Computation of Population and Physiological Risk Parameters from Cancer Data

by

David Hensle

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degrees of

Bachelor of Science in Electrical Engineering and Computer Science

and

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 2003

Author .....
Department of Electrical Engineering and Computer Science
May 9, 1990

Certified by .....
William G. Thilly
Professor, Biological Engineering
Thesis Supervisor

Accepted by .....
Arthur C. Smith
Chairman, Department Committee on Graduate Students

Computation of Population and Physiological Risk Parameters from Cancer Data

by

David Hensle

Submitted to the Department of Electrical Engineering and Computer Science
on May 9, 1990, in partial fulfillment of the
requirements for the degrees of
Bachelor of Science in Electrical Engineering and Computer Science
and
Master of Engineering in Electrical Engineering and Computer Science

Abstract

A biological hypothesis known as the "two-stage cancer model" conceptualizes carcinogenesis in mathematical terms. Prof. W. Thilly (MIT) and Prof. S. Morgenthaler (ETH,) and their students, especially Dr. Pablo Herrero-Jimenez have organized age- and birth cohort-specific cancer mortality data from the United States (1900-1997). Unfortunately, no computer program exists that permits cancer researchers to develop and explore quantitative hypotheses about the values of biological parameters through the analysis of this large and comprehensive data set.

In this thesis, initial efforts of the Morgenthaler and Thilly groups have been extended by improving and transporting a preliminary Fortran program incorporating an approximate model into a Java-based program. This program, CancerFit, can be used by cancer researchers without backgrounds in mathematics or computer programming. CancerFit is the first of its kind that deals with the carcinogenic process as a whole, rather than in smaller, individual components.

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

Acknowledgments

I would like to thank Professor W. Thilly for being the driving force behind and my advisor for this work. Professor S. Morgenthaler also deserves a great deal of my thanks for providing the groundwork and origins of this program in addition to helping with the mathematical reasoning throughout the process. And finally, I would like to acknowledge the significant contributions of Aaron Fernandes and Jonathan Jackson in the initial port of the Fortran program into the current Java platform, which began as an 8.672 project.

Contents

List of Figures

List of Tables

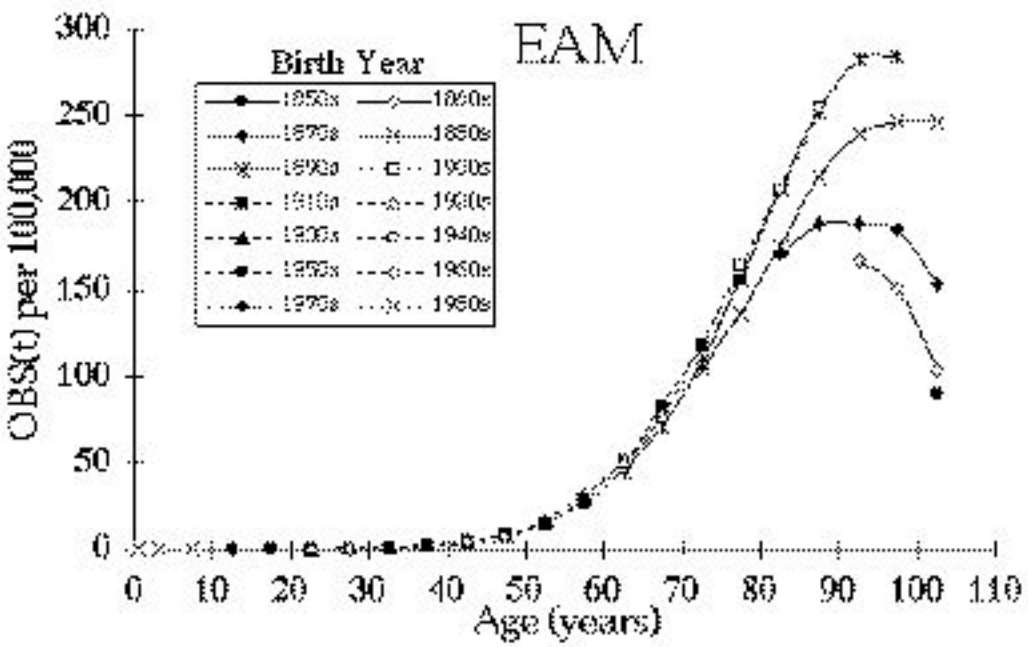

Figure 1: Observed mortality rates from colon cancer for European American Males

1 Introduction and Preliminary Work

A gap exists between historically recorded cancer mortality data and the current probabilistic models used to explain the molecular and cellular biology of cancer. Herrero-Jimenez et al., (see [6, 7] extended the algebraic models for cancer initiation and promotion developed by others, principally Nordling [14], Richard Doll [1, 2], Knudson [8] and Moolgavkar [13, 10, 9, 11, 12, 3, 4]. Herrero-Jimenez and his collaborators specifically introduced algebra to account for the possibility that members of large populations display heterogeneity with regard to genetic and environmental risk factors and that any environmental risk factors but not inherited genetic factors are expected to have changed significantly in recent human history. The program resulting from this thesis, CancerFit, is a Java implementation of a FORTRAN application developed by Professor Stephan Morgenthaler of cole Polytechnique Fdrale de Lausanne, modified to explore these hypotheses.

It has been observed that the mortality rate by most types of cancer as a function of age is low up to age 50, rises approximately linearly from age 60 to 85, reaches a maximum around age 90, and then decreases significantly by age 100 (see Figure 1).

There exist a number of possible explanations for the rise and fall in these curves. For example, the model of Herrero-Jimenez et al. assumed that the rise resulted from a linear increase in the rate of appearance of initiated colonies, and the fall was due to the depletion of a specific at-risk population [6]. In other words, if there exists a subpopulation at risk for colon cancer that subpopulation should show an age-specific death rate greater than that in a subpopulation not at risk of colon cancer. This hypothesis is put forward to explain the maximum point of mortality in the age specific colon cancer death rate as seen in Figure 1 [6].

2 Background

Under the existing model, Herrero-Jimenez et al. posited that certain types of cancer arise only in an at-risk subpopulation as a result of inherited and environmental risk factors. Adopting the notation of Herrero-Jimenez et al. [6, 7] and denoting the fraction of the population at-risk for a particular form of cancer, e.g. intestinal cancer, as "F". The fraction of this subpopulation that would have died from cancer, had they not died of a related disease sharing the same risk factors as the defined F, as "f". Furthermore, within this subpopulation at risk, two-stage carcinogenesis was modeled as a series independent and sequential "n" initiation and "m" promotion events. Initiation was modeled as the accumulation of "n" genetic events occurring in the same normal cell, giving rise to the first cell in a preneoplastic lesion, e.g. an adenoma in the colon. In the surviving, slow-growing adenoma (preneoplastic colony), promotion wass modeled as the acquisition of "m" rare, but not necessarily genetic, events causing an adenoma cell to become cancerous. To model the growth of a preneoplastic colony the parameters "α" and "β", correspond to cell division and death rates, respectively, leading to the parameter for net growth rate, α - β. Another algebraic parameter, (α - β)/α , represented the probability that a newly initiated cell will give rise to a surviving, growing colony using the hypothetical case wherein all cells in a preneoplastic colony can divide, die and give rise to a neoplastic cell with equal probability. A variant hypothesis, the "stem cell only" hypothesis considers the possibility that only stem cells are initiated and continue as stem cells in preneoplasia with only stem cells within the preneoplastic colony being capable of promotion. In this specific case β=0 is inserted into the computations to permit exploration of this biologically interesting hypothesis developed by Drs. Elena Gostjeva and William Thilly.

2.1 Subpopulation(s) at risk

These definitions arise from a model of carcinogenesis in which inherited and environmental factors can either provide provide a condition necessary for carcinogenesis

or modify the rate of acquisition of such a condition. With regard to necessary conditions, two fractions are posited that a fraction, "G", of the population inherited a required genetic condition and a fraction, "E", experienced a required environmental condition.

While in general, $F = G_1E_1 + G_2E_2 + ... - (G_1E_1)(G_2E_2) - ...$, this possibility is abbreviated as F = GE at this stage of constructing a general cancer model.

A second factor "f" represents the hypothesis that the same conditions of inherited risk and environmental risk will lead to the death of an individual by a different mode than the one under study. "f" is specifically the fraction of a birth cohort at risk (within subpopulation F) that would have been expected to die of the observed disease during a lifespan of ∼140 years *given no other forms of death other than those created by the factors determining F.*

2.2 The Two Stage Model

In this model the carcinogenesis process is divided into two rate limiting stages: "initiation" and "promotion".

2.2.1 Initiation

Initiation is the process by which a normal cell is becomes an initiated cell. The phenotypic difference acquired in initiation is the ability of the mutated cell to give rise to a slowly growing "prenoplastic" colony. The initiation process can occur in an unknown subset of cells in a tissue in which the maximum number of such cells in a growing individual is defined by N_{max}. It should be noted that "cell at risk" is to date biologically undefined. The CancerFit program permits researchers to explore wide numerical values such as the set of all stem cells or the set of all cells. Built into CancerFit is the mass of boys and girls as a function of age permitting the number of cells at risk to be modeled as increasing from birth to maturity. The required "n" events of initiation, i, j, ...n each occurring at rate $R_i, R_j, ...R_n$ events per cell year or $r_i\tau, r_j\tau, ...r_j\tau$ events per cell division where τ is the number of stem cell turnover

divisions per year. The notion of a turnover unit is an important concept in the modeling of the carcinogenic process.

