to DoubleNumberField
    JCheckBox maleCheckBox = new JCheckBox() ;
    JCheckBox survivalCheckBox = new JCheckBox();
    JCheckBox reportErrorCheckBox = new JCheckBox();

    //Allowing N and M to be varied
    JLabel nLabel1 = new JLabel("Input the desired value of n") ;
    JLabel nLabel2 = new JLabel("(the number of initiation events)") ;
    DoubleNumberField nNum = new DoubleNumberField(ninit, 3);
    JLabel mLabel1 = new JLabel("Input the desired value of m") ;
    JLabel mLabel2 = new JLabel("(the number of promotion events)") ;
    DoubleNumberField mNum = new DoubleNumberField(minit , 3);
    JLabel sLabel1 = new JLabel("Input the desired value of S") ;
    JLabel sLabel2 = new JLabel("(the survival fraction)") ;
    DoubleNumberField survivalNum = new DoubleNumberField(survivalDefault , 3);

  // need to create some reminder to do this/check to make sure this was done
    JButton setupNArray = new JButton("Setup the Growth Function of Cells at Risk") ;
    JButton next = new JButton("Next >>") ;

    Vector boxList = new Vector();

    JPanel bottomDisplay = new JPanel();

    SetParametersWindow(){
      boxList.add(gammaBox);
      boxList.add(nMaxBox);
      boxList.add(initiateableAgeBox);

      this.setLayout(new GridLayout(3, 1));

      //Create the layout to enter in all of the parameters.
      JPanel temp = new JPanel();
      temp.setLayout(new GridLayout(9, 1));
      temp.add(growthLabel1) ;
      temp.add(growthLabel2) ;
      temp.add(growthLabel3) ;
      temp.add(createEastWestPanel(new JLabel("Gamma: (Correction value for Topology)"),
gammaBox));
      temp.add(createEastWestPanel(new JLabel("NMax: (Maximum Number of Cells at Risk)"),
nMaxBox));
      temp.add(createEastWestPanel(new JLabel("Initiatable Age: (Age through which Initiation
Occurs)"), initiateableAgeBox));
      temp.add(createEastWestPanel(new JLabel("Check if sex is Male"), maleCheckBox)) ;
      setupNArray.setActionCommand("SetParametersWindow.setupNArray");
      setupNArray.addActionListener(tabbedPaneListener);
      temp.add(setupNArray) ;
      this.add(temp);
```

```
        //Create area for the user to input the values of N and M
        temp = new JPanel() ;
        temp.setLayout(new GridLayout(11, 3));
        temp.add(nLabel1) ;
        temp.add(nLabel2) ;
        temp.add(createEastWestPanel(new JLabel("n: "), nNum));
        temp.add(mLabel1) ;
        temp.add(mLabel2) ;
        temp.add(createEastWestPanel(new JLabel("m: "), mNum));
        temp.add(sLabel1) ;
        temp.add(sLabel2) ;
        temp.add(createEastWestPanel(new JLabel("S: "), survivalNum));
            //          this.add(temp) ;

            //          temp = new JPanel();
            //          temp.setLayout(new GridLayout(2,2));
            temp.add(createEastWestPanel(new JLabel("Check if you want to enter Survival Data File"),
survivalCheckBox));
            temp.add(createEastWestPanel(new JLabel("Check if you want to enter Reporting Error Data
File"), reportErrorCheckBox));
            this.add(temp);



        temp = new JPanel();
        next.setActionCommand("SetParametersWindow.next");
        next.addActionListener(tabbedPaneListener);
        temp.add(next) ;
        this.add(temp);
    }

}


/** Allows the user to change the boundaries and calculates a best fit
 * to the data using the set bounds. */
class BestFitWindow extends JPanel{
    int textBoxSize = 6;
  // add some sort of plot to plot OBS**? 4/21/04
    //This is just a ton of fields that we need references to to get the
    //values out of.
    DoubleNumberField FrlBox = new DoubleNumberField(Frl, textBoxSize);
    DoubleNumberField FruBox = new DoubleNumberField(Fru, textBoxSize);
    DoubleNumberField flBox = new DoubleNumberField(fl, textBoxSize);
    DoubleNumberField fuBox = new DoubleNumberField(fu, textBoxSize);
    DoubleNumberField clBox = new DoubleNumberField(cl, textBoxSize);
    DoubleNumberField cuBox = new DoubleNumberField(cu, textBoxSize);
    DoubleNumberField alminbelBox = new DoubleNumberField(alminbel, textBoxSize);
    DoubleNumberField alminbeuBox = new DoubleNumberField(alminbeu, textBoxSize);
    DoubleNumberField betalBox = new DoubleNumberField(betal, textBoxSize);
    DoubleNumberField betauBox = new DoubleNumberField(betau, textBoxSize);
    DoubleNumberField rAlBox = new DoubleNumberField(rAl, textBoxSize);
    DoubleNumberField rAuBox = new DoubleNumberField(rAu, textBoxSize);
    DoubleNumberField alphaBox = new DoubleNumberField(alpha, textBoxSize);
    DoubleNumberField ageWeightlBox = new DoubleNumberField(weightsAgel, textBoxSize);
```

```
DoubleNumberField ageWeightuBox = new DoubleNumberField(weightsAgeu, textBoxSize);

WholeNumberField Frnum = new WholeNumberField(nloop[0], 3);
WholeNumberField fnum = new WholeNumberField(nloop[1], 3);
WholeNumberField cnum = new WholeNumberField(nloop[2], 3);
WholeNumberField alminbenum = new WholeNumberField(nloop[3], 3);
WholeNumberField betanum = new WholeNumberField(nloop[4], 3);
WholeNumberField rAnum = new WholeNumberField(nloop[5], 3);

Vector boxList = new Vector();

JCheckBox FrCheckBox = new JCheckBox();
JCheckBox fCheckBox = new JCheckBox();
JCheckBox cCheckBox = new JCheckBox();
JCheckBox alminbeCheckBox = new JCheckBox();
JCheckBox betaCheckBox = new JCheckBox();
JCheckBox rACheckBox = new JCheckBox();
JCheckBox ageBox = new JCheckBox();
JCheckBox rssBox = new JCheckBox();

JButton findFit = new JButton("Find Fits");
JButton findBestFit = new JButton("Find Single Best Fit");
JButton cancel = new JButton("Cancel");
JProgressBar progressBar = new JProgressBar();

JPanel bottomDisplay = new JPanel();

String filename = "TempData.txt";

BestFitWindow(){
   boxList.add(FruBox);
   boxList.add(FrlBox);
   boxList.add(fuBox);
   boxList.add(flBox);
   boxList.add(clBox);
   boxList.add(cuBox);
   boxList.add(alminbelBox);
   boxList.add(alminbeuBox);
   boxList.add(betalBox);
   boxList.add(betauBox);
   boxList.add(rAlBox);
   boxList.add(rAuBox);
   boxList.add(alphaBox);
   boxList.add(ageWeightlBox);
   boxList.add(ageWeightuBox);

   //Create the layout to enter in all of the parameters.
   JPanel temp = new JPanel();
   JPanel temp2 = new JPanel();
   JPanel temp3 = new JPanel();
   //this.setLayout(new GridLayout(6, 3)) ;
   temp.setLayout(new GridLayout(9, 3));
   JLabel boundLabel = new JLabel("Bounds");
   temp3.add(boundLabel) ;
   //this.add(temp3) ;
   //temp.add(temp2) ;
```

```
temp2 = new JPanel() ;
JLabel fixed = new JLabel("Fixed: ");
temp2.add(new JLabel("F                    "));
temp2.add(FrCheckBox);
temp.add(temp2);
temp.add(createEastWestPanel(new JLabel("Lower: "), FrlBox));
temp.add(createEastWestPanel(new JLabel("Upper: "), FruBox));
temp2 = new JPanel();
temp2.add(new JLabel("f                    "));
temp2.add(fCheckBox);
temp.add(temp2);
temp.add(createEastWestPanel(new JLabel("Lower: "), flBox));
temp.add(createEastWestPanel(new JLabel("Upper: "), fuBox));
temp2 = new JPanel();
temp2.add(new JLabel("cInit              "));
temp2.add(cCheckBox);
temp.add(temp2);
temp.add(createEastWestPanel(new JLabel("Lower: "), clBox));
temp.add(createEastWestPanel(new JLabel("Upper: "), cuBox));
temp2 = new JPanel();
temp2.add(new JLabel("Alpha - Beta  "));
temp2.add(alminbeCheckBox);
temp.add(temp2);
temp.add(createEastWestPanel(new JLabel("Lower: "), alminbelBox));
temp.add(createEastWestPanel(new JLabel("Upper: "), alminbeuBox));
temp2 = new JPanel();
temp2.add(new JLabel(" cProm           "));
temp2.add(rACheckBox);
temp.add(temp2);
temp.add(createEastWestPanel(new JLabel("Lower: "), rAlBox));
temp.add(createEastWestPanel(new JLabel("Upper: "), rAuBox));
temp2 = new JPanel();
temp2.add(new JLabel("Value for Beta:     "));
//temp2.add(betaCheckBox);
betaCheckBox.setSelected(true) ;
temp.add(temp2);
temp.add(createEastWestPanel(new JLabel(""), betalBox));
temp.add(new JLabel("")) ;
temp2 = new JPanel();
//temp2.add(new JLabel("Value for Alpha:     "));
temp2.add(new JLabel("     "));
temp.add(temp2);
//temp.add(createEastWestPanel(new JLabel(""), alphaBox));
this.add(temp);

//Create area for the user to input the number of iterations per
//variable
temp = new JPanel();
JLabel tempLabel = new JLabel("Iterations of F, f, cInit, Alpha-Beta, cProm");
temp.add(tempLabel);
temp.add(Frnum);
temp.add(fnum);
temp.add(cnum);
temp.add(alminbenum);
temp.add(rAnum);
this.add(temp);
```

```
    //Create area for user to decide wether or not to use RSS
    temp = new JPanel();
    JLabel rssLabel = new JLabel("Check here to use RSS") ;
    temp.add(rssLabel) ;
    temp.add(rssBox);
    if (windicator == 1)
      rssBox.setSelected(true) ;
    else
      rssBox.setSelected(false);
    this.add(temp);

    //Create area for user to input the range of ages to weight
    temp = new JPanel();
    temp2 = new JPanel();
    temp2.add(new JLabel("Age range on which to weight  "));
    temp.add(temp2);
    temp.add(createEastWestPanel(new JLabel("Lower: "), ageWeightlBox));
    temp.add(createEastWestPanel(new JLabel("Upper: "), ageWeightuBox));
    this.add(temp);

    temp = new JPanel();
    findFit.setActionCommand("BestFitWindow.FindFit");
    findFit.addActionListener(tabbedPaneListener);
    findBestFit.setActionCommand("BestFitWindow.FindBestFit");
    findBestFit.addActionListener(tabbedPaneListener);
    cancel.setActionCommand("BestFitWindow.Cancel");
    cancel.addActionListener(tabbedPaneListener);
    temp.add(findFit);
    temp.add(progressBar);
    temp.add(cancel);
    this.add(temp);
    cancel.setEnabled(false);
  }

// plot the OBS, OBS*, or OBS** when you enter the Find Fits,
// this way people know what they are fitting to- 4/21/04

public void plot() {
      plotted_OBS = true;
      graph.clearData();
      double normalizedIncidents ;
      /*double numlines;
      double thisAge = t[numlines-1] ;
      for (int j= numlines; j <50; j++){
        thisAge = thisAge + 5.0;
        t[j] = thisAge;
      */
        //Initialize and plot the graph

      double max_cases = 0;
      //double max_t = 0; - instead use set_maxt since we want the x-axis to be the same
      double temp_max = 0;
      for (int i =0; i<t.length; i++) {
        temp_max = cases_adj[i]*(100000.00/pop[i]) ;
        if(max_cases < temp_max) {
```