The Turnover Unit The basic structure of a turnover unit is a single stem cell that gives rise to some number of cells that are approximated by binomial growth and eventually end in a terminal layer. The exact size of a turnover unit varies depending on the tissue being studied and may vary within a tissue as a function of anatomical position. A general model of a turnover unit and how it might grow into a preneoplastic colony is depicted below in Figure 2.

Transitional Probability of Survival Once a cell is initiated it is posited to have a division rate "α" and a death rate "β"' such that (α - β) is the net growth rate of the initiated colony. However, since the early death of the initial or the first few cells would eliminate such a colony, the probability of such a colony surviving is (α - β) / α and that the average number of cell equivalents is the surviving colony would not be 1 cell, but rather α / (α - β) cells.[13]

(In one variation of the cancer hypothesis, only stem cells are at risk for which β = 0 and therefore (α - β) / α = 1.)

2.2.2 Promotion

This stage is imagined as the slow growth of a preneoplastic colony in which cells at risk of becoming neoplastic increase with rate (α - β) per year and require "m" rare events to be transformed into neoplastic, or cancer, cells.

In general, "m" may equal 0, 1, 2, ... and have rare events A,B,C...M which occur at rate $R_A, R_B, ...R_M$ per cell division. No genetic changes have been identified as necessary for the process of promotion in either humans or animals, so it is important to consider the case where m = 0. This special case corresponds to a preneoplastic cell transforming into a neoplastic cell not by a rare event, but by a rare condition within the preneoplastic colony created simply by the growth of the colony and concomitant biochemical changes within the colony.

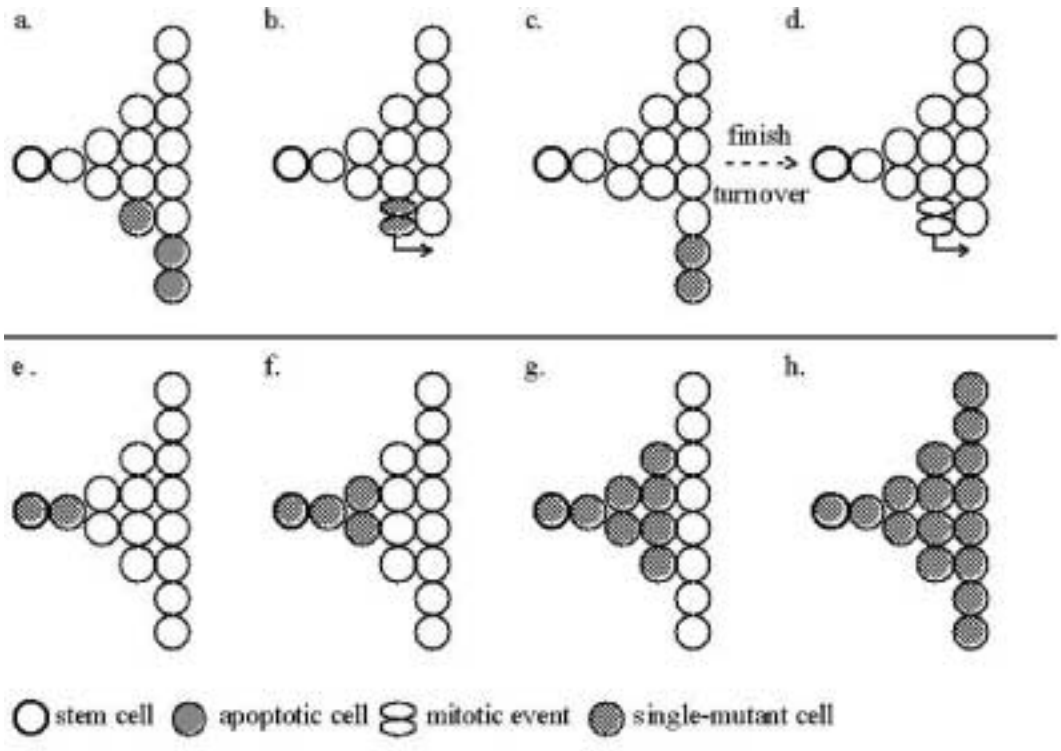

Figure 2: (Modified from [7]) *Upper panel*: Acquisition and subsequent loss of first mutation through normal cell turnover: a. A transition cell has acquired the first mutation during a previous division. Two terminal cells undergo apoptosis. b. The mutant transition cell fills in the terminal layer with two single-mutant cells. c. The rest of the turnover unit undergoes tissue renewal, with no more accumulation of single-mutant cells, since all other transitional cells are normal. d. Within the next round of turnover, the two mutants are replaced by two wild type cells from previous transition layer. The entire turnover unit then consists of normal wild type cells. *Lower panel*: Transformation of normal turnover unit into a mutant one through normal cell turnover: e. The stem cell acquires a first mutation. After the next full turnover, a copy of the mutant replaces the transition cell in the first transition layer. f. The mutant in the first transition layer divides and replaces the two normal cells in the next transition layer during the next turnover. The stem cell divides again to maintain a mutant copy in the first transition layer. g. With every subsequent turnover, step f is repeated, and the mutant cells propagate further. h. Eventually, the terminal layer is replaced entirely by mutant cells, and the entire unit becomes a preneoplastic colony.

2.3 Table of Definitions

The equations used to used mathematically model the two-stage carcinogenesis process are derived below, but first the definitions of all the terms are repeated here for convenience.

Population Parameters

h	The historical specific birth cohort (e.g. 1880-1889)
t	The age at death or incidence, depending on dataset used
G(h,t)	Fraction of population that has inherited genetic risk
E(h,t)	Fraction of the population that has experienced a required environmental risk factor
F(h,t)	G x E; Fraction of population that is at risk; $F_1 + F_2 + ... + F_n$
f(h,t)	Subpopulation that is at risk but dies of another disease caused by the risk factors defining F
S(h, t)	The survival rate for a particular birth cohort and age
R(h, t)	The error in recording actual cause of death for a particular birth cohort and age

Initiation Parameters

a	Age at which initiation occurs
a_{max}	The age through which initiation is allowed to occur
N_a	The number of cells at risk at age a
N_{max}	The maximum number of cells at risk throughout all of life
γ	The topological correction factor for the growth of a particular tissue
n	The number of events required for the initiation process to occur
$R_i, R_j, ...$	The rates per cell year of the events needed for initiation
$r_i, r_j, ...$	The rates per cell division of the events needed for initiation
τ	The number of stem cell turnover divisions per year

Transitional Probabilities

α	The division rate of cells at risk of promotion within a preneoplastic colony
β	The death rate of cells at risk of promotion within a preneoplastic colony
$(\alpha-\beta)/\alpha$	The probability of survival of a preneoplastic colony in which cells at risk of promotion have division rate alpha and death rate beta events/cell year.
$\alpha/(\alpha-\beta)$	The average initial number of surviving cells at risk of promotion in a preneoplastic colony.

Promotion Parameters

$(\alpha-\beta)$	The net growth rate of a preneoplastic colony (doublings per year)
m	The number of rare events required for the process of promotion to occur
$R_A, R_B, ...$	The rates per cell division of the rare events need for promotion
$2^{(\alpha-\beta)(t-a)}$	The number of cells resulting from each $\alpha/(\alpha-\beta)$ initial surviving cells in a preneoplastic colony at time (t-a).

Cascade Parameters

$V_{OBS}(t)$	The expected number of potentially lethal events (promotions) per individual at risk during year "t".
$P_{OBS}(t)$	The probability of death during "t" given S(h,t) = 0.

2.4 Derivation of the Equation

The following sections have been broadly abstracted from Herrero-Jimenez et al [6] and subsequently modified to reflect new developments to the mathematical model.

2.4.1 Primary population data sets

The primary population dataset is a set of age-specific cancer mortality data, which can be obtained from Professor William G. Thilly of the division of biological engineering at MIT now publicly available at http://epidemiology.mit.edu. These data were collected and organized from printed records of the US Public Health service and the US Bureau for the Census (1890-1997). The observed cancer mortality rate, OBS(t) is defined as:

$$OBS(h,t) = \frac{recorded\ deaths\ from\ lung\ cancer\ from\ birth\ cohort\ at\ age\ t}{recorded\ population\ size\ from\ birth\ cohort\ at\ age\ t} \quad (1)$$

Figure 1 is a graphical representation of OBS(t) for this particular data set.

The observed mortality rate is affected by the possibility that a cancer fatality is not correctly recorded as such. To account for this a function R(t) is defined as an estimate of the probability of accurately recording the cause of death:

$$R(h,t) = \frac{recorded\ deaths\ from\ specified\ causes\ at\ age\ t}{all\ recorded\ deaths\ from\ birth\ cohort\ at\ age\ t} \quad (2)$$

Assuming that the proportion of cancer deaths among deaths with unrecorded causes is about the same as the proportion of cancer deaths among all deaths with recorded diagnoses and that there is some fraction that survives the disease once they have been diagnosed with it. The function S(h, t) is defined as an estimate of the survival fraction. Taking this error into account, the "actual" cancer mortality rate OBS*(t) is estimated to be:

$$OBS*(t) = OBS(t)R(t)[1 - S(t)] \quad (3)$$

N.B. S(h,t) is not easily obtained. For cancer types with improvements in diagnosis and treatment, e.g., colon cancer or pediatric leukemia, these values have dramatically increased in the past fifty years. Even so, S(h,t) at old age approaches zero as physicians rarely treat the extremely aged with a malignancy. For other cancers such as esophageal, lung and pancreatic cancers, S(h,t) = 0 is, sadly, still a reasonable approximation at all ages. R(h,t) was significantly less than one especially for the aged in the first half of the twentieth century. However, its values have been approximated for all "h" and "t" so that data reaching back to 1900 can be used in the analyses addressed herein. ([7])

A second primary dataset used is the Swedish Cancer Database provided by Kari Hemminki[5]. Statistics Sweden created a family database, the "Second Generation

Register" in 1995. It initially included offspring born in Sweden in 1941 and their biological parents as families for a total of 6 millionindividuals. It was renamed the "Multigenerational Register" and expanded to include offspring born after 1931 by the fall of 2000. The Swedish Cancer Database correlates the Multigenerational Register with the Swedish Cancer Registry to provide the familial cancer datasets. The Swedish Family Cancer Database is the single largest population-based dataset ever used to study familial cancer.