```
          max_cases = temp_max;
          //ageAtMax = t[i] ;
          //ageAtMaxInt = i ;
         }
         /*if(max_t < t[i]) {
          max_t = t[i];
          }*/
        }
        graph.setStart(0,0);
        graph.setYLabel("OBS**(h,t) per 100,000");
        int yInterval = Math.round(Math.round(max_cases/10));
        graph.setEnd(set_maxt + 30, max_cases + yInterval*3);
        BufferedImage img = new BufferedImage(5,5, BufferedImage.TYPE_INT_RGB);
        Graphics2D g2 = img.createGraphics();
        g2.setColor(Color.yellow);
        g2.fill(new java.awt.geom.Ellipse2D.Float(0,0, 5, 5));
        dataLineOBS = graph.addLine(img);  //Set the Global dataLine
        graph.setInterval(10,yInterval);
        graph.calibrate();
        graph.drawAxis();
        graph.removeLine(dataLine);
         //for(int i = 0; i < cases.length; i++){
         //  double temp = 1 - tot[i];
         //  System.out.println("Case: " + cases[i] + "TOT: " + temp);
         //  System.out.println("Cases/(1-tot):" + cases_adj[i]);
         //}
         for(int i =0; i<t.length; i++) {
          if(t[i] == 0)
            break;
          else {
            //normalize incidents per 100,000 - completed above
            normalizedIncidents = (cases_adj[i])*(100000.00/pop[i]) ;
            graph.addPoint(dataLineOBS, t[i], normalizedIncidents);
          }
         }
        /* }*/
}


/** This finds the single best fit line given the parameters. */
  void runBestFit() {
     resetSolutions();
     selectFitWindow.entryTable.updateTable(solutions);
     setBounds();
     selectFitWindow.clearAll();

     //Get all the parameters from the pervious window.
     slopemax = estimateWindow.maxSlope.getValue();
     detamax = estimateWindow.maxDeta.getValue();

     findFit.setEnabled(false);
     findBestFit.setEnabled(false);
     cancel.setEnabled(true);

     runFindFitWorker = new SwingWorker() {
        public Object construct() {
```

```
            runFindFits();
            return null;
        }

        //Runs on the event-dispatching thread.
        public void finished() {
            findFit.setEnabled(true);
            findBestFit.setEnabled(true);
            bestfits(new File(filename), slopemax, detamax);
            //There should be some further testing done here to zoom in
            //on the best values.  The code is here for the matrix stuff.


            //Now we graph it
            BufferedImage img = new BufferedImage(5,5, BufferedImage.TYPE_INT_RGB);
            Graphics2D g2 = img.createGraphics();
            g2.setColor(Color.green);
            g2.fill(new java.awt.geom.Ellipse2D.Float(0,0, 5, 5));
            int linenumber = graph.addLine(img);

            solutions[0][0] = Fraco;
            solutions[0][1] = fo;
            solutions[0][2] = co;
            solutions[0][3] = brtho;
            solutions[0][4] = detho;
            solutions[0][5] = rAo;
            solutions[0][6] = disto;
            solutions[0][7] = slpo * (double)100000;
            solutions[0][8] = detao;
            nfits = 1;

            selectFitWindow.entryTable.updateTable(solutions);

            graphLine(Fraco, fo, co, brtho, detho, rAo, slpo, detao, linenumber, 45, 120);

            cancel.setEnabled(false);
            breakOut = false;
        }
    };
    runFindFitWorker.start();  //required for SwingWorker 3
}

void runFindFitWorker() {
    //This code multithreads running best fits so we can still see the window.
    findFit.setEnabled(false);
    findBestFit.setEnabled(false);
    cancel.setEnabled(true);
    runFindFitWorker = new SwingWorker() {
        public Object construct() {
            runFindFits();
            return null;
        }

        //Runs on the event-dispatching thread.
        public void finished() {
            findFit.setEnabled(true);
```

```
            findBestFit.setEnabled(true);
            selectFitWindow.clearAll();
            selectFitWindow.entryTable.updateTable(solutions);
                    // CHANGED BELOW FROM 9 to 10 to add Equations Page - 5/5/04
            tabbedPane.setSelectedIndex(10);
            cancel.setEnabled(false);
            breakOut = false;
        }
    };
    runFindFitWorker.start();  //required for SwingWorker 3
}

void runFindFits() {
    resetSolutions();
    selectFitWindow.entryTable.updateTable(solutions);
    setBounds();
    selectFitWindow.clearAll();

    //Get all the parameters from the pervious window.
    slopemax = estimateWindow.maxSlope.getValue();
    detamax = estimateWindow.maxDeta.getValue();

    //Clear the file if it exists, I should check to see if
    //the user wants to overwrite the file, I will do this
    //later if it matters.
    try {
        File file = new File(filename);
        file.delete();
        FileWriter writer = new FileWriter(filename, true);
        BufferedWriter fileWriter = new BufferedWriter(writer);
        evalgrid(nobs, fileWriter);
        bestfits(file, slopemax, detamax);
        goodfits(file, slopemax, detamax);
        writer.close();
        selectFitWindow.entryTable.updateTable(solutions);
        }
    catch (Exception e) {
        System.out.println("File error in runFindFits.");
        e.printStackTrace();
    }
}
void setProgress(int val) {
    progressBar.setValue(val);
}

void setVals(double[] lower, double[] upper) {
    FrlBox.setText("" + lower[0]);
    FruBox.setText("" + upper[0]);
    flBox.setText("" + lower[1]);
    fuBox.setText("" + upper[1]);
    clBox.setText("" + lower[2]);
    cuBox.setText("" + upper[2]);
    alminbelBox.setText("" + lower[3]);
    alminbeuBox.setText("" + upper[3]);
    betalBox.setText("" + lower[4]);
    betauBox.setText("" + upper[4]);
```

```
    rAlBox.setText("" + lower[5]);
    rAuBox.setText("" + upper[5]);

    for(int i=0; i < boxList.size(); i++) {
      ((JTextField)boxList.get(i)).setCaretPosition(0);
    }
  }

  void setMU(double lower, double upper){
    alminbelBox.setText(truncString(String.valueOf(lower), 4));
    alminbeuBox.setText(truncString(String.valueOf(upper), 4));
  }

  void setC(double lower, double upper){
    clBox.setText(truncString(String.valueOf(lower), 4));
    cuBox.setText(truncString(String.valueOf(upper), 4));
  }

  void setrA(double lower, double upper){
    rAlBox.setText(String.valueOf(lower));
    rAuBox.setText(String.valueOf(upper));
  }

  void setFr(double lower, double upper){
    FrlBox.setText(String.valueOf(lower).substring(0,6));
    FruBox.setText(String.valueOf(upper).substring(0,6));
  }

  void setf(double lower, double upper){
    flBox.setText(String.valueOf(lower).substring(0,6));
    fuBox.setText(String.valueOf(upper).substring(0,6));
  }

  void resetCheckBoxes() {
    FrCheckBox.setSelected(false);
    fCheckBox.setSelected(false);
    cCheckBox.setSelected(false);
    alminbeCheckBox.setSelected(false);
    betaCheckBox.setSelected(false);
    rACheckBox.setSelected(false);
    ageBox.setSelected(false);
  }

  //Sets all of the bounds
  void setBounds() {
    Frl = FrlBox.getValue();
    Fru = FruBox.getValue();
    fl = flBox.getValue();
    fu = fuBox.getValue();
    cl = clBox.getValue();
    cu = cuBox.getValue();
    alminbehl = alminbel = alminbelBox.getValue();
    alminbehu = alminbeu = alminbeuBox.getValue();
    betal = betalBox.getValue();
    betau = betalBox.getValue();
    rAl = rAlBox.getValue();
```

```
rAu = rAuBox.getValue();
alpha = alphaBox.getValue();

    // May want to check this- fourth is being hardcoded to 1 ; - JK 11.18.03
nloop[0] = Frnum.getValue();
nloop[1] = fnum.getValue();
nloop[2] = cnum.getValue();
nloop[3] = alminbenum.getValue();
nloop[4] = 1 ;
nloop[5] = rAnum.getValue();

weightsAgel = ageWeightlBox.getValue();
weightsAgeu = ageWeightuBox.getValue();
ageIndexl = (new Long(Math.round(weightsAgel/5.0))).intValue() ;
ageIndexu = (new Long(Math.round(weightsAgeu/5.0))).intValue() ;

if(rssBox.isSelected()== true)
   windicator = 1 ;
else
   windicator = 0 ;

if(FrCheckBox.isSelected() == true) {
   pindicator[0] =1;
   nloop[0] = 1;
   Fru = Frl;
} else {
   nloop[0] = nloopOriginal[0];
}

if(fCheckBox.isSelected() == true) {
   pindicator[1] = 1;
   nloop[1] = 1;
   fu = fl;
} else {
   nloop[1] = nloopOriginal[1];
}

if(cCheckBox.isSelected() == true) {
   pindicator[2] = 1;
   nloop[2] = 1;
   cu = cl;
} else {
   nloop[2] = nloopOriginal[2];
}

if(alminbeCheckBox.isSelected() == true) {
   pindicator[3] = 1;
   nloop[3] = 1;
   alminbeu = alminbel;
} else {
   nloop[3] = nloopOriginal[3];
}

if(betaCheckBox.isSelected() == true) {
   pindicator[4] = 1;
   nloop[4] = 1;
```

```
            betau = betal;
        } else {
            nloop[4] = nloopOriginal[4];
        }

        if(rACheckBox.isSelected() == true) {
            pindicator[5] = 1;
            nloop[5] = 1;
            rAu = rAl;
        } else {
            nloop[5] = nloopOriginal[5];
        }

        if(ageBox.isSelected()){
            //This operation takes foreveer!
            runABMod = true;
        } else {
            runABMod = false;
        }

    }
}


//Window for directly inputting parameter values and seeing there plot
class GraphSingleFitWindow extends JPanel{
    int textBoxSize = 6;

    //This is just a ton of fields that we need references to to get the
    //values out of.
    DoubleNumberField FrBox = new DoubleNumberField(Frl, textBoxSize);
    DoubleNumberField fBox = new DoubleNumberField(fl, textBoxSize);
    DoubleNumberField cBox = new DoubleNumberField(cl, textBoxSize);
    DoubleNumberField alminbeBox = new DoubleNumberField(alminbel, textBoxSize);
    DoubleNumberField betaBox = new DoubleNumberField(betal, textBoxSize);
    DoubleNumberField rABox = new DoubleNumberField(rAl, textBoxSize);
    DoubleNumberField distBox = new DoubleNumberField(0.0, textBoxSize);

    //Allowing N and M to be varied
    DoubleNumberField nNum = new DoubleNumberField(ninit, 3);
    DoubleNumberField mNum = new DoubleNumberField(minit , 3);

    //Allow for user to input values for rB, rC (more if equation can generalize that far)
    DoubleNumberField rBNum = new DoubleNumberField(rB, textBoxSize);
    DoubleNumberField rCNum = new DoubleNumberField(rC, textBoxSize);

    JButton plotButton = new JButton("Plot") ;
    JButton unplotButton = new JButton("Unplot") ;
    JButton pobsButton = new JButton("Pobs") ;

    Vector boxList = new Vector();

    JPanel bottomDisplay = new JPanel();

    String filename = "TempData.txt";
```

```
GraphSingleFitWindow(){
   boxList.add(FrBox);
   boxList.add(fBox);
   boxList.add(cBox);
   boxList.add(alminbeBox);
   boxList.add(betaBox);
   boxList.add(rABox);
   boxList.add(distBox);