2.4.2 Calculating the fraction of the population at risk

The model of Herrero-Jimenez et al. contained two key biological assumptions. First, they assumed that not all people are necessarily at risk for the disease, and second, they assume that the risk factors for cancer are also risk factors for other deadly diseases. Defined here are the parameters in the Herrero-Jimenez model, and an explanation of their analytical approach:

"F" denotes the fraction of the population at lifetime risk for a particular form of cancer. This would be the abstract fraction that would die of this form of cancer absent any other form of death. "f" denotes the fraction of that risk group that would have died of the observed form cancer were there a related disease having the same lifetime risk factors but again absent any unrelated form of death. $P_{OBS}(t)$ is the probability that an at-risk individual dies of cancer at age t, where it is assumed that death occurs shortly after promotion, but there is some survival rate S(t). So then: $e^{-\frac{1}{f}\int V_{OBS}(t)dt}$ = the probability that a person at risk for cancer has not yet died of any disease at time t, and so:

$$OBS*(t) = \frac{F \cdot P_{OBS}(t) \cdot e^{-\frac{1}{f}\int (1-S(t))V_{OBS}(t)dt}}{F \cdot e^{-\frac{1}{f}\int (1-S(t))V_{OBS}(t)dt} + (1-F)} \tag{4}$$

Simplifying, to obtain the final form of OBS*(t):

$$OBS*(t) = \frac{F \cdot P_{OBS}(t)}{F + (1-F) \cdot e^{\frac{1}{f}\int (1-S(t))V_{OBS}(t)dt}} \tag{5}$$

The inclusion of the exact probability that an at-risk individual dies at age t

($P_{OBS}(t)$) instead of the approximate use of the expected number of lethal events occurring in at-risk individuals at age t ($V_{OBS}(t)$) is a key change from the model put forth by Herrero-Jimenez et al. and is discussed in greater detail below.

2.4.3 Two stage Carcinogenesis Model

As mentioned above, the carcinogenesis process is modeled in two stages: initiation and promotion. Discussed here is the derivation of the two stages separately and how they fit together as a whole.

Initiation (in the case where n is assumed to be 2) The number of cells as a function of age is N_a, then there are $2R_iN_a$ cells acquiring the first initiation mutation at age a. Thus the total number of initiated cells in the organ becomes $2R_iN_a$. Furthermore, with age, mutant turnover units would accumulate linearly throughout the body, so that the total number of cells with the first mutation becomes $2R_iaN_a$. Any one of these cells may suffer the second initiation event, which occurs at a rate of R_j per cell year. When a cell has lost both alleles of the tumor suppressor gene, it is considered to be initiated. Thus, at age a, the expected number of newly initiated cells is $2R_iR_jaN_a$. For convenience define the term $C_{init} = 2R_iR_jN_{max}$, this term represents the constant of initiation.

Transitional Probability and Stochastic Redistribution Each initiated cell survives stochastic extinction with probability $(\alpha - \beta)/\alpha$, and the surviving colony doubles at a rate of $(\alpha - \beta)$. In the original model, this term is held constant, but later this thesis will examine what happens when its value is allowed to very over different spaces. Now the initiation term can be written as follows:

$$2R_iR_j\frac{\alpha - \beta}{\alpha}aN_a \tag{6}$$

Equation 6 then represents the expected number of surviving initiated colonies per organ per year. Since stochastic processes cannot increase or decrease the total number of initiated cells in a population, an average surviving initiated colony will

contain $\alpha/(\alpha-\beta)$ cells.

Promotion (in the case where m is assumed to be 1) Promotion, involves the growth of a surviving colony from age a to age t (at a rate of α-β) while the promotion event is taking place at a rate of R_A per cell division. An extra factor of $\alpha/[(\alpha-\beta)ln2]$ is introduced to account for the total number of cell divisions per population doubling in the preneoplastic colony. Furthermore, consider that a newly promoted cell could undergo further stochastic redistribution, thus leading to a deadly tumor, with probability $(\alpha_c-\beta_c)/\alpha_c$. Placing these terms together gives the average delay time between the initiation of preneoplastic colony and its subsequent promotion as follows:

$$1-e^{-(r_A\cdot(\frac{\alpha}{\alpha-\beta})^2\cdot\frac{\alpha_c}{\alpha_c-\beta_c}}\cdot 2^{((\alpha-\beta)\cdot(t-a)} \tag{7}$$

An entire three event(n=2 + m=1) process is illustrated in Figure 3.

In order to calculate the probability of promotion within each year of life, one may simply take the derivative of the bracketed term in equation 7. This allows us to model age-specific incidence data. The final form of $V_{OBS}(t)$ is the convolution of initiation at any age "a" and promotion at age "t", such that 0 < a < t. For n=2 and m=1, the explicit form is as follows:

$$V_{OBS}(t)=C_{init}\frac{\alpha-\beta}{\alpha}\int_0^t aN_a\frac{d(1-e^{-(r_A\cdot(\frac{\alpha}{\alpha-\beta})^2\cdot\frac{\alpha_c-\beta_c}{\alpha_c}}\cdot 2^{((\alpha-\beta)\cdot(t-a))})}{d(t-a)}da \tag{8}$$

Basically, $V_{OBS}(t)$ is the expected number of cells that are initiated at age a, and then promoted at age t. $V_{OBS}(t)$ can also be thought of as the expected number of lethal events at any age t since dearth would be expected within a few years of promotion without medical intervention.

Based on the expression for $V_{OBS}(t)$ the equation for $P_{OBS}(t)$ can be derived based on a simple application of the Poisson distribution. Here note that the probability of acquiring any number of events (Probability(x$\rangle$0)) is simply 1 - Probability(acquiring zero events). Then based on the Poisson distribution the final form of $P_{OBS}(t)$ is as

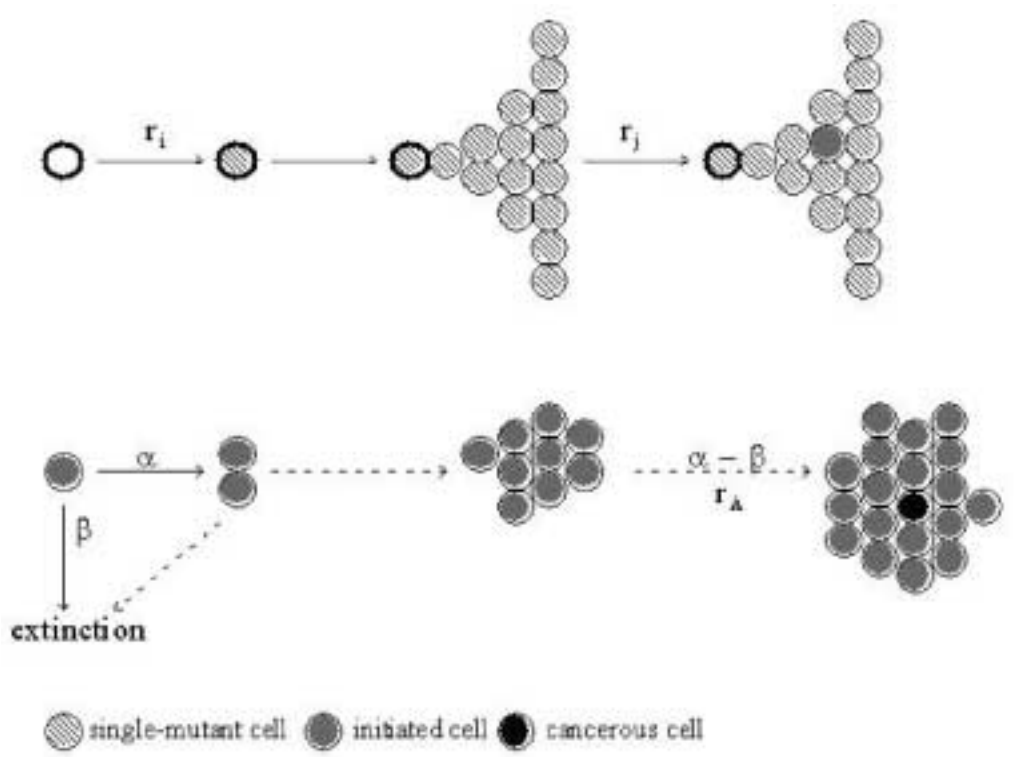

Figure 3: Three-event carcinogenesis model. A cell acquires the first mutation at a rate R_i. Each subsequent descendent cell can acquire the second mutation at a rate R_j. A cell that has undergone both mutations is said to be initiated, and now has an elevated division rate (α) and death rate (β). Initiated cells then undergo a process of stochastic growth, during which these cells can become extinct before colonizing the organ. Assuming that the colony has survived stochastic growth to become an adenoma, then the colony can undergo further growth at a doubling rate of $\alpha - \beta$. Each of these cells can acquire a third mutational event at rate R_A , and further undergo stochastic growth at rate of $\alpha_C - \beta_C$. This phase of growth is rapid, usually killing an untreated individual by three years. Formally, tumor growth and metastasis is termed "progression" a "third stage" of carcinogenesis.

follows:

$$P_{OBS}(t) = 1 - e^{-V_{OBS}(t)} \tag{9}$$

Number of Cells at Risk, N_a In order to define the rate at which an individual accumulates new preneoplastic cells, one needs to know the rates of each of the "n" required rare events, and the number of cells at risk at any given age a, N_a. Since cancer research has yet to define just which cells are at risk of turning into preneoplastic cells, it has been necessary to create a fairly general approach to encompass several possibilities. With regard to the definition of "cells at risk" my collaborators and I have recognized that theoretically any cell, stem transition, and even a newly created "terminal" cell could give rise to a preneoplastic colony upon experiencing the last of "n" required initiating events. While my collaborators are exploring the possibility that only stem cells can become preneoplastic cells, they require a cancer model that includes all possibilities. Another theoretical possibility is that there is an age limitation upon when in life initiation is possible. For instance my collaborators, noting that the calculated growth rate of preneoplastic colonies is about the same as the growth rate of children from infancy through puberty, are considering the idea that only juvenile cells are capable of initiation. A variant of this idea is the more restrictive hypothesis that only juvenile stem cells are at risk of initiation.