   //Create the layout to enter in all of the parameters.
   JPanel temp = new JPanel();
   JPanel temp2 = new JPanel();
   temp.setLayout(new GridLayout(8, 3));
   temp.add(createEastWestPanel(new JLabel("F Value: "), FrBox));
   temp.add(createEastWestPanel(new JLabel("f Value: "), fBox));
   temp.add(createEastWestPanel(new JLabel("cInit Value: "), cBox));
   temp.add(createEastWestPanel(new JLabel("Alpha - Beta Value: "), alminbeBox));
   temp.add(createEastWestPanel(new JLabel("cProm Value: "), rABox));
   temp.add(createEastWestPanel(new JLabel("Beta Value:"), betaBox));
   temp.add(createEastWestPanel(new JLabel("Distance:"), distBox));
   this.add(temp);

   temp = new JPanel();
   plotButton.setActionCommand("GraphSingleFitWindow.plot");
   plotButton.addActionListener(tabbedPaneListener);
   temp.add(plotButton) ;
   unplotButton.setActionCommand("GraphSingleFitWindow.unplot");
   unplotButton.addActionListener(tabbedPaneListener);
   temp.add(unplotButton) ;
   pobsButton.setActionCommand("GraphSingleFitWindow.plotPobs");
   pobsButton.addActionListener(tabbedPaneListener);
   temp.add(pobsButton) ;
   this.add(temp);
}

void plot()
{
   double Fr = FrBox.getValue();
   double f = fBox.getValue();
   double c = cBox.getValue();
   double rA = rABox.getValue();
   double beta = betaBox.getValue();
   double alminbe = alminbeBox.getValue();
   double brth = Math.log((double)2) * (beta + alminbe);
   double deth = Math.log((double)2) * beta;;
   double slp = Double.parseDouble(estimateWindow.maxSlope.getText());
   double deta = Double.parseDouble(estimateWindow.maxDeta.getText());

   BufferedImage img = new BufferedImage(5,5, BufferedImage.TYPE_INT_RGB);
   Graphics2D g2 = img.createGraphics();
   g2.setColor(Color.red);
   g2.fill(new java.awt.geom.Ellipse2D.Float(0,0, 5, 5));
   graphSingleFitLine = graph.addLine(img); //Set the Global dataLine
   double dist = graphLine(Fr, f, c, alminbe, beta, rA, slp, deta, population, graphSingleFitLine, true);
   distBox.setText(truncString(dist+"", 2)) ;
}
```

```
    void unplot()
    {
       graph.removeExcessLines(dataLine);
    }

    void plotPobs(){
       double Fr = 1.0;
       double f = 1.0;
       double c = cBox.getValue();
       double rA = rABox.getValue();
       double beta = betaBox.getValue();
       double alminbe = alminbeBox.getValue();
       double brth = Math.log((double)2) * (beta + alminbe);
       double deth = Math.log((double)2) * beta;
       double slp = Double.parseDouble(estimateWindow.maxSlope.getText());
       double deta = Double.parseDouble(estimateWindow.maxDeta.getText());

       graphPobs(Fr, f, c, alminbe, beta, rA, slp, deta, population);
    }

}
//End GraphSingleFitWindow


 class CompareRatioWindow extends JPanel{
    int textBoxSize = 6;

    //This is just a ton of fields that we need references to to get the
    //values out of.
    DoubleNumberField FrBox = new DoubleNumberField(Frl, textBoxSize);
    DoubleNumberField fBox = new DoubleNumberField(fl, textBoxSize);
    DoubleNumberField cBox = new DoubleNumberField(cl, textBoxSize);
    DoubleNumberField alminbeBox = new DoubleNumberField(alminbel, textBoxSize);
    DoubleNumberField betaBox = new DoubleNumberField(betal, textBoxSize);
    DoubleNumberField rABox = new DoubleNumberField(rAl, textBoxSize);
    DoubleNumberField distBox = new DoubleNumberField(0.0, textBoxSize);

    //Allowing N and M to be varied
    DoubleNumberField nNum = new DoubleNumberField(ninit, 3);
    DoubleNumberField mNum = new DoubleNumberField(minit , 3);

    //Allow for user to input values for rB, rC (more if equation can generalize that far)
    DoubleNumberField rBNum = new DoubleNumberField(rB, textBoxSize);
    DoubleNumberField rCNum = new DoubleNumberField(rC, textBoxSize);

   JButton plotRatioButton = new JButton("Plot Ratio") ;
   JButton unplotButton = new JButton("Unplot Fit Ratio") ;
   JButton plotFitRatioButton = new JButton("Plot Fit Ratio");
   //JButton pobsButton = new JButton("Pobs") ;

    Vector boxList = new Vector();

    JPanel bottomDisplay = new JPanel();

    String filename = "TempData.txt";
```

```
BufferedImage img;// = new BufferedImage(5,5, BufferedImage.TYPE_INT_RGB);
Graphics2D g2;// = img.createGraphics();
GraphWindow ratioGraph; // = new GraphWindow(400, 400) ;
//boolean first_time = true;
ArrayList originalfit = new ArrayList();
int ratioFitlinenumber;

CompareRatioWindow(){
    boxList.add(FrBox);
    boxList.add(fBox);
    boxList.add(cBox);
    boxList.add(alminbeBox);
    boxList.add(betaBox);
    boxList.add(rABox);
    boxList.add(distBox);

    //Create the layout to enter in all of the parameters.
    JPanel temp = new JPanel();
    JPanel temp2 = new JPanel();
    temp.setLayout(new GridLayout(8, 3));
    temp.add(createEastWestPanel(new JLabel("F Value: "), FrBox));
    temp.add(createEastWestPanel(new JLabel("f Value: "), fBox));
    temp.add(createEastWestPanel(new JLabel("cInit Value: "), cBox));
    temp.add(createEastWestPanel(new JLabel("Alpha - Beta Value: "), alminbeBox));
    temp.add(createEastWestPanel(new JLabel("cProm Value: "), rABox));
    temp.add(createEastWestPanel(new JLabel("Beta Value:"), betaBox));
    temp.add(createEastWestPanel(new JLabel("Distance:"), distBox));
    this.add(temp);

    temp = new JPanel();
    plotRatioButton.setActionCommand("CompareRatioWindow.setScale");
    plotRatioButton.addActionListener(tabbedPaneListener);
    temp.add(plotRatioButton) ;
        plotFitRatioButton.setActionCommand("CompareRatioWindow.plotFit");
        plotFitRatioButton.addActionListener(tabbedPaneListener);
        temp.add(plotFitRatioButton);
        /*      unplotButton.setActionCommand("CompareRatioWindow.unplot");
    unplotButton.addActionListener(tabbedPaneListener);
    temp.add(unplotButton) ; */
    this.add(temp);

        //BufferedImage img = new BufferedImage(5,5, BufferedImage.TYPE_INT_RGB);
        //Graphics2D g2 = img.createGraphics();
        //          GraphWindow ratioGraph = new GraphWindow(400, 400) ;
}

public void setScale(){
    xstartBox = new WholeNumberField(0, 5) ;
    xendBox = new WholeNumberField(100, 5) ;
    ystartBox = new WholeNumberField(0, 5) ;
    yendBox = new WholeNumberField(2, 5) ;
    JButton graph = new JButton("Create Graph") ;

    JFrame setScaleFrame = new JFrame("Set Scale") ;
    JPanel temp = new JPanel() ;
```

```
    temp.setLayout(new GridLayout(8, 3));
    temp.add(new JLabel("X    ")) ;
    temp.add(createEastWestPanel(new JLabel("Start:  "), xstartBox));
    temp.add(createEastWestPanel(new JLabel("End:  "), xendBox));
    temp.add(new JLabel("Y    ")) ;
    temp.add(createEastWestPanel(new JLabel("Start:  "), ystartBox));
    temp.add(createEastWestPanel(new JLabel("End:  "), yendBox));
    //setScaleFrame.getContentPane().add(temp) ;

    //temp = new JPanel() ;
    graph.setActionCommand("CompareRatioWindow.plotRatio");
    graph.addActionListener(tabbedPaneListener);
    temp.add(graph) ;

    setScaleFrame.getContentPane().add(temp) ;
    setScaleFrame.setSize(200, 200) ;
    setScaleFrame.setVisible(true) ;
  }


//Creates a new window that displays a graph of Pobs
  public void plotRatio(){
  //create a popup frame
  JFrame ratioFrame = new JFrame("Ratio") ;
  ratioGraph = new GraphWindow(400, 400) ;

  //Setup the GraphWindow
  ratioGraph.setStart(xstartBox.getValue(), ystartBox.getValue());
  ratioGraph.setXLabel("Age (years)");
  ratioGraph.setYLabel("Ratio OBS1 / OBS2");
  ratioGraph.setEnd(xendBox.getValue(), yendBox.getValue());
  //  BufferedImage img = new BufferedImage(5,5, BufferedImage.TYPE_INT_RGB);
  // Graphics2D g2 = img.createGraphics();
      img = new BufferedImage(5,5, BufferedImage.TYPE_INT_RGB);
      g2 = img.createGraphics();
  g2.setColor(Color.green);
  g2.fill(new java.awt.geom.Ellipse2D.Float(0,0, 5, 5));
  int lineNum = ratioGraph.addLine(img);
  ratioGraph.setOffset(45,45);
  ratioGraph.setInterval(Math.round(Math.round(xendBox.getValue()/10)), 0.1);
  ratioGraph.calibrate();
  ratioGraph.drawAxis();


      // ADDED THESE TWO LINENUMS
  img = new BufferedImage(5,5, BufferedImage.TYPE_INT_RGB);
  g2 = img.createGraphics();
  g2.setColor(Color.blue);
  g2.fill(new java.awt.geom.Ellipse2D.Float(0,0, 5, 5));
  int lineNum2 = ratioGraph.addLine(img);

      img = new BufferedImage(5,5, BufferedImage.TYPE_INT_RGB);
  g2 = img.createGraphics();
  g2.setColor(Color.blue);
  g2.fill(new java.awt.geom.Ellipse2D.Float(0,0, 5, 5));
  int lineNum3 = ratioGraph.addLine(img);
```

```
      boolean t_greater = false;
      if (t[0] >= t2[0]){
       t_greater = true;
      }

      int offset;
       if(t_greater){
         for(offset = 0; offset < t2.length; offset++){
          if(t[0] == t2[offset]){
                break;
          }
         }
       }
       else{
         for(offset = 0; offset < t.length; offset++){
          if(t[offset] == t2[0]){
                break;
          }
         }
       }

       int temp_len = Math.min(t.length, t2.length);

       for(int i =0; i<temp_len-offset-1; i++) { //t.length
      if(t[i] == 0)
       break;
      else {

       ratioGraph.addPoint(lineNum, t[i], ((Double) Ratio_global.get(i)).doubleValue());
       ratioGraph.addPoint(lineNum2, t[i], ((Double) Ratio_plus_global.get(i)).doubleValue());
       ratioGraph.addPoint(lineNum3, t[i], ((Double) Ratio_minus_global.get(i)).doubleValue());


      }
       }

 ratioFrame.getContentPane().add(ratioGraph) ;
 ratioFrame.setSize(450, 450) ;
 ratioFrame.setVisible(true) ;

 }

 void plotFit()
 {

  double Fr = FrBox.getValue();
       double f = fBox.getValue();
       c = cBox.getValue();
       rA = rABox.getValue();
       double beta = betaBox.getValue();
       double alminbe = alminbeBox.getValue();
       brth = Math.log((double)2) * (beta + alminbe);
       deth = Math.log((double)2) * beta;;
       //slp = Double.parseDouble(estimateWindow.maxSlope.getText());
       //deta = Double.parseDouble(estimateWindow.maxDeta.getText());
```

```
            //slopemax = slp * Math.pow(10,5);
            //detamax = deta;
            // ArrayList originalfit = new ArrayList();
            //          boolean first_time = true;
            ArrayList otherfit = new ArrayList();

            img = new BufferedImage(5,5, BufferedImage.TYPE_INT_RGB);
            g2 = img.createGraphics();
            g2.setColor(Color.red);
            g2.fill(new java.awt.geom.Ellipse2D.Float(0,0, 5, 5));
            if (ratioFitlinenumber > 2){
             ratioGraph.removeLine(ratioFitlinenumber);
            }
            ratioFitlinenumber = ratioGraph.addLine(img);

            twostage(nobs);


            for(int ii=0; ii<nobs+extension; ii++) {
             double fit = Pobs[ii] * Fr/ (Fr + ((double)1 - Fr) * Math.exp((1-survivalFraction[ii])*intPobs[ii] /
f)) * population;
             otherfit.add(new Double(fit));
            }
            // set the first calculation to the original value
            if(first_time == true){
             //          System.out.println("Here");
             first_time = false;
             originalfit = new ArrayList();
             for(int loop = 0; loop < otherfit.size(); loop++){
              originalfit.add((Double) otherfit.get(loop));
             }
            }
             // get the ratio at each point and graph
             for(int loop2 = 0; loop2 < otherfit.size(); loop2++){
              double pt = (((Double) otherfit.get(loop2)).doubleValue()) / (((Double)
originalfit.get(loop2)).doubleValue());
              //if (loop2 > 10 && loop2 < 21){
               //System.out.println("Original:");
               //System.out.println(((Double)originalfit.get(loop2)).doubleValue());
               //System.out.println("Other:");
               //System.out.println(((Double)otherfit.get(loop2)).doubleValue());
              //}
               ratioGraph.addPoint(ratioFitlinenumber, t[loop2], pt);
             }

             ratioGraph.repaint();

    }


    /*  public void valueChanged(ListDataEvent e) {
         if (e.getValueIsAdjusting()) {
          return;
         }
         //Plot the data
         ListSelectionModel lsm = (ListSelectionModel)e.getSource();
```

```
        if (lsm.isSelectionEmpty()) {
         return;
        } else {
         ratioGraph.removeLine(ratioFitlinenumber);
         plotFit();
        }
        }*/

    void unplot()
    {
        ratioGraph.removeExcessLines(dataLine);
    }



  }
 //End CompareRatioWindow

class SelectFitWindow extends JPanel implements ActionListener{
    JLabel directions = new JLabel();
    EntryTable entryTable = new EntryTable();
    JTable entryView = null;
    int checkcount = 0;
    JButton reRun = new JButton("<< Re-Run Find Fits");
    JButton plot = new JButton("Plot");
    JButton unPlot = new JButton("UnPlot");
    JButton plotPobs = new JButton("Plot Pobs");
    JButton save = new JButton("Save Results") ;
    JButton data =  new JButton("Show Data") ;
    int lineNumber; //the line number we are using to graph with.

    SelectFitWindow() {
      //Create the Layout
      this.setLayout(new BoxLayout(this, BoxLayout.Y_AXIS));

      //have to use the tmp jpanel because I could not get the layout to
      //properly display the directions otherwise.
      JLabel directions = new JLabel();
      JPanel tmp = new JPanel();
      directions.setText("Please select the parameter list you want to fit better.");
      tmp.add(directions);
      directions = new JLabel();
      directions.setText("Click Plot to view the line, UnPlot to remove the last line you added.");
      tmp.add(directions);
      this.add(tmp);

      //Create and add the table and its listener for row selection
      //and popup menu
      entryView = new JTable(entryTable);
      entryView.setPreferredScrollableViewportSize(new Dimension(300, 290));
      entryView.addMouseListener(new PopupListener(initPopupMenu()));
      ListSelectionModel rowSM = entryView.getSelectionModel();
      rowSM.addListSelectionListener(entryTable);

      JScrollPane scrollPne = new JScrollPane(entryView);
      this.add(scrollPne);
```

```
    //Setup the Bottom Buttons
    reRun.setActionCommand("SelectFitWindow.reRun");
    reRun.addActionListener(tabbedPaneListener);
    plot.setActionCommand("SelectFitWindow.Plot");
    plot.addActionListener(this);
    unPlot.setActionCommand("SelectFitWindow.unPlot");
    unPlot.addActionListener(this);
    plotPobs.setActionCommand("SelectFitWindow.plotPobs");
    plotPobs.addActionListener(tabbedPaneListener);
    save.setActionCommand("SelectFitWindow.save");
    save.addActionListener(tabbedPaneListener);
    data.setActionCommand("SelectFitWindow.data");
    data.addActionListener(tabbedPaneListener);

    //Add in the Bottom Buttons.
    tmp = new JPanel();
    //tmp.add(reRun); // take out the Re-Run Find Fits button
    tmp.add(plot);
    tmp.add(unPlot);
    tmp.add(plotPobs);
    tmp.add(save) ;
    tmp.add(data) ;
    this.add(tmp);

    //Create a line to use to graph whatever the currently selected
    //row is.
    BufferedImage img = new BufferedImage(5,5, BufferedImage.TYPE_INT_RGB);
    Graphics2D g2 = img.createGraphics();
    g2.setColor(Color.blue);
    g2.fill(new java.awt.geom.Ellipse2D.Float(0,0, 5, 5));
    lineNumber = graph.addLine(img);
}

public void clearAll() {
    entryTable.rowCount = 0;
    entryTable.data = null;
    entryTable.selectedIndex = 0;
}

//Goes back to the previous window using the selected line's parameters
//to redo the bounds.
public void reRun() {
    //make a lower and upper array for bestFitWindow
    double lower[] = new double[6];
    double upper[] = new double[6];
    for(int i = 0; i < 6; i ++) {
        lower[i] = solutions[entryTable.selectedIndex][i] * .95;
        upper[i] = solutions[entryTable.selectedIndex][i] * 1.05;
    }
        lower[4] = solutions[entryTable.selectedIndex][4] ;
        upper[4] = solutions[entryTable.selectedIndex][4] ;
    bestFitWindow.resetCheckBoxes();
    bestFitWindow.setVals(lower, upper);
}
```

```
//Plots the selected entry using the temporary line created by this
//Panel
void tempPlot(int row) {
   int index = row;
   graphLine(solutions[index][0],
   solutions[index][1],
   solutions[index][2],
   solutions[index][3],
   solutions[index][4],
   solutions[index][5],
   solutions[index][6],
   solutions[index][7],
   population,
   lineNumber);

   first_time = true;
       compareRatioWindow.FrBox.setValue(solutions[index][0]);
       compareRatioWindow.fBox.setValue(solutions[index][1]);
       compareRatioWindow.cBox.setValue(solutions[index][2]);
       compareRatioWindow.alminbeBox.setValue(solutions[index][3]);
       compareRatioWindow.betaBox.setValue(solutions[index][4]);
       compareRatioWindow.rABox.setValue(solutions[index][5]);
       compareRatioWindow.distBox.setValue(solutions[index][6]);
       //entryTable.setValueAt(tempArea+"", index, 8) ;
}

public void setScale(){
   xstartBox = new WholeNumberField(0, 5) ;
   xendBox = new WholeNumberField(200, 5) ;
   ystartBox = new WholeNumberField(0, 5) ;
   yendBox = new WholeNumberField(Math.round(Math.round(maxcases*10)), 5) ;
   JButton graph = new JButton("Create Graph") ;

   JFrame setScaleFrame = new JFrame("Set Scale") ;
   JPanel temp = new JPanel() ;
   temp.setLayout(new GridLayout(8, 3));
   temp.add(new JLabel("X    ")) ;
   temp.add(createEastWestPanel(new JLabel("Start:  "), xstartBox));
   temp.add(createEastWestPanel(new JLabel("End:  "), xendBox));
   temp.add(new JLabel("Y    ")) ;
   temp.add(createEastWestPanel(new JLabel("Start:  "), ystartBox));
   temp.add(createEastWestPanel(new JLabel("End:  "), yendBox));
   //setScaleFrame.getContentPane().add(temp) ;

   //temp = new JPanel() ;
   graph.setActionCommand("SetScale.graph");
   graph.addActionListener(tabbedPaneListener);
   temp.add(graph) ;

   setScaleFrame.getContentPane().add(temp) ;
   setScaleFrame.setSize(200, 200) ;
   setScaleFrame.setVisible(true) ;
}

//Creates a new window that displays a graph of Pobs
public void plotPobs(){
```

```
        //create a popup frame
        JFrame pobsFrame = new JFrame("Pobs") ;
        GraphWindow pobsGraph = new GraphWindow(400, 400) ;

        //Setup the GraphWindow
        pobsGraph.setStart(xstartBox.getValue(), ystartBox.getValue());
        pobsGraph.setXLabel("Age (years)");
        pobsGraph.setYLabel("POBS*(t) per 100,000");
        pobsGraph.setEnd(xendBox.getValue(), yendBox.getValue());
        BufferedImage img = new BufferedImage(5,5, BufferedImage.TYPE_INT_RGB);
        Graphics2D g2 = img.createGraphics();
        g2.setColor(Color.green);
        g2.fill(new java.awt.geom.Ellipse2D.Float(0,0, 5, 5));
        int linenumber = pobsGraph.addLine(img);
        pobsGraph.setOffset(45,45);
        pobsGraph.setInterval(Math.round(Math.round(xendBox.getValue()/10)),
Math.round(Math.round(yendBox.getValue()/10)));
        pobsGraph.calibrate();
        pobsGraph.drawAxis();


        for(int ii=0; ii<nobs+extension; ii++) {
          pobsGraph.addPoint(linenumber, t[ii], Frac*Pobs[ii]*population) ;
        }

        pobsFrame.getContentPane().add(pobsGraph) ;
        pobsFrame.setSize(450, 450) ;
        pobsFrame.setVisible(true) ;

        }

        //*************Listner Functions*****************//
        boolean make_plot = false;
        //Allows this to be an action listener
        public void actionPerformed(ActionEvent e) {
          String action = e.getActionCommand();
          if(action.compareTo("SelectFitWindow.Plot") == 0) {
            int index = entryTable.selectedIndex;
            //graph the line, assuming we want the full population.
            graphLine(solutions[index][0],
            solutions[index][1],
            solutions[index][2],
            solutions[index][3],
            solutions[index][4],
            solutions[index][5],
            solutions[index][6],
            solutions[index][7],
            population);
                    make_plot = true;
          } else if(action.compareTo("SelectFitWindow.unPlot") == 0) {
              if (make_plot == true){
                    graph.removeLastLine();
                    make_plot = false;
              }
```

```
    } else if(action.compareTo("SelectFitWindow.plotPobs") == 0) {

    }
  }

  //****************Inner Classes*********************//

  //A to show the various values and allow the user to pick a parameter
  //list to user
  public class EntryTable extends AbstractTableModel implements ListSelectionListener {
    Vector columnNames = new Vector();
    int rowCount = 0;
    Object[][] data;
    int selectedIndex = 0;

    public EntryTable() {
      columnNames.add("F");
      columnNames.add("f");
      columnNames.add("cInit");
      columnNames.add("Alpha-Beta");
      columnNames.add("Beta");
      columnNames.add("cProm");
      columnNames.add("Distance");
      columnNames.add("X");
      columnNames.add("Area");
      columnNames.add("Sel");
    }

    //Initializes the table
    public void updateTable(double[][] sols) {

      data = new Object[nfits][10];

      sortedsolutions = solutions ;

      //changed here
      for(int i =0; i < Math.min(nfits, 100); i++) {
        for(int ii = 0; ii < 7; ii++) {
          if ((ii == 0) | (ii == 1)){
            if (Double.toString(sortedsolutions[i][ii]).length() > 6 )
              data[i][ii] = Double.toString(sortedsolutions[i][ii]).substring(0, 6) ;
            else
              data[i][ii] = Double.toString(sortedsolutions[i][ii]) ;
          }
          else {
            int tempIndex = Double.toString(sortedsolutions[i][ii]).lastIndexOf("E") ;
            int tempIndex2 = Double.toString(sortedsolutions[i][ii]).lastIndexOf(".") ;
            int length = Double.toString(sortedsolutions[i][ii]).length() ;
            if (tempIndex > 0)
              if (tempIndex > 4)
                data[i][ii] = Double.toString(sortedsolutions[i][ii]).substring(0, 4) +
Double.toString(sortedsolutions[i][ii]).substring(tempIndex);
              else
                data[i][ii] = Double.toString(sortedsolutions[i][ii]) ;
            else if (tempIndex2 > 0)
              if (length > tempIndex2+4)
```