It would thus appear that the upper limit on the number of cells at risk is the set of all cells and the upper limit on the age at risk is undefined except by the limit of a finite lifespan. The lower limit might be the number of stem cells increasing from infancy through maturity, but lower limits might exist although without any apparently plausible physiological rationale at this point in research history.

Of course the number of cells in an organ from which particular forms of tumors arise may be expected to increase with age. [It is amazing but apparently true that biologists have not yet determined if organ growth happens by increases in stem cell number, increase in turnover unit size or both. Prof. E.E. Furth, University of Pennsylvania Medical School has examined juvenile and adult colons and communicated

to us her conclusion that the colon grows by increasing the number of crypts and not the size of crypts. This indicates growth by increasing stem cell number alone.]

In order to provide a means to analyze the outcomes for the wide range of possibilities presented, the growth of a particular tissue is modeled relative to the growth of the body as a whole. As a first approximation, a child's mass is a simple exponential with growth rate 0.16 for females and 0.158 for males extending to age 14.5 for females and 16.5 for males. ([6]) Such growth may be grossly approximated as the increase in mass of a sphere. Each organ may then be given a topological correction factor, γ, that relates its growth rate to the growth rate of a sphere. For example, the layers of cells of the esophagus from which esophageal cancers arise can be modeled as a cylinder, which has a surface area growth rate of approximately $\frac{2}{3}$ of the growth rate of the mass of a sphere thus giving the esophagus a γ of $\frac{2}{3}$.

In the CancerFit Program, the researcher may enter any number of cells as being at risk in a fully grown individual, N_{max}, the topological correction factor for the organ, γ, and the maximum age within which initiation is physiologically permitted, a_{max}. The idea represented by a_{max}, that an individual might escape risk of cancer simply by not having any cells initiated by age a_{max} creates a stochastic risk for each individual that is treated separately in the next section.

From the growth of children shown in Figure 4 below, the following equations model the growth of an organ relative to the growth of the whole body weight:

$$N_a = \begin{cases} N_{max} \div 2^{(0.16\cdot 15\cdot\gamma)+(1.2\cdot\gamma)\cdot(1.5-a)}, & 0 \le a \le 1.5; \\ N_{max} \div 2^{(0.16\cdot\gamma)\cdot(16.5-a)}, & 1.5 \le a \le 16.5; \\ N_{max}, & 16.5 \le a. \end{cases} \tag{10}$$

Based on Equation 10, using values for U.S. males, it is clear that the primary concern in calculating N_a is knowledge of the maximum number of cells at risk in an organ. The problem, as noted above, is that cancer researchers have not defined what cells are at risk among all of the cells in a tissue from which tumors arise. Obviously, there cannot be more cells at risk then there are cells in an organ and for cancer to develop there must be at least one cell at risk. But these are very wide bounds

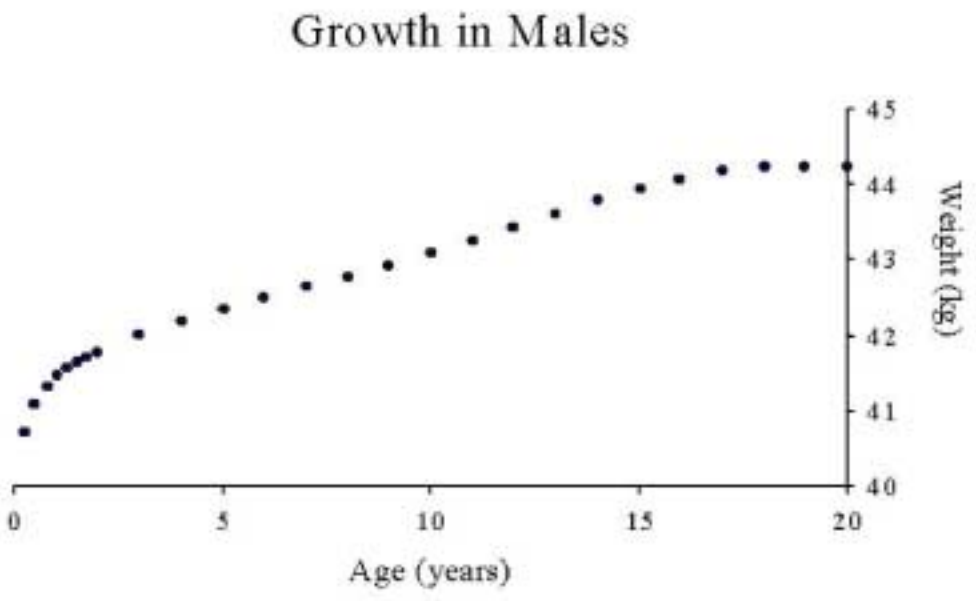

Figure 4: Weight of average sized male as a function of age (from [7]).

allowing for great variation. Two of the more popular notions are that either all cells are at risk or only stem cells are at risk.

Age-delimited Risk At least one element of physiologic cancer risk was recognized during the preparation of this thesis and has been addressed by appropriate modification of the CancerFit program. This form of risk relates to the observation that certain forms of cancer caused by inherited mutations in tumor suppressor genes appear early but not later in the lives of persons ascertained to be at genetic risk.

An example is the juvenile condition of retinoblastoma. Knudsen[8] analyzed and later Vogel[15] further discussed the phenomena of bi-lateral and uni-lateral retinoblastsoma in individuals inheriting heterozygosity of the retinoblastoma tumor suppressor gene, Rb. These tumors appeared only in young patients with retinoblastoma risk decreasing to very low values by age 8 and with no case recoded over age 17. (See Figure 5).

By analogy to this age-delimited occurrence of retinoblastoma CancerFit permits exploration of a more general concept of a time-limited process. One such time limitation could be the period that corresponds to the time in life in which humans

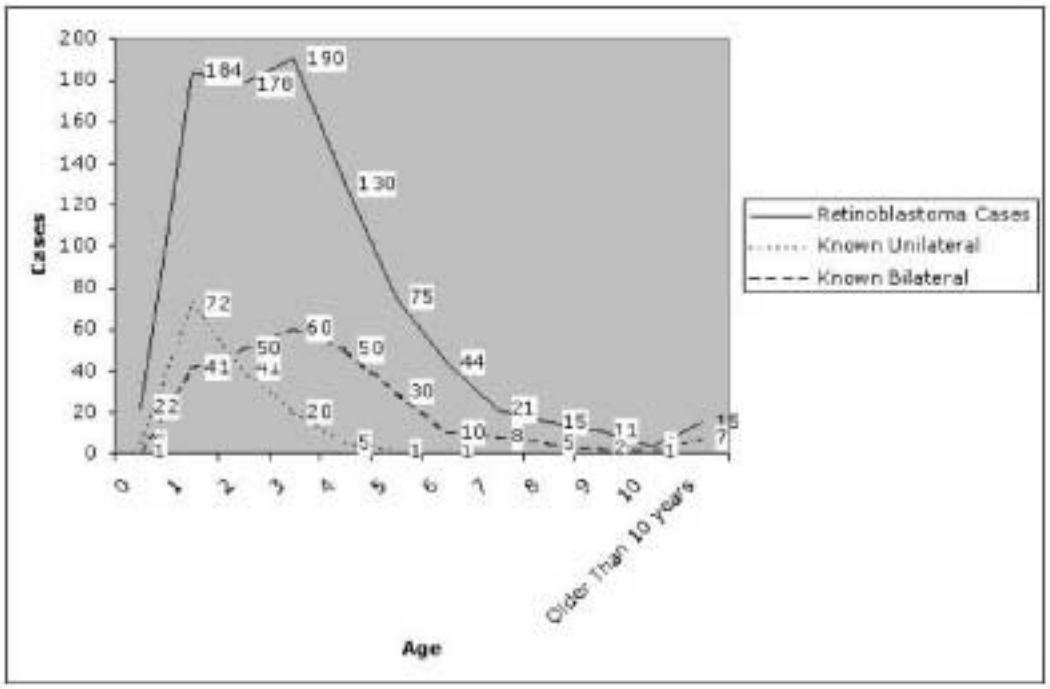

Figure 5: Distribution of ages of first diagnosis if retinoblastoma (adapted from Vogel [15]).

are still growing to their full adult size, when tissue stem cells have net positive growth rates. The notion of the initiation process only having a limited time window in which to occur has given the following hypothesis: that instead of a normal adult cell of zero net growth rate needing some set of events to give them a small but positive net growth rate, these events occurring in a juvenile cell might simply prevent its normal transition to an adult stem cell creating a colony growing at the rate of juvenile cells, which may be sufficient to characterize a preneoplastic colony. (W.Thilly, personal communication.) By chance there could be some people who are both genetically and environmentally at risk for all required events of carcinogenesis that simply do not develop a preneoplastic colony by a specified age related to tissue maturation physiology. These individuals would then no longer be at lifetime risk of that cancer despite being within the subpopulation at risk. This takes the previously continuous function that represents risk and truncates it at the end of some finite risk window, thus allowing a notion of discontinuous risk. Based on this define the fraction of

X	Cinit
0.1	0.0623
0.5	0.311
1.0	0.623

Table 1: Examples of how X, stochastic risk, can have various values based on the factors that define C_{init}. Other parameters are based on numbers for colon cancer: $\alpha = 9$ (as for adenomas of the colon), $\alpha - \beta = 0.143$, $N_{max} = 8.5 \cdot 10^{10}$ (assuming all cells are at risk), and an $a_{max} = 17$.

people who are still at risk within that window as at "stochastic risk". The fraction of the population within "F" at stochastic risk can be defined for any maximum age, a_{max}. At a_{max} the value of N_a is defined by its value from the hypothesis. For a hypothesis in which a_{max} is the age of maturity it would be 17 years for males and 15 years for females.

$$X(a_{max}) = \frac{C_{init}}{N_{max}} \cdot \frac{\alpha - \beta}{\alpha} \int_0^{a_{max}} a \cdot N_a da \tag{11}$$

Table 1 gives examples of how various feasible values of C_{init} can be used to obtain different levels of stochastic risk when initiation is limited to juvenile growth.

3 A Walkthrough the CancerFit Program

The following is a complete walkthrough of the CancerFit program. Treated here are Swedish colo-rectal incidence data that was created from the Swedish Family Cancer Database mentioned above. All Swedish persons recorded as parents and diagnososed for the first time with colo-rectal cancer within the years 1958-2000.

3.1 Creating a Data File

Before actually using the CancerFit program, the user needs to create a data file in the proper format. This file can be created using any type of simple text editor (for example NotePad for Windows or TextEdit for MACs, but not Word or Excel), or typing data into the program itself as discussed below. The format of the data file is strictly as follows: (Age) (Incidence/Mortality) (Population Living), see Table 2 below for an example. Mortality data for the U.S. (1900-1997) are available for the United States and Japan (1952-1995) are available in tables available at http://epidemiology.mit.edu. For cancers such as esophageal, pancreatic and lung, these mortality data reasonably approximate incidence data. But for cancers with historically improving survival rates such as colorectal cancer or Hodgkins lymphoma, an independent collection of historical age specific survival rates must also be constructed as in Herrero-Jimenez et al.[7].

3.2 The Data File View

This first step in CancerFit is importing the data file. To do this the user clicks "Open File" and selects the desired data file to load.

After the data file has been loaded, a graphical representation of the data will appear on the right portion of the screen marked "Data Plot". In this screen the user can also make modifications to the data file and then save those changes by clicking on the "Save File" button. (Note: when saving a file remove the top two lines that appear in the data window.) To then see the effects of the changed data file, re-open the file. To advance to the next step, click the button labeled "Next Step >>".

Age	Incidence	Living Population
2.5	1	5458452
7.5	8	7541468
12.5	46	9895568
17.5	108	12636812
22.5	165	15081315
27.5	281	16722876
32.5	460	17187982
37.5	883	16895335
42.5	1621	16567574
47.5	2844	15861931
52.5	4772	14633580
57.5	7120	12951101
62.5	9865	11189137
67.5	12518	9111675
72.5	13151	6730473
77.5	11368	4285316
82.5	6829	2214508
87.5	2759	833058
92.5	482	186732
97.5	30	20212

Table 2: A sample data file. The columns in order are Age Mortality/Incidence Population Living. Set of all colorectal cancers recorded among all Swedish parents 1958-1997. From Kari Hemminki, Karolinska Institutet, Stockholm.

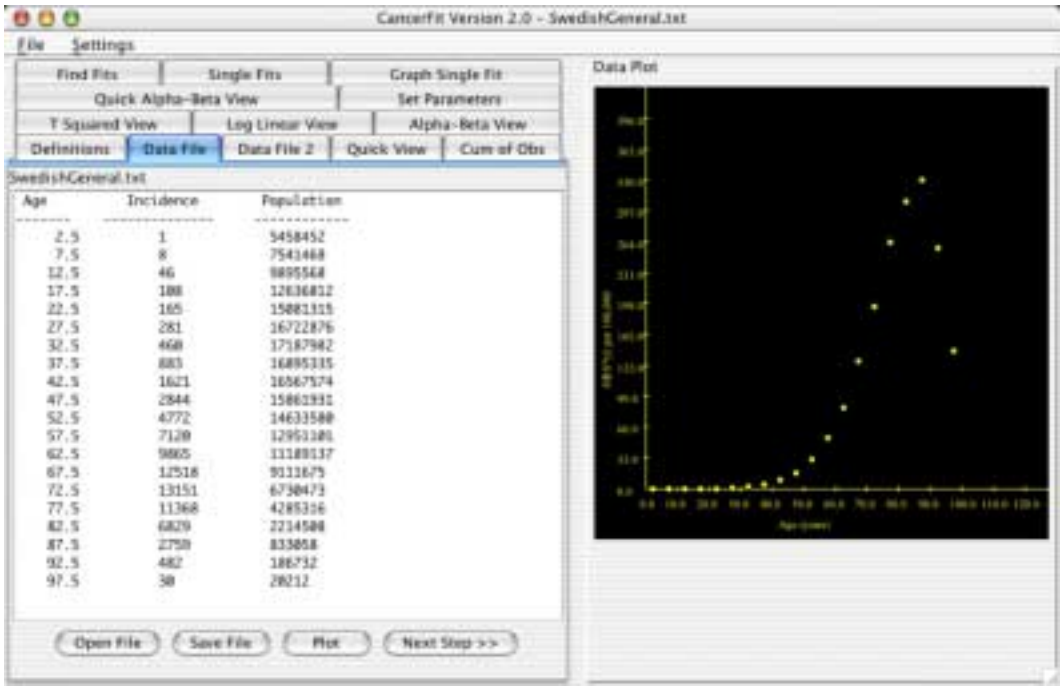

Figure 6: On the left, is the data from the selected file and on the right its corresponding graphical representation.

3.3 Data Breakdown

The next screen displays data by intervals of five years between the midpoint of the intervals of the age-specific data values. This permits calculation of estimates of the log_2 dOBS(t)/dt) which for the age interval between maturity and late middle age serves as an initial (over)estimate of the $\alpha - \beta$ term. The data on this page is arranged in three columns the age for the first data point, the age for the second data point, and the value of $\alpha - \beta$ calculated for the interval defined by those points. This first approximation is based on the data points given by the columns "Starting Age" and "Ending Age" and may be sometimes improved by calculating the best fit for $\alpha - \beta$ through young adulthood. However, early cancer syndromes such as familial adenomatous polyposis coli for colorectal cancer or van Hippel-Landau syndrome for kidney cancer can bias the estimates sought for late onset cancers. Further the group of individuals contracting a cancer early in adulthood should comprise a subset with values of $\alpha - \beta$ somewhat higher than the average population whose deaths occur much later in life. As is shown below, these estimates take on a wide range of values,

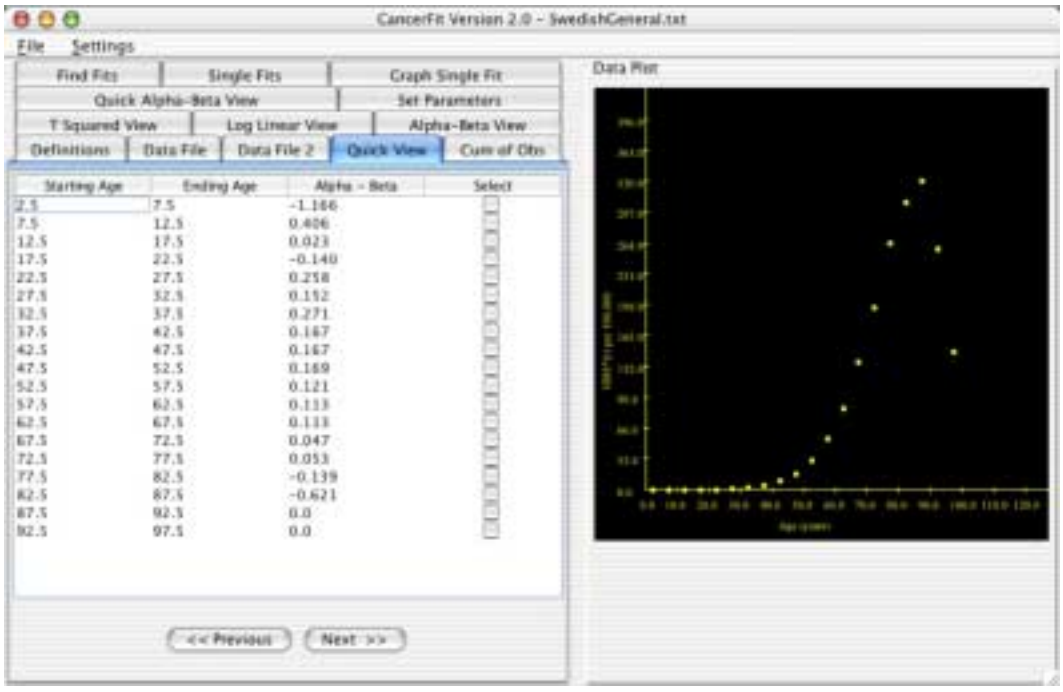

Figure 7: Here the "Quick View" screen is shown, which gives a quick estimate of $\alpha - \beta$ on a point by point basis.

but can still be used to get a general feel for the range of $\alpha - \beta$.