```
                    data[i][ii] = Double.toString(sortedsolutions[i][ii]).substring(0, tempIndex2+4);
                  else
                    data[i][ii] = Double.toString(sortedsolutions[i][ii]);
                else
                  data[i][ii] = Double.toString(sortedsolutions[i][ii]) ;
              }
            }
            if (minit == 0.0)
              data[i][5] = Double.parseDouble((String)data[i][5]) /
((Double.parseDouble((String)data[i][3]) +Double.parseDouble((String)data[i][4])) /
Double.parseDouble((String)data[i][3])) + "";
            data[i][7] = truncString(Double.toString(calculateX(initiateableAge,
sortedsolutions[i][3]/(sortedsolutions[i][3]+sortedsolutions[i][4]), sortedsolutions[i][2])), 4) ;

            //calculate the area under the curve and display it
            //double area = 0.0 ;
            double area =getArea(Double.parseDouble((String)data[i][0]),
Double.parseDouble((String)data[i][1]), Double.parseDouble((String)data[i][2]),
Double.parseDouble((String)data[i][3]), Double.parseDouble((String)data[i][4]),
Double.parseDouble((String)data[i][5]), sortedsolutions[i][6], sortedsolutions[i][7]) * 5.0;
            data[i][8] = truncString(Double.toString(area), 4) ;

            data[i][9] = new Boolean(false);
            rowCount++;
          }
          this.fireTableRowsInserted(0, nfits-1);
        }

        public int getColumnCount() {
          return columnNames.size();
        }

        public int getRowCount() {
          return rowCount;
        }

        public String getColumnName(int col) {
          return (String)columnNames.get(col);
        }

        public Object getValueAt(int row, int col) {
          return data[row][col];
        }

        public Class getColumnClass(int c) {
          return getValueAt(0, c).getClass();
        }



        public void setValueAt(Object value, int row, int col) {
          data[row][col] = value;
          selectedIndex = row;
          if(value.equals(new Boolean(true))) {
            //turn all the other ones off
```

```
            for(int i=0; i < rowCount; i++) {
               if(i == row) {
                  continue;
               } else
                  data[i][9] = new Boolean(false);
               }
            }
            fireTableDataChanged();
         }

         public boolean isCellEditable(int row, int col) {
            if(col == 9) {
               return true;
            } else
               return false;
         }

         public void valueChanged(ListSelectionEvent e) {
            if (e.getValueIsAdjusting()) {
               return;
            }
            //Plot the data
            ListSelectionModel lsm = (ListSelectionModel)e.getSource();
            if (lsm.isSelectionEmpty()) {
               return;
            } else {
                        graph.removeLine(lineNumber);
               int row = entryView.getSelectedRow();
               tempPlot(row);
            }
         }

      }

}


 /** Settings is a JFrame window that pops up that allows the user to change
  * the settings.*/
 public class SettingsWindow extends JFrame implements ActionListener{

    boolean fullSizeGraph = true;
    JCheckBox checkBox = new JCheckBox();

    SettingsWindow(){
       this.setSize(200,100);
       this.setTitle("Settings");
       JPanel temp = new JPanel();
       temp.add(new JLabel("Full Graph Size"));
       checkBox.addActionListener(this);
       temp.add(checkBox);
       this.getContentPane().add(temp);
       JButton tempB = new JButton("OK");
       tempB.setActionCommand("OK");
       tempB.addActionListener(this);
       temp.add(tempB);
```

```
    }

    public void actionPerformed(ActionEvent e){
        String action = e.getActionCommand();
        if(action.compareTo("OK") == 0) {
            fullSizeGraph = checkBox.isSelected();
            this.show(false);
        }
    }

}


/*************************** Math ****************************/

//This resets all the variables so that a new file and testing can be run.
//not all the globals need to be reset, only those like nobs, pop, and
//others.
public void resetVars() {
    for(int i = 0; i < t.length; i++){
        obs[i] = 0;
        age[i] = 0;
        t[i] = 0;
        cases[i] = 0;
            // added 4/21/04
            cases_adj[i] = 0;
        pop[i] = 0;
        w[i] = 0;
        fit[i] = 0;
    }
    nobs = 0;
    reRan = false;
    nfits = 0;
    resetSolutions();
}

//Clears the solutions, this function exists in case we want to change it
//to a vector so that we don't have to change it everywhere in the code.
public void resetSolutions() {
    solutions = new double[1000][9];
}

// Vector is slp, slope, deta (all arrays) and then slopemax, detamax (in that order)
public Vector estimate(double[] t, double[] cases, double[] pop){

    windicator = 0;
    basedist = 0.0;

    for (int i = 1; i <= nobs; i++){
        obs[i] = cases[i] / pop[i] ;
        w[i] = 0.0 ;
        basedist = Math.max(basedist, Math.abs(obs[i]));
    }

    slopemax = 0.0 ;
```

```
    detamax = 0.0 ;
    double slope[] = new double[nobs];
    double deta[] = new double[nobs];

    for (int i = 1; i <= nobs; i++){
      // some of this conditional seems redundant
      if ( (i < nobs) && (t[i] > 20.0) ){
        // this uses log to base e
        slp[i] = ( (Math.log(obs[i+1]) - Math.log(obs[i])) / (t[i+1] - t[i]) ) ;
        slope[i] = (obs[i+1] - obs[i]) / (t[i+1] - t[i]) ;
        deta[i] = t[i] - (obs[i] / slope[i]) ;
        if (slope[i] > slopemax){
          slopemax = slope[i] ;
          detamax = deta [i] ;
        }
      }
    }

    Vector returnVec = new Vector();
    returnVec.add(slp);
    returnVec.add(slope);
    returnVec.add(deta);
    returnVec.add(Double.toString(slopemax));
    returnVec.add(Double.toString(detamax));
    return returnVec;
}

public double getAgeIntervals(int low, int high, double[] t, double[] obs){
    double sxx = 0.0 ;
    double ssxx = 0.0 ;
    double sxy = 0.0 ;
    double syy = 0.0 ;

    for (int i = low; i <= high+1; i++){
      sxx = sxx + t[i] ;
      syy = syy + ( Math.log(obs[i]) / Math.log(2.0) ) ;
      ssxx = ssxx + Math.pow(t[i], 2.0 );
      sxy = sxy + ( t[i] * Math.log(obs[i]) / Math.log(2.0) ) ;
    }

    double factor = ( (double) (2 + high -low) ) ;
    double almbet = (sxy - syy * sxx / factor ) / (ssxx - sxx * sxx / factor );
    return almbet;
}


public void twostage(int nobs) {
  double agemax = t[nobs - 1] + (double)1 + 50.0;
  double dt = agemax / (double)ngrid;
  double t, u, x1, x2, ff, dl, sprime;
    for (int i = 0; i <= ngrid; i++) {
      t = (double)i * dt;
      x1 = (double)0;
      x2 = (double)t;
      ff = -1.0 * DL(x1) * Sprime(x2) / (double)2;
      for (int j = 1; j < igrid; j++) {
```

```
            u = (double)j / (double)igrid;
            x1 = t * u;
            x2 = t * ((double)1 - u);
            ff = ff - DL(x1) * Sprime(x2);
         }
         x1 = t;
         x2 = (double)0;
         ff = ff - DL(x1) * Sprime(x2) / (double)2;
         ff = t * ff / (double)igrid;
         hfineA[i] = ff;
         age[i] = t;
      }
      inthfineA[0] = hfineA[0];
      for (int i = 1; i <= ngrid; i++)
         inthfineA[i] = inthfineA[i - 1] + hfineA[i];
      for (int i = 0; i <= ngrid; i++)
         inthfineA[i] = dt * (inthfineA[i] - (hfineA[i] + hfineA[1]) / (double)2);
      //Select those values that are really needed
      select(nobs);
      return;
   }

   public double DL(double x) {
      if (x < initiateableAge+1){
         double integralN = 0.0 ;
         int intX = Math.round(Math.round(Math.floor(x)));
         double remainderX = x - Math.floor(x) ;
         for (int i = 0; i <= intX; i++)
            integralN = integralN + N[i] ;
         integralN = integralN + (N[intX+1] * remainderX) ;
         double result = c * (integralN / NMax) * Math.pow(x, (double)(ninit - 1));
         //make change when beta = 0?  Prof. Morgenthaler says no
         //if (betal == 0.0)
            //result = result* (alpha/alpha-betal) ;
         //double result = c * Math.pow(x, (double)(ninit - 1));
         return result;
      }
      else
         return 0.0 ;
   }


   public double Sprime(double x) {
      double disc = Math.sqrt(Math.pow((brth - deth), 2.0) + (double)4 * rA * deth * brth);
      double rho1 = (brth + deth + disc) / (double)2;
      double rho2 = (brth + deth - disc) / (double)2;
      double B1 = (double)1 - rho2 / (((double)1 - rA) * brth);
      double B2 = rho1 / (((double)1 - rA) * brth) - (double)1;
      double r = Math.exp(disc * -1.0 * x);
      double S = (rho1 * B1 * r + rho2*B2) / ((1 - rA) * brth * (B1 * r + B2));
      double result = (1 - rA) * brth * Math.pow(S, (double)2) - (brth + deth) * S + deth;
      if ((minit == 1.0) || (minit == 0.0))
         return result;
      else
         return minit * result * Math.pow((1-S), (minit - 1)) ;
      }
```

```
/* This method finds the fitted age closest to the observed age
  This is where the algorithm is implemented. - 9/19/03 JK
*/
public void select(int nobs) {
   int iinf, isup;

   for (int i = 0; i < nobs+extension; i++) {
   //for (int i = 0; i < 30; i++) {
      // Find the bounding age values using the fact that age is sorted
      //boolean zeroIndicator = false;
      iinf = -1;
      do {
         iinf = iinf + 1;
      } while ((age[iinf] <= t[i]) && (iinf < 500));
      //if(zeroIndicator == false && age[iinf] == 0) {
      //  zeroIndicator = true;
      //} else if(zeroIndicator == true && age[iinf] ==0) {
      //  return;
      //}
      //iinf = iinf + 1;
      //}
      isup = iinf;
      iinf = isup - 1;
      // Interpolate
      // I see this as a potential problem (array out of bounds)
            hA[i] = hfineA[iinf] * (age[isup] - t[i]) + hfineA[isup]*(t[i] - age[iinf]);
            hA[i] = hA[i] / (age[isup] - age[iinf]);
            inthA[i] = inthfineA[iinf] * (age[isup] - t[i]) + inthfineA[isup] * (t[i] - age[iinf]);
            inthA[i] = inthA[i] / (age[isup] - age[iinf]);
   }

   double oneoverx = (1.0/calculateX(initiateableAge, (brth*Math.log(2)-
deth*Math.log(2))/(brth*Math.log(2)),c)) ;
   if (newEquation){
   for (int j = 0; j <nobs+extension; j++){
         // NEED TO CHANGE THIS TO Pobs[j] = hA[j]) - to test differences between Pobs = Vobs
         // AND unsimplified term
         // hA[j] = Vobs[j] - 4/26/04 JK
      Pobs[j] = 1 - Math.pow(Math.E, -1.0*hA[j]) ;
      intPobs[j] = inthA[j] ;
      }
      //Pobs[j] = oneoverx*hA[j] ;
   /*double sum = 0.0 ;
   for (int k = 0; k <nobs+extension; k++){
      sum = Pobs[k] + sum ;
      intPobs[k] = sum ;
      }*/
   }
   else {
   for (int j = 0; j <nobs+extension; j++){
      Pobs[j] = hA[j] ;
      intPobs[j] = inthA[j] ;
   }
   }
```

```
    return;
  }


public void evalgrid(int nobs, BufferedWriter fileWriter) {
    double dFr = (Fru-Frl) / (double)((Math.max(nloop[0], 2)) - (double)1);
    double df = (fu - fl) / (double)((Math.max(nloop[1], 2)) - (double)1);
    double dc = (cu - cl) / (double)((Math.max(nloop[2], 2)) - (double)1);
    double dalmbe = (alminbeu-alminbel) / (double)((Math.max(nloop[3], 2)) - (double)1);
    double dbeta = (betau-betal) / (double)((Math.max(nloop[4], 2)) - (double)1);
    double drA = (rAu - rAl) / (double)((Math.max(nloop[5], 2)) - (double)1);
    double slp, fit, prevfit, dist, deta, slpmax, alminbe, beta;

        // fix obs so that we fit to the
        // correct obs- either obs, obs*,
        // or obs** depending on what
        // information the user chooses to
        // enter - 4/21/04
        //double[] temp_obs = new double[50];
        for(int loop = 0; loop < obs.length; loop++){
         //obs[loop] = 0; // reset to 0
         //temp_obs[loop] = obs[loop];
         obs[loop] = cases_adj[loop] / pop[loop];
         //System.out.println("OBS: " + temp_obs[loop] + "OBS**: " + obs[loop]);
        }


    try {
       for (int i1 = 0; i1 < nloop[2]; i1++) {
          int temp = (int)((double)i1/(double)nloop[2] * 100.0);
          bestFitWindow.setProgress(temp);  //tells the GUI how far we are
          for (int i2 = 0; i2 < nloop[3]; i2++) {
             for (int i3 = 0; i3 < nloop[4]; i3++) {
                for (int i4 = 0; i4 < nloop[5]; i4++) {
                   c = cl + ((double)i1 * dc);
                   alminbe = alminbel + ((double)i2 * dalmbe);
                   beta = betal + ((double)i3 * dbeta);
                   double X = calculateX(initiateableAge, alminbe/(alminbe-beta), c) ;
                   rA = rAl + ((double)i4 * drA);

                   disto = Math.pow(10,10);
                   // Math.log = ln.  Is this what we want?
                   brth = Math.log((double)2) * (beta + alminbe);
                   deth = Math.log((double)2) * beta;
                   twostage(nobs);
                   // Now, loop over the values of F and f
                   for (int iii = 0; iii < nloop[0]; iii++) {
                      Frac = Frl + (double)iii * dFr;
                      for (int jjj = 0; jjj < nloop[1]; jjj++) {
                         f = fl + (double)jjj * df;

                         slp = 0.0;
                         fit = 0.0;
                         prevfit = 0.0;
                         dist = 0.0;
                         slpmax = 0.0;
```

```
                    deta = 0.0;


                    // want to separate OBS and OBS* calculations
                                    // we will calculate OBS and then calculate OBS* in separate steps
                                    // this will allow more flexible later (and clarity in code)
                    for (int ii = nobs - 1; ii >= 0; ii--) {
                      if(breakOut == true) {
                        bestFitWindow.setProgress(100);
                        return;
                      }
                      // what is this if newEquation
                      // statement doing? doesn't
                      // seem to have affect
                                    // first calculate OBS
                      if (newEquation){
                        fit = Pobs[ii] * Frac/ (Frac+ ((double)1 - Frac) * Math.exp((1-
survivalFraction[ii])*intPobs[ii] / f));
                      }
                      fit = Pobs[ii] * Frac / (Frac + ((double)1 - Frac) * Math.exp((1-
survivalFraction[ii])*intPobs[ii] / f));
                                    // now calculate OBS* --> NOT HERE 4/20/04 JK
                                    //        fit = fit / ((1-
survivalFraction[ii])*(reportError[ii])*(1-tot[ii]));
                      if (ii < nobs - 1)
                        slp = (prevfit - fit) / (t[ii+1] - t[ii]);
                      prevfit = fit;
                      if (slp > slpmax) {
                        slpmax = slp;
                        deta = t[ii] - obs[ii] / slp;
                      }
                      if (windicator == 1)
                        //weight distance metric to take into account Variance (case/pop^2)
                        if (weightsAgel > 100.0)
                          //w[ii] *
                          dist = dist + (Math.pow(fit-obs[ii], (double)2)) ;
                        else if ((ii > ageIndexl) && (ii < ageIndexu)){
                          dist = dist + (Math.pow(fit-obs[ii], (double)2) / (cases_adj[ii] /
(Math.pow(pop[ii], (double)2))));
                        }
                        else if (ii < ageIndexl)
                          dist = dist + (Math.pow(fit-obs[ii], (double)2) / (cases_adj[ageIndexl] /
(Math.pow(pop[ageIndexl], (double)2))));
                        else if (ageIndexu < nobs)
                          dist = dist + (Math.pow(fit-obs[ii], (double)2) / (cases_adj[ageIndexu] /
(Math.pow(pop[ageIndexu], (double)2))));
                      if (windicator != 1)
                          dist = Math.max(dist, Math.abs(fit-obs[ii])) ;
                      if (dist >= disto)
                        break;
                    }
                    if (dist < disto) {
                      disto = dist;
                      Fraco = Frac;
                      fo = f;
                      co = c;
```

```
                        alminbeo = alminbe;
                        betao = beta;
                        rAo = rA;
                        slpo = slpmax;
                        detao = deta;
                      }
                    }
                  }
                  fileWriter.write(padString(Fraco +"", padLength) + " " + padString(fo+"", padLength) + "
" + padString(co + "", padLength) + " " + padString(alminbeo + "", padLength) + " " + padString(betao +
"", padLength) + " " + padString(rAo + "", padLength) + " " + padString(disto + "", padLength) + " " +
padString(slpo + "", padLength) + " " + padString(detao +"", padLength) + "\n");

                }
              }
            }
          }
          fileWriter.close();
        } catch (Exception e) {
          System.out.println("File error in evalgrid.");
          e.printStackTrace();
        }
        //we are done
        bestFitWindow.setProgress(100);
        return;
      }


/* This method takes a file (labeled 7 in the Fortran code), along with
   values for slopemax and detamax, and examines the file for the best fit.
 */

    // Note the changed parmeter list!!!

    public void bestfits(File file, double slpm, double detam) {

      try {
        BufferedReader fileReader = new BufferedReader(new FileReader(file));
        String line = fileReader.readLine();

        double dist = 1e30;
        while(line != null) {
          StringTokenizer tokenizer = new StringTokenizer(line);

          double Frac = Double.parseDouble(tokenizer.nextToken());
          double f = Double.parseDouble(tokenizer.nextToken());
          double c = Double.parseDouble(tokenizer.nextToken());
          double brth = Double.parseDouble(tokenizer.nextToken());
          double deth = Double.parseDouble(tokenizer.nextToken());
          double rA = Double.parseDouble(tokenizer.nextToken());
          double di = Double.parseDouble(tokenizer.nextToken());
          double slp = Double.parseDouble(tokenizer.nextToken());
          double deta = Double.parseDouble(tokenizer.nextToken());

          if (di < dist && slp > slpm && deta > detam) {
            dist = di;
```

```
            dio = di;
            Fraco = Frac;
            fo = f;
            co = c;
            brtho = brth;
            detho = deth;
            rAo = rA;
            slpo = slp * (double)100000;
            detao = deta;
          }

          line = fileReader.readLine();
        }
      } catch (Exception ex) {
        System.out.println("I/O error in bestfits");
        //uncomment for debugging
        System.out.println(ex.toString());
        ex.printStackTrace();
      }
      return;
}

// Note the changed parameter list!!!

public void goodfits(File file,  double slpm, double detam) {
      nfits = 0; //We have no fits yet.
      try {
        BufferedReader fileReader = new BufferedReader(new FileReader(file));
        String line = fileReader.readLine();
        solutions[0][6] = 0.0 ;

        while(line != null) {
          StringTokenizer tokenizer = new StringTokenizer(line);

          double Frac = Double.parseDouble(tokenizer.nextToken());
          double f = Double.parseDouble(tokenizer.nextToken());
          double c = Double.parseDouble(tokenizer.nextToken());
          double brth = Double.parseDouble(tokenizer.nextToken());
          double deth = Double.parseDouble(tokenizer.nextToken());
          double rA = Double.parseDouble(tokenizer.nextToken());
          double di = Double.parseDouble(tokenizer.nextToken());
          double slp = Double.parseDouble(tokenizer.nextToken());
          double deta = Double.parseDouble(tokenizer.nextToken());

          //take only the top 20
          //first we fill the first 20 sorted by dist
          //if ((slp > slpm) && (deta > detam) && (di != 0) && (di < (dio * 1.2))){
          //if (di < (dio * 1.2)){
          //changed here
          if (nfits < 100){
            int i = 0 ;
            while ((i < nfits) && (di > solutions[i][6]))
              i++ ;
            if (i == nfits){   //insert at the end
              solutions[i][0] = Frac;
              solutions[i][1] = f;
```

```
      solutions[i][2] = c;
      solutions[i][3] = brth;
      solutions[i][4] = deth;
      solutions[i][5] = rA;
      solutions[i][6] = di;
      solutions[i][7] = slp * (double)100000;
      solutions[i][8] = deta;
      nfits++;
    }
    else { //the solutions fits into the middle someplace
      int j = 0 ;
      for (j = nfits; j > i; j--){
        solutions[j][0] = solutions[j-1][0] ;
        solutions[j][1] = solutions[j-1][1] ;
        solutions[j][2] = solutions[j-1][2] ;
        solutions[j][3] = solutions[j-1][3] ;
        solutions[j][4] = solutions[j-1][4] ;
        solutions[j][5] = solutions[j-1][5] ;
        solutions[j][6] = solutions[j-1][6] ;
        solutions[j][7] = solutions[j-1][7] ;
        solutions[j][8] = solutions[j-1][8] ;
      }
      solutions[i][0] = Frac;
      solutions[i][1] = f;
      solutions[i][2] = c;
      solutions[i][3] = brth;
      solutions[i][4] = deth;
      solutions[i][5] = rA;
      solutions[i][6] = di;
      solutions[i][7] = slp * (double)100000;
      solutions[i][8] = deta;
      nfits++;
    }
  }
  //Now we only care if a solution has a smaller distance than one of those
  //in the solutions array.  Since it is sorted it has to be smaller than the last one
  //we also need to preserve the sorted order
  else {
          if (di < solutions[nfits-1][6]){
      //find the first dist that is smaller than it
      int j = nfits - 2;
      while ((j > 0) && (di < solutions[j][6]))
        j-- ;
      //now shift all solutions down and place the new one in
      for (int k = nfits-1; k > j; k--){
        solutions[k][0] = solutions[k-1][0] ;
        solutions[k][1] = solutions[k-1][1] ;
        solutions[k][2] = solutions[k-1][2] ;
        solutions[k][3] = solutions[k-1][3] ;
        solutions[k][4] = solutions[k-1][4] ;
        solutions[k][5] = solutions[k-1][5] ;
        solutions[k][6] = solutions[k-1][6] ;
        solutions[k][7] = solutions[k-1][7] ;
        solutions[k][8] = solutions[k-1][8] ;
      }
      solutions[j][0] = Frac;
```

```
                solutions[j][1] = f;
                solutions[j][2] = c;
                solutions[j][3] = brth;
                solutions[j][4] = deth;
                solutions[j][5] = rA;
                solutions[j][6] = di;
                solutions[j][7] = slp * (double)100000;
                solutions[j][8] = deta;
              }
            }
            //}

            line = fileReader.readLine();
          }
          //bubble sort
              // need to make it sort by increasing distance
        double[] temp = new double[9] ;
        for (int i = nfits ; i > 0 ; i--)
          for (int j = 0; j < i -1; j++)
              if (solutions[j][6] > solutions[j+1][6]){ // was solutions[j][0] > solutions[j+1][0]
              temp = solutions[j] ;
              solutions[j] = solutions[j+1] ;
              solutions[j+1] = temp ;
            }
        } catch (Exception ex) {
          System.out.println("I/O error in goodfits");
          //uncomment for debugging
          System.out.println(ex.toString());
          ex.printStackTrace();
        }