3.4 The Estimation of $\alpha - \beta$

The next step of CancerFit begins with the screen labeled "Alpha-Beta View." This screen shows the data plotted as log_2 (d OBS*(t) / dt) per 100,000 as a function of age as shown in Figure 8.

In Herrero-Jimenez et al. [6], a method for the estimation of $\alpha - \beta$ is given through calculating the log_2 derivative of OBS(t) scale was derived. The data for this function plotted in the left panel of Figure 8 allows for a better estimation of the $\alpha - \beta$ term than obtained through the data breakdown in the "Quick View" screen. To make use of this the user takes note of a region of near linearity (for example 30 - 50 in the above figure) and marks them for use as in the screen "Quick Alpha-Beta View" indicated in Figure 9 below. After the endpoints of the region have been marked, the button "Calculate Alpha-Beta" lights up. Click this button and the program will calculate an estimate of $\alpha - \beta$ and two standard deviations around that value. To

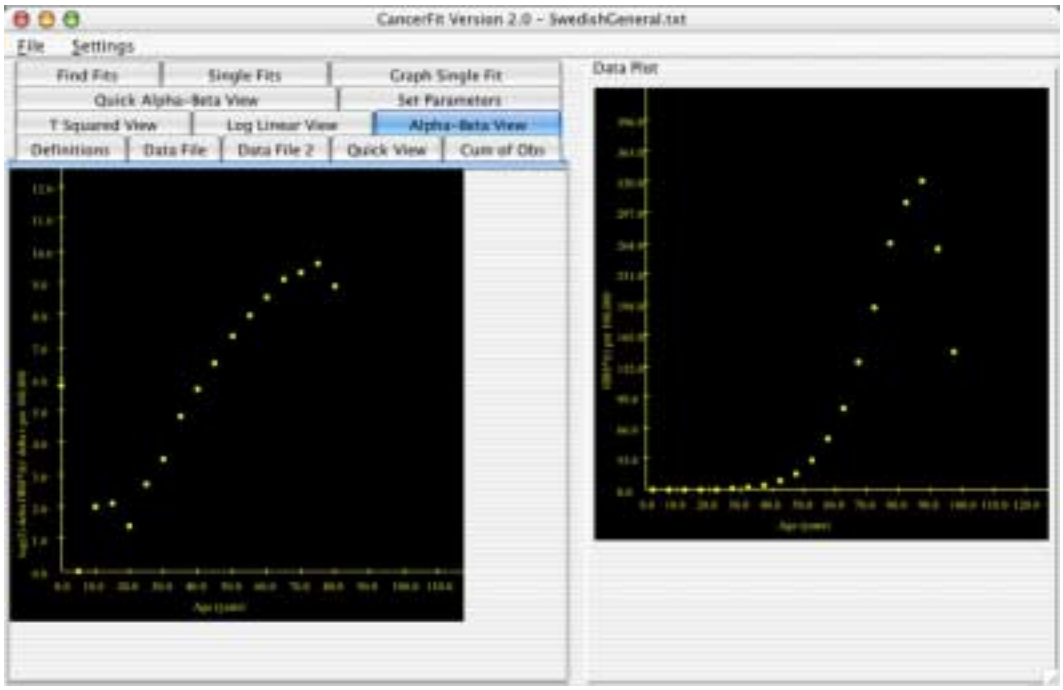

Figure 8: On the left panel of this screen a graphical display of the derivative of OBS(t) shown on a log_2 scale. Note the near-linear region from ages 30 through 50.

proceed to the next step, click the "Next >>" button. (It should be noted that since the data used to calculate this value is based on the fraction of people that die earlier in life than average, it is thought to use an over estimate and as such a viable upper limit on the "true" value of $\alpha - \beta$.)

3.5 Defining the Equation Parameters

The screen labeled "Set Parameters" allows for the user to define the growth function of the cells at risk, the number of initiation and promotion events, and the fraction of the population that survives despite a cancer diagnosis. Defining the growth function of the cells at risk involves four parameters: γ, N_{max}, a_{max}, and the sex of the population. Imbedded in the program is a set of body weights for males and females (from [6]) which defines the growth rate from birth to maturity, 14.5 years for females 16.5 years for males. If "Male" is not checked the default data set for body size is the female data.

These options allow researchers to explore specific hypotheses about when in life

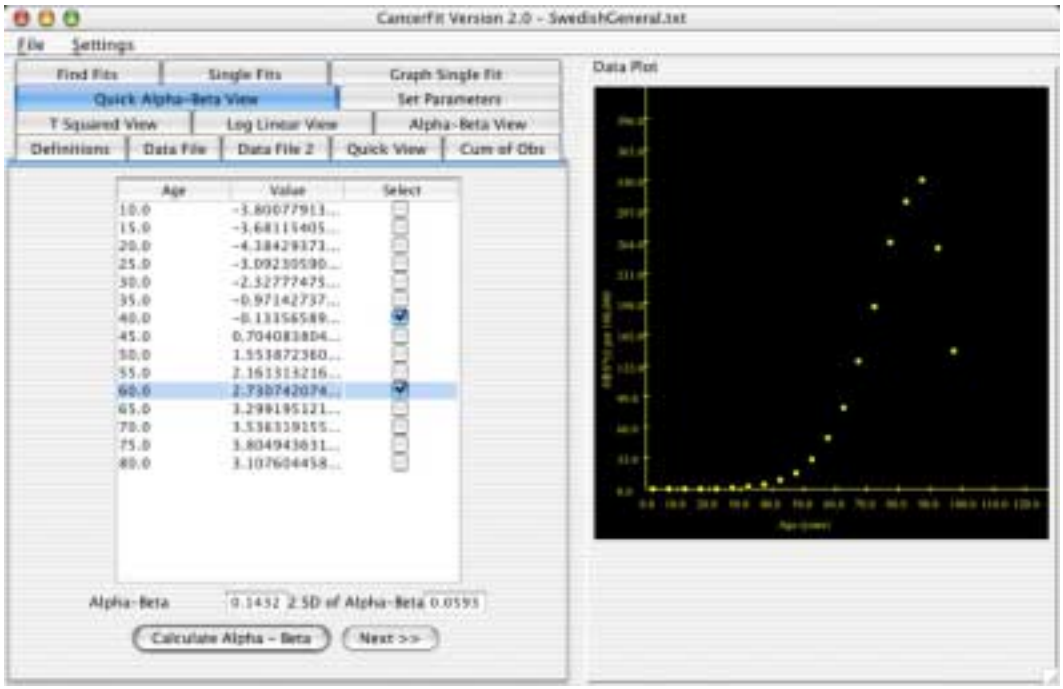

Figure 9: This figure contains the numerical breakdown of the graph shown in Figure 8. Note the approximation of $\alpha - \beta$ calculated for the selected range.

cancers can begin. If the hypothesis considered places no limit on the age at which initiation is possible one can enter "100" as a value of a_{max} and consider the effect of limitations on a_{max} simply be observing the effect of a_{max} on values of parameters calculated. In general it appears that values of a_{max} greater than 30 years yield equivalent values for all parameters from an age-specific cancer incidence dataset indicating that initiation of cancers occurs in childhood through young adulthood. More specific hypotheses can be explored by varying the initial parameters on this page. For instance, the hypothesis that cancer initiation is limited to juvenile cells may be represented by setting a_{max} at 15 for females and 17 for males.

One can explore hypotheses regarding which cells are at risk by specifying that value also. For instance there are about $8.5{\cdot}10^{10}$ cells in an adult male colon and about $4.2 \cdot 10^7$ stem cells (assuming one stem cell per colonic crypt). The hypothesis that all cells are at risk of initiation can be tested by using the maximum cell number and the hypothesis that only stem cells are at risk involves use of the stem cell number. As it appears that epithelial tissues such as the colonic epithelium grow by increasing

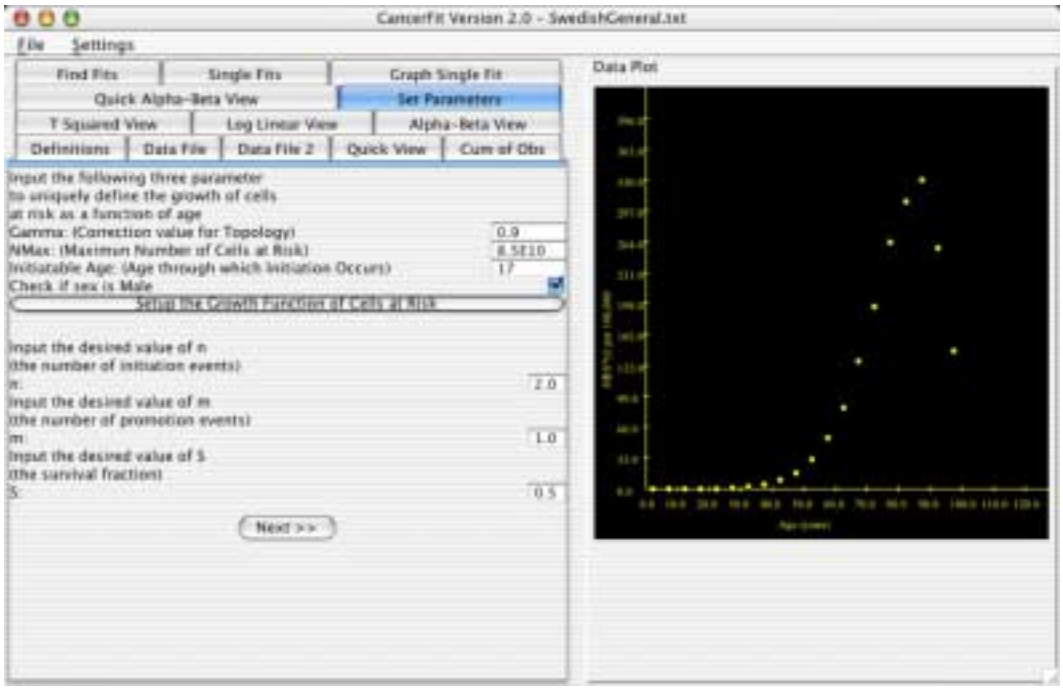

Figure 10: This screen contains the inputs to set the parameters for colorectal cancer. Here all cells of the colon to be at risk, but only through the period of juvenile growth.