        System.out.println("done");
        return;
    }


    /* This method computes the incidence rates of the fraction
      at risk model for carcinogenesis.
     */
    public void survival(double[] age, double[] pobs, double agemax) {

        //Determines the number of points returned.

        double dt = agemax / (double)ngrid;
        double t, u, x1, x2, ff;

        for (int i = 0; i < ngrid - 1; i++) {
          t = i * dt;
          x1 = 0.0;
          x2 = t;
          ff = -1.0 * DL(x1) * Sprime(x2) / 2.0;
          for (int j = 0; j < igrid - 1; j++) {
            u = (double)j / (double)igrid;
            x1 = t * u;
            x2 = t * (1.0 - u);
            ff = ff - DL(x1)*Sprime(x2);
```

```
      }
      x1 = t;
      x2 = 0.0;
      ff = ff - DL(x1) * Sprime(x2) / 2.0;
      ff = t * ff/(double)igrid;
      pobs[i + 1] = ff;
      age[i + 1] = t;
    }
    inthA[0] = pobs[0];
    for(int i = 1; i < ngrid; i ++ ) {
      inthA[i] = inthA[i- 1] + pobs[i];
    }
    for (int i = 1; i <= ngrid - 1; i++) {
      inthA[i] = inthA[i] - (pobs[i] + pobs[0]) / 2.0;
      pobs[i] = pobs[i] = pobs[i] * Frac / (Frac + (1.0 - Frac) * Math.exp(inthA[i] * dt / f));
    }
    for(int i =0; i < ngrid - 1; i ++) {
      System.out.println(age[i] + " " + pobs[i]);

    }

    return;
}

    /* The N array is the growth of cells at risk as a function of age.
     * It is taken from the data in Figure 7 of Dr. Pablo Herrero's Thesis.
     * gamma is a factor that effects the growth equations by account for
     * the topology of the organ being analyzed.  nMax is the maximum number
     * of cells at risk over the lifetime.  initiateableAge is the last age
     * at which initiation can occur.
     */
    public void setupNArray(double gamma, double nMax, int iA, boolean male){
        setupGrowthFunction = true; // indicate that Growth Function (N array) has been setup
        NMax = nMax ;
      initiateableAge = iA ;
      if (male) {
        for (int i = 0; i < 2; i++)
          N[i] = nMax / Math.pow(2, ((0.16*gamma*15) + (1.2*gamma) * (1.5 - i)));

        for (int i = 2; i < 17; i++)
          N[i] = nMax / Math.pow(2, (0.16*gamma*(16.5 - i))) ;

        for (int i = 17; i < ageMaxForN; i++)
          N[i] = nMax ;
      }
      else {
        for (int i = 0; i < 2; i++)
          N[i] = nMax / Math.pow(2, ((0.16*gamma*15) + (1.2*gamma) * (1.5 - i)));

        for (int i = 2; i < 15; i++)
          N[i] = nMax / Math.pow(2, (0.16*gamma*(14.5 - i))) ;

        for (int i = 15; i < ageMaxForN; i++)
          N[i] = nMax ;
      }
    }
```

```
    /* Calculates the value of Pinit for use in calculating the X factor
     * in the expression F = GEX and then returns X.
     */
    public double calculateX(int age, double amboa, double cinit){
      double intPinit = 0.0;
      for (int i = 0; i <= age; i++)
        intPinit = intPinit + (N[i]*cinit/NMax*amboa*i) ;
      return (1 - Math.pow(Math.E, -1.0 * intPinit)) ;
    }


  /*************************************************************************/
  /************************ NON-MATH ***********************************/
  /*************************************************************************/
  //graphLine is overloaded.  if the last element is a int, then that represents
  //the number of the line to use.  Otherwise, it takes one less argument and
  //graph line makes a new line with a random color.

  public void graphLine(double F, double f, double cinit, double aminusb, double beta, double rA, double
slp, double deta, double population) {

    brth = Math.log(2) * (beta + aminusb);
    deth = Math.log(2) * (beta);
    this.Frac = F;
    this.f = f;
    this.c = cinit;
    this.brth = brth;
    this.deth = deth;
    this.rA = rA;
    this.slopemax = slp * Math.pow(10,5);
    this.detamax = deta;
    twostage(nobs);

    BufferedImage img = new BufferedImage(5,5, BufferedImage.TYPE_INT_RGB);
    Graphics2D g2 = img.createGraphics();
    g2.setColor(Color.blue);
    g2.fill(new java.awt.geom.Ellipse2D.Float(0,0, 5, 5));
    int linenumber = graph.addLine(img);

    double X = 1.0 ;
    if (newEquation){
      X = calculateX(initiateableAge, (brth-deth)/brth, cinit) ;
    }
    for(int ii=0; ii<nobs+extension; ii++) {
      double fit = Pobs[ii] * Frac/ (Frac + ((double)1 - Frac) * Math.exp((1-
survivalFraction[ii])*intPobs[ii] / f)) * population;
      graph.addPoint(linenumber, t[ii], fit);
    }
    graph.repaint();
  }

  //overloaded, allows you to specify the line number
  public void graphLine(double F, double f, double cinit, double aminusb, double beta, double rA, double
slp, double deta, double population, int lineNum) {
```

```
      brth = Math.log(2) * (beta + aminusb);
      deth = Math.log(2) * (beta);
      this.Frac = F;
      this.f = f;
      this.c = cinit;
      this.brth = brth;
      this.deth = deth;
      this.rA = rA;
      this.slopemax = slp * Math.pow(10,5);
      this.detamax = deta;
      twostage(nobs);

      double X = 1.0 ;
      if (newEquation){
        X = calculateX(initiateableAge, (brth-deth)/brth, cinit) ;
      }
      tempArea = 0.0 ;
      for(int ii=0; ii<nobs+extension; ii++) {
        double fit = Pobs[ii] * Frac/ (Frac + ((double)1 - Frac) * Math.exp((1-
survivalFraction[ii])*intPobs[ii] / f)) * population;
        tempArea = tempArea + fit ;
        graph.addPoint(lineNum, t[ii], fit);
      }

      graph.repaint();
    }

    //overloaded for graphSingleFitWindow returns the RSS distance from the data
    public double graphLine(double F, double f, double cinit, double aminusb, double beta, double rA,
double slp, double deta, double population, int lineNum, boolean different) {

      brth = Math.log(2) * (beta + aminusb);
      deth = Math.log(2) * (beta);
      this.Frac = F;
      this.f = f;
      this.c = cinit;
      this.brth = brth;
      this.deth = deth;
      this.rA = rA;
      this.slopemax = slp * Math.pow(10,5);
      this.detamax = deta;
      twostage(nobs);

      double dist = 0.0 ;
      double X = 1.0 ;
      if (newEquation){
        X = calculateX(initiateableAge, (brth-deth)/brth, cinit) ;
      }
      tempArea = 0.0 ;
      for(int ii=0; ii<nobs+extension; ii++) {
        double fit = Pobs[ii] * Frac/ (Frac + ((double)1 - Frac) * Math.exp((1-
survivalFraction[ii])*intPobs[ii] / f)) * population;
            //   fit = fit / ((1-survivalFraction[ii])*(reportError[ii])*(1-tot[ii]));
        tempArea = tempArea + fit ;
        graph.addPoint(lineNum, t[ii], fit);
```

```
        if (ii < nobs)
          dist = dist + (Math.pow((fit/population)-obs[ii], (double)2)) ;
      }


        //Also graph Pobs - commented out 04/21/04- don't want to graph Pobs on main graph
        /* BufferedImage img = new BufferedImage(5,5, BufferedImage.TYPE_INT_RGB);
      Graphics2D g2 = img.createGraphics();
      g2.setColor(Color.green);
      g2.fill(new java.awt.geom.Ellipse2D.Float(0,0, 5, 5));
      int linenumber = graph.addLine(img);

      //remove old Pobs
      if (pobsline != -12)
        graph.removeLine(pobsline) ;
      pobsline = linenumber ;

      for(int ii=0; ii<nobs; ii++) {
        graph.addPoint(linenumber, t[ii], Frac*Pobs[ii]*population) ;
      }
      */
      //get the new Max Slope ('S') and Delta
      double newSlope = 0.0;
      double newDelta = 0.0;
      double age = 0.0;
      for(int j = 1; j<nobs; j++){
        double tempSlope = (Pobs[j+1]*population - Pobs[j]*population) / (t[j+1] - t[j]);
        if (tempSlope > newSlope){
          newSlope = tempSlope ;
          newDelta = t[j] - (Pobs[j]*population / newSlope) ;
          age = (j-1)*5 - 2.5;
        }
      }

      graph.repaint();
      return dist ;
    }

    //overloaded, allows you to specify the line number and the number of data points, and the max age.
    public void graphLine(double F, double f, double cinit, double aminusb, double beta, double rA, double
slp, double deta, int lineNum, int numPoints, int maxAge) {

      brth = Math.log(2) * (beta + aminusb);
      deth = Math.log(2) * (beta);
      this.Frac = F;
      this.f = f;
      this.c = cinit;
      this.brth = brth;
      this.deth = deth;
      this.rA = rA;
      this.slopemax = slp * Math.pow(10,5);
      this.detamax = deta;
      double[] temp1 = new double[numPoints];
      double[] temp2 = new double[numPoints];
      int ngridOld = ngrid;
      ngrid = numPoints - 1;
```

```
        survival(temp1, temp2, 110.0);
        System.out.println("Uses survival in graphLine around line 2900") ;

        for(int ii=0; ii<numPoints; ii++) {
          graph.addPoint(lineNum, temp1[ii], temp2[ii] * Math.pow(10, 5));

        }
        for(int i =0; i < ngrid - 1; i ++) {
          System.out.println(temp1[i] + " " + temp2[i]);

        }
        ngrid = ngridOld;
        graph.repaint();
    }


    //graph Pobs
    public void graphPobs(double F, double f, double cinit, double aminusb, double beta, double rA, double
slp, double deta, double population) {
        brth = Math.log(2) * (beta + aminusb);
        deth = Math.log(2) * (beta);
        this.Frac = F;
        this.f = f;
        this.c = cinit;
        this.brth = brth;
        this.deth = deth;
        this.rA = rA;
        this.slopemax = slp * Math.pow(10,5);
        this.detamax = deta;
        twostage(40);

        BufferedImage img = new BufferedImage(5,5, BufferedImage.TYPE_INT_RGB);
        Graphics2D g2 = img.createGraphics();
        g2.setColor(Color.green);
        g2.fill(new java.awt.geom.Ellipse2D.Float(0,0, 5, 5));
        int linenumber = graph.addLine(img);

        double X = 1.0 ;
        if (newEquation){
          X = calculateX(initiateableAge, (brth-deth)/brth, cinit) ;
        }
        for(int ii=0; ii<40; ii++) {
          graph.addPoint(linenumber, t[ii], Frac*Pobs[ii]*population) ;
        }

        graph.repaint();
    }


     public double getArea(double F, double f, double cinit, double aminusb, double beta, double rA,
double slp, double deta) {
        brth = Math.log(2) * (beta + aminusb);
        deth = Math.log(2) * (beta);
        this.Frac = F;
        this.f = f;
        this.c = cinit;
```

```
        this.brth = brth;
        this.deth = deth;
        this.rA = rA;
        this.slopemax = slp * Math.pow(10,5);
        this.detamax = deta;
        twostage(40);

        double area = 0.0 ;
        double fit ;
        for (int i = 0; i < 40; i++){
              // System.out.println("Frac : " + Frac + "survalFraction[i] : " + survivalFraction[i] + "intPobs[i]
: " + intPobs[i] + "f: " + f + "total : " + (Frac+ (1.0 - Frac) * Math.exp((1-survivalFraction[i])*intPobs[i] /
f)));
            fit = Pobs[i] * Frac/ (Frac+ (1.0 - Frac) * Math.exp((1-survivalFraction[i])*intPobs[i] / f));
                //fit = fit / ((1-survivalFraction[i])*(reportError[i])*(1-tot[i]));
                //System.out.println("Area: " + area);
                //System.out.println("Fit : " + fit);
            area = area + fit;
        }
            //double area_temp;
            //area_temp = area / 100000.0;
            //System.out.println("Area: " + area_temp);
        return area ;
      }

  // add function below to allow survivalFraction array to have one value (for ease in setting defaults)
  public void setupSurvivalArray(double def){
    for(int i = 0; i < 50; i++)
      survivalFraction[i] = def;
  }

  // add function below to allow report_error
  public void setupReportErrorArray(double def){
    for(int i = 0; i < 50; i++)
      reportError[i] = def;
  }

  // add function below to initialize tot array
  public void setupTOTArray(double def){
    for(int i = 0; i < 50; i++)
      tot[i] = def;
  }



    /** @Returns a JPanel with the first compenent on the west and the second
     * compenent on the east of a BorderLayout Panel. */
    public static JPanel createEastWestPanel(JComponent west, JComponent east) {
        JPanel temp = new JPanel();
        temp.setLayout(new BorderLayout());
        temp.add(west, BorderLayout.WEST);
        temp.add(east, BorderLayout.EAST);
        return temp;
    }

    /** @Returns a String identical to the one passed in with a certain number
```

```
 * of decimal places. or null if the string passed in is null. */
public static String truncString(String string, int decimalPlaces) {
   String returner ;
   if(string == null) {
      returner = null;
   }
   int index = string.indexOf(".");
   if(index == -1){
      returner = string;
   } else if (decimalPlaces == 0) {
      returner = string.substring(0, index);
   }
   String tempString = string;
   if(string.length() > index + decimalPlaces + 1) {
      tempString = string.substring(0, index + decimalPlaces + 1);
   }
   returner = tempString;

   int indexOfE = string.indexOf("E");
   if(indexOfE == -1)
      return returner ;
   else return (returner + string.substring(indexOfE));
}


/** Runs a Save Dialog Box so the user can select the file to save to
 *  And writes the file */
private void runSaveDialog(int whichOne) {
   JFileChooser chooser = new JFileChooser();
   chooser.setApproveButtonText("Save");

   //display the save dialog;
   int returnVal = chooser.showOpenDialog(Fit.this);

   if (returnVal == JFileChooser.APPROVE_OPTION) {
      String filename = chooser.getSelectedFile().getAbsolutePath();
      try {
         //Clear the file if it exists, I should check to see if
         //the user wants to overwrite the file, I will do this
         //later if I have time.
         File file = new File(filename);
         file.delete();
         FileWriter writer = new FileWriter(filename, true);
         BufferedWriter fileWriter = new BufferedWriter(writer);
         if (whichOne == 1)
            fileWriter.write(dataFileView.dataView.getText());
         else
            fileWriter.write(dataFileView2.dataView.getText());
         fileWriter.close();

      } catch (Exception e2) {
         throw new RuntimeException("Error writing file");
      }
   }
}
```

```
  /** Runs a Save Dialog Box so the user can select the name of the file
   * to save the results to and writes the file */
  private void runSaveDialog2() {
    JFileChooser chooser = new JFileChooser();
    chooser.setApproveButtonText("Save");

    //display the save dialog;
    int returnVal = chooser.showOpenDialog(Fit.this);

    if (returnVal == JFileChooser.APPROVE_OPTION) {
      String filename = chooser.getSelectedFile().getAbsolutePath();
      try {
        //Clear the file if it exists, I should check to see if
        //the user wants to overwrite the file, I will do this
        //later if I have time.
        File file = new File(filename);
        file.delete();
        FileWriter writer = new FileWriter(filename, true);
        BufferedWriter fileWriter = new BufferedWriter(writer);
        //get all of the data needed to save
        fileWriter.write("The Growth Function\n") ;
        fileWriter.write("Gamma:  " + setParametersWindow.gammaBox.getText()+"\n") ;
        fileWriter.write("NMax:  " + setParametersWindow.nMaxBox.getText()+"\n") ;
        fileWriter.write("Initiateable Age:  " + setParametersWindow.initiateableAgeBox.getText()+"\n")
;

        fileWriter.write("\nThe Upper and Lower Limits"+"\n") ;
        fileWriter.write("F:  " + bestFitWindow.FruBox.getText() + "    " +
bestFitWindow.FrlBox.getText()+"\n") ;
        fileWriter.write("f:  " + bestFitWindow.fuBox.getText() + "    " +
bestFitWindow.flBox.getText()+"\n") ;
        fileWriter.write("cInit:  " + bestFitWindow.cuBox.getText() + "    " +
bestFitWindow.clBox.getText()+"\n") ;
        fileWriter.write("Alpha-Beta:  " + bestFitWindow.alminbeuBox.getText() + "    " +
bestFitWindow.alminbelBox.getText()+"\n") ;
        fileWriter.write("cProm:  " + bestFitWindow.rAuBox.getText() + "    " +
bestFitWindow.rAlBox.getText()+"\n") ;
        fileWriter.write("Beta:  " + bestFitWindow.betauBox.getText()+"\n") ;

        fileWriter.write("\nThe Looping Values"+"\n") ;
        fileWriter.write("F:  " + bestFitWindow.Frnum.getText()+"\n") ;
        fileWriter.write("f:  " + bestFitWindow.fnum.getText()+"\n") ;
        fileWriter.write("cInit:  " + bestFitWindow.cnum.getText()+"\n") ;
        fileWriter.write("Alpha-Beta:  " + bestFitWindow.alminbenum.getText()+"\n") ;
        fileWriter.write("cProm:  " + bestFitWindow.rAnum.getText()+"\n") ;

        fileWriter.write("\nThe Fits"+"\n") ;
        fileWriter.write("F        f        cInit        Alpha-Beta    Beta        cProm        Distance    X \n")
;
        for (int i = 0; i < nfits; i++){
          for (int j = 0; j < 7; j++){
            if ((minit == 0.0) && (j == 5))
              fileWriter.write(sortedsolutions[i][5] / ((sortedsolutions[i][3] + sortedsolutions[i][4]) /
sortedsolutions[i][3]) + "    ") ;
            fileWriter.write(sortedsolutions[i][j]+"    ") ;
          }
```

```
            fileWriter.write(truncString(Double.toString(calculateX(initiateableAge,
sortedsolutions[i][3]/(sortedsolutions[i][3]+sortedsolutions[i][4]), sortedsolutions[i][2])), 4)) ;
            fileWriter.write("\n");
          }


          fileWriter.close();

        } catch (Exception e2) {
          throw new RuntimeException("Error writing file");
        }
      }
    }


    /** Creates a popup Menu.
     * @requires: every element in Vec to be a String.
     * @returns: a popup menu whose entries are in order of Vec and have the text
     * of vec and have the listener as their listener.  If a string = DIVISION
     * then a divider bar is put in instead of the word DIVISION
     */
    private static JPopupMenu createPopupMenu(Vector vec, ActionListener listener) {
      JPopupMenu menu = new JPopupMenu();

      for(int i=0; i < vec.size(); i ++){
        String text = (String)vec.get(i);
        if(text.compareTo("DIVISION") == 0) {
          menu.addSeparator();
        } else {
          JMenuItem item = new JMenuItem(text);
          item.addActionListener(listener);
          menu.add(item);
        }
      }
      return menu;
    }

    private JPopupMenu initPopupMenu(){
      JPopupMenu menu = new JPopupMenu();
      JMenuItem menuItem = new JMenuItem("Add Line to Group Fits Window");
      menuItem.addActionListener(new PopupMenuListener());
      menu.add(menuItem);
      menuItem = new JMenuItem("Add All Lines to Group Fit Window");
      menuItem.addActionListener(new PopupMenuListener());
      menu.add(menuItem);
      menuItem = new JMenuItem("Add Line to A-B Mod Window");
      menuItem.addActionListener(new PopupMenuListener());
      menu.add(menuItem);
      menu.addSeparator();
      menuItem = new JMenuItem("Re-run Good Fits with this line");
      menuItem.addActionListener(new PopupMenuListener());
      menu.add(menuItem);
      menuItem = new JMenuItem("Delete Line");
      menuItem.addActionListener(new PopupMenuListener());
      menu.add(menuItem);
```

```
    return menu;
  }


  /** takes in a string and returns the first double number it finds. It
   * deletes everything up to and including the first number it finds from
   * the buffer. **/
  double getDouble(StringBuffer stringBuffer) {
    int start, end;
    int cntr = 0;

    while(Character.isDigit(stringBuffer.charAt(cntr)) != true){
      cntr++;
    }
    start = cntr;

    while(Character.isWhitespace(stringBuffer.charAt(cntr)) != true) {
      cntr++;
    }

    end = cntr;
    double retval = Double.parseDouble(stringBuffer.substring(start, end));
    stringBuffer.replace(start, end, "");
    return retval;
  }

  /** Pad String takes a String and an int.  If the String is longer than int,
   * the string is returned, otherwise, a number of blank spaces is added to
   * the end of the string so that the returned string is equal to int. */
  String padString(String string, int length) {
    int lngth = string.length();
    for(int i = lngth; i < length; i++){
      string = string + " ";
    }
    return string;
  }

  /********************** Main ***************************/
  public static void main(String[] args) {
    Fit window = new Fit();
    window.setTitle("CancerFit Version " + Fit.versionNumber);
    window.setSize(930, 600);
    window.setVisible(true);
  }
}
```

References

[1] P. Artimage and R. Doll. A two-stage theory of carcinogenesis in relation to the age distribution of human cancer. *British Journal of Cancer*, 1957.

[2] H.A. Coller, K. Khrapko, N.D. Bodyak, E. Nekhaeva, P. Herrero-Jimenez, and W.G. Thilly. High frequency of homoplasmic mitochondrial DBA mutations in human tumors can be explained without selection. *Nature Genetics*, 2001.

[3] K. Hemminki and P. Boffetta. Multiple primary cancers as clues to environmental and heritable causes of cancer and mechanisms of carcinogenesis. *Science Publication*, 2004.

[4] K. Hemminki and X. Li. Familial and second primary pancreatic cancers: a nationwide epidemiologic study from Sweden. *Cancer*, 2003.

[5] D. Hensle. Computation of Population and Physiological Risk Parameters from Cancer Data. MIT Thesis, 2003.

[6] P. Herrero-Jimenez, S. Morgenthaler, and W.G. Thilly. Analysis of lung cancer mortality rates and cigarette use in the United States. *Mutation Research*, 2000.

[7] P. Herrero-Jimenez, G. Thilly, P.J. Southam, A. Tomita-Mitchell, S. Morgenthaler, E.E. Furth, and W.G. Thilly. Fundamental and Molecular Mechanisms of Mutagenesis. *Mutation Research*, 1998.

[8] P. Herrero-Jimenez and W.G. Thilly. Mortality Data. http://epidemiology.mit.edu, MIT, 2000.

[9] P. Herrero-Jimenez, A. Tomita-Mitchell, E.E. Furth, S. Morgenthaler, and W.G. Thilly. Population risk and physiological rate parameters for colon cancer. The union of an explicit model for carcinogenesis with the public health records of the United States. *Mutation Research Frontiers*, 1999.

[10] S.H. Moolgavkar, A. Dewanji, and D.J. Venzon. A stochastic two-stage model for cancer risk assessment. *Risk Analysis*, 1988.

[11] S.H. Moolgavkar and A.G. Jr. Knudson. Mutation and cancer: A model for human carcinogenesis. *Journal of the National Cancer Institute*, 1981.

[12] S.H. Moolgavkar and D.J. Venzon. Two-event models for carcinogenesis: Incidence curves for childhood and adult tumors. *Mathematical Biosciences*, 1979.