the number of stem cells with an approximately constant stem cell/total cell ratio, the program uses the total body mass as a first approximation factor for growth of the epithelial cell population modified by the factor "γ" to reflect a lower growth rate in organs increasing at rates less than total body mass.

Each of these parameters can be defined for use in subsequent calculations by typing them in or checking the appropriate check box. One *must* then click "Setup the Growth Function of Cells at Risk" in order to import the desired parameter settings for subsequent computations.

The lower portion of the screen allows the user to input the desired number of initiation and promotion events as well as the survival fraction by typing them into the appropriate text fields. Once all the parameters are set as desired, click the "Next >>" button to continue to the final step.

3.6 Finding Fits

Once the user has set all of the parameters for the equations, the next challenging steps are to set the bounds for each variable that they would like to test. In the screen marked "Find Fits", the user is allowed to input the ranges for the variables F, f, C_{init}, $\alpha - \beta$, and r_A as well as the exact value for β. Clicking the check box next to any of the variables allows the user to fix that variable at the exact value expressed in its lower bound box.

The user is then asked to specify the number of iterations desired for examination of each variable. For example, if the user set the limits of F at 0.1 and 1 and then asked for 10 iterations, the program would calculate fits for F = 0.1, 0.2, 0.3,... 1.0. While if the user used the same limits and asked for 20 iterations the program would use F = 0.1, 0.15, 0.2, 0.25,... 1.0.

The final adjustment a user can make to the fitting process is to specify how he would like to determine the goodness of fit. The basic setup gives goodness of fit as the maximum distance between a fitted point and a data point, i.e. the greatest difference between the observed and calculated value of OBS(t) for all values of t addressed. The user can change this to use a residual sum of squares metric instead by clicking the RSS checkbox. This parameter sums the squares of differences between calculated values and observation over the entire range of t used.

To further fine tune the distance metric in the future a means by which the user can select a region of the data on which to weight the response by typing the desired range into lower and upper age weight boxes has been included. Any age range outside of the dataset will be able to be treated as if there were no weighting. For example, the range 1000 to 1000 would result in no weighting in which case the RSS values are the default output. This capability is under development by Prof. S, Morgenthaler at ETH, Lausanne and was included herein to prepare the appropriate computational platform.

After everything is set to test the user's hypothesis, the "Find Fits" button is clicked to start the fitting computations.

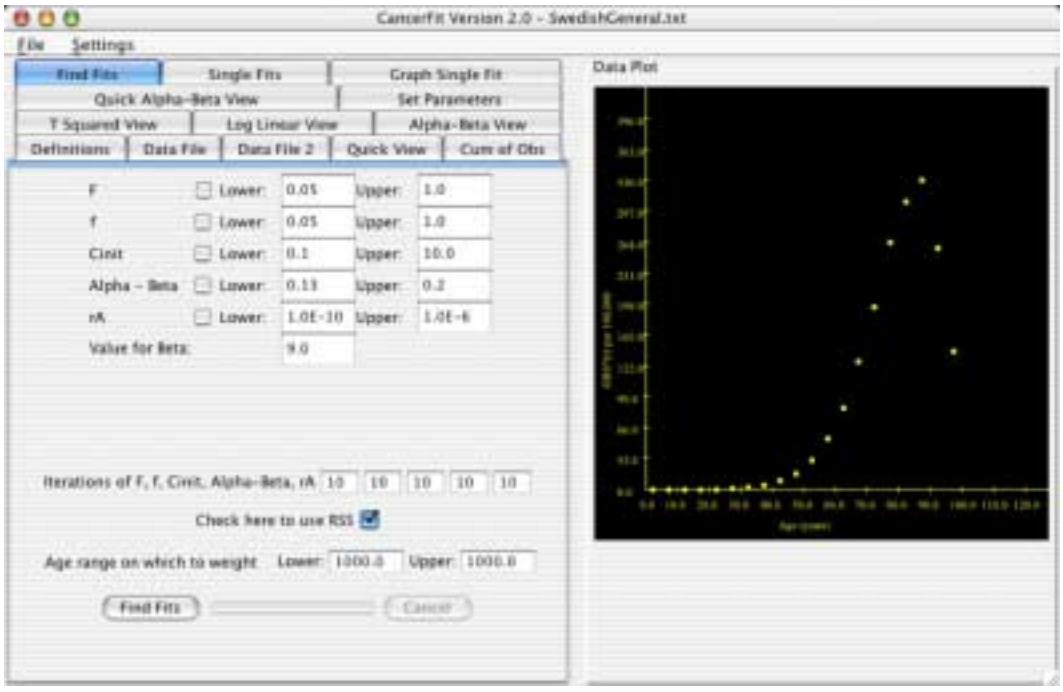

Figure 11: Here a basic setup for colo-rectal cancer is shown using residual sum of squares and relatively few iterations for each variable.

3.7 Results of Fitting

The window that pops up after the fitting process is done contains the hundred best fits that can be found with the specified parameters and limits. They are sorted by values of F and show what specific values they used for each variable as well as their distance from the dataset. Clicking on a result will plot it to the right in blue with its corresponding $P_{OBS}(t)$ shown in green. Clicking on the "Plot Pobs" button will bring up a small window that allows the user to see the full graph of $P_{OBS}(t)$ of the highlighted result. The "Show Data" button will bring up a chart of values for the highlighted result. The button labeled "<< Re-Run Find Fits" can be used to set the bounds of the "Find Fits" window at 95% and 105% of the values for the highlighted result. Additionally, the button labeled "Save Results" can be pressed to save all of the results as well as the parameters and bounds used to generate them to a file. One needs to set up a separate file beforehand into which these parameter settings and calculated results may be imported. What is saved are all the inputs from "Parameter Settings", "Find Fits" and the results from "Select Fit" in a simple

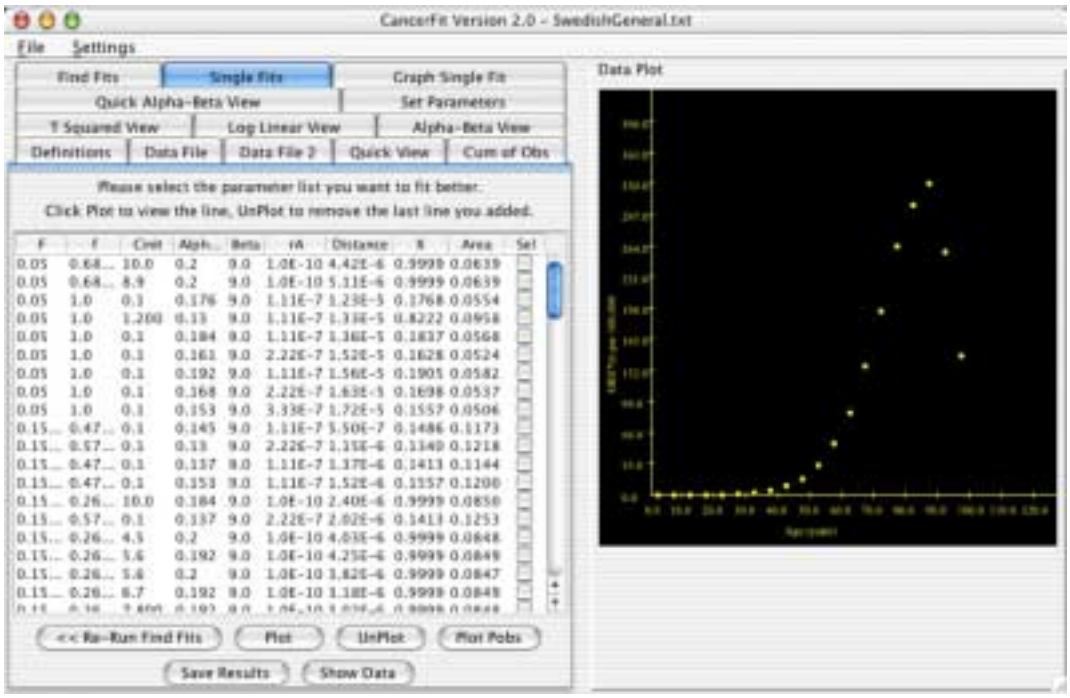

Figure 12: Displayed here are the results of our attempt to fit the colo-rectal data. It should be noted that as the user scrolls down he would see equally good fits even as F increases in value.

text display as shown in Figure 13.

3.8 Plotting a Specific Fit

The final window of CancerFit allows the user to plot a fit for specific values for each variable. The user simply inputs the values for each variable in their appropriate boxes and clicks the "Plot" button. The program will then plot the fit in red and the $P_{OBS}(t)$ graph in green and give the residual sum of squares distance from the dataset in the box labeled "Distance." Additionally, this window allows the user to view the $P_{OBS}(t)$ graph for those same values. This functionality allows the user to get a greater familiarity with the equations and how specific shifts in the variables affect the shape of the curve.

```
The Growth Function
Gamma:  0.9
NMax:  8.5E10
Initiateable Age:  17
Sex:  Male

n:  2
m:  1
S:  0.5

The Upper and Lower Limits
F:  1.0    0.05
f:  1.0    0.05
Cinit:  10.0    0.1
Alpha-Beta:  0.2    0.13
rA:  1.0E-6    1.0E-10
Beta:  9.0

The Looping Values
F:  10
f:  10
Cinit:  10
Alpha-Beta:  10
rA:  10

The Fits
F       f       Cinit   Alpha-Beta  Beta    rA          Distance     X
0.05    0.68    10.0    0.2         9.0     1.0E-10     4.42E-6     0.9999
0.05    0.68    8.9     0.2         9.0     1.0E-10     5.11E-6     0.9999
0.05    1.0     0.1     0.176       9.0     1.11E-7     1.23E-5     0.1768
0.05    1.0     1.20    0.134       9.0     1.11E-7     1.33E-5     0.8222
0.05    1.0     0.1     0.184       9.0     1.11E-7     1.36E-5     0.1837
0.05    1.0     0.1     0.16        9.0     2.22E-7     1.52E-5     0.1628
```

Figure 13: A truncated version of the file created by saving the results from Figure 12.

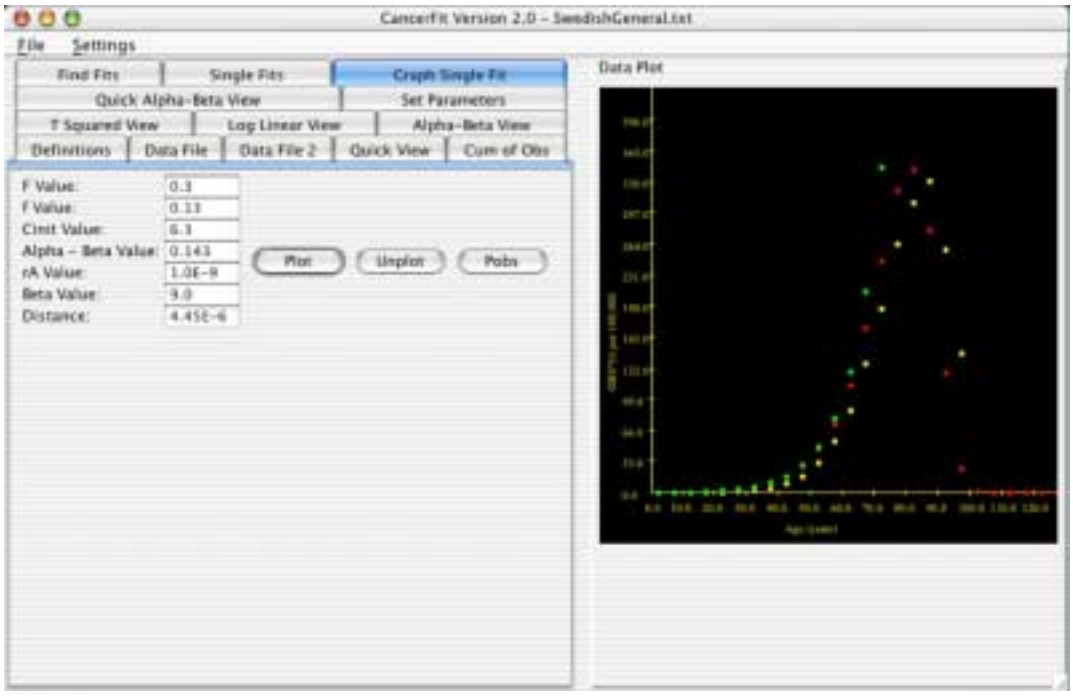

Figure 14: The resulting fit is shown in red on the left panel, while the values used to generate it and its residual sum of squares values are shown on the right.

3.9 Perspective through Alternative Views

The next three screens show the data displayed in other graphical forms. These screens are included to show the user that there could be alternative ways of looking at the data and thus modeling to fit the data. These views reminder researchers not to fall into the same trap of Armitage and Doll [1, 2]-that of limiting their view to too small a scope and getting stuck on one track.

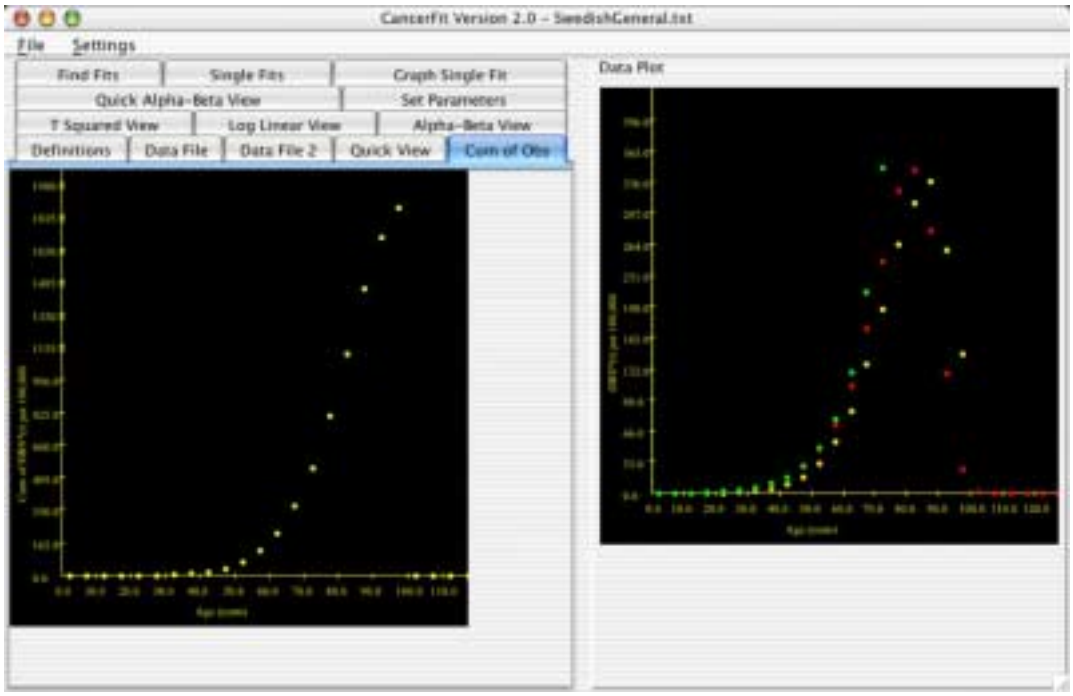

Figure 15: On the left the integral of OBS(t) is shown.

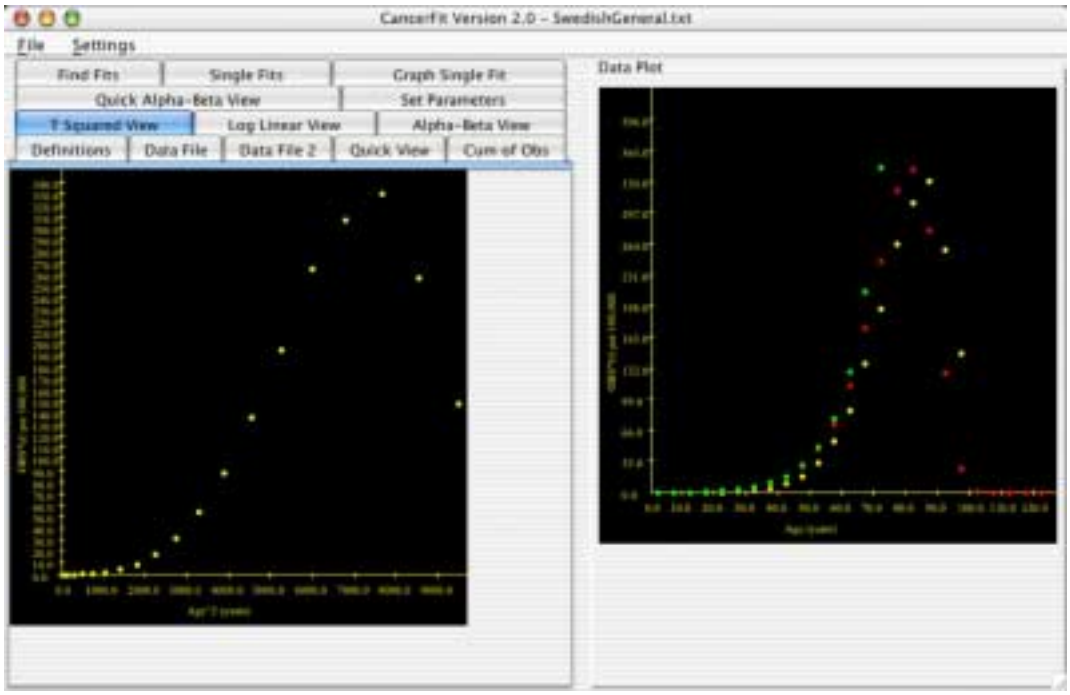

Figure 16: On the left OBS(t) plotted versus age^2 is shown. This would correspond to having three initiation events (n = 3).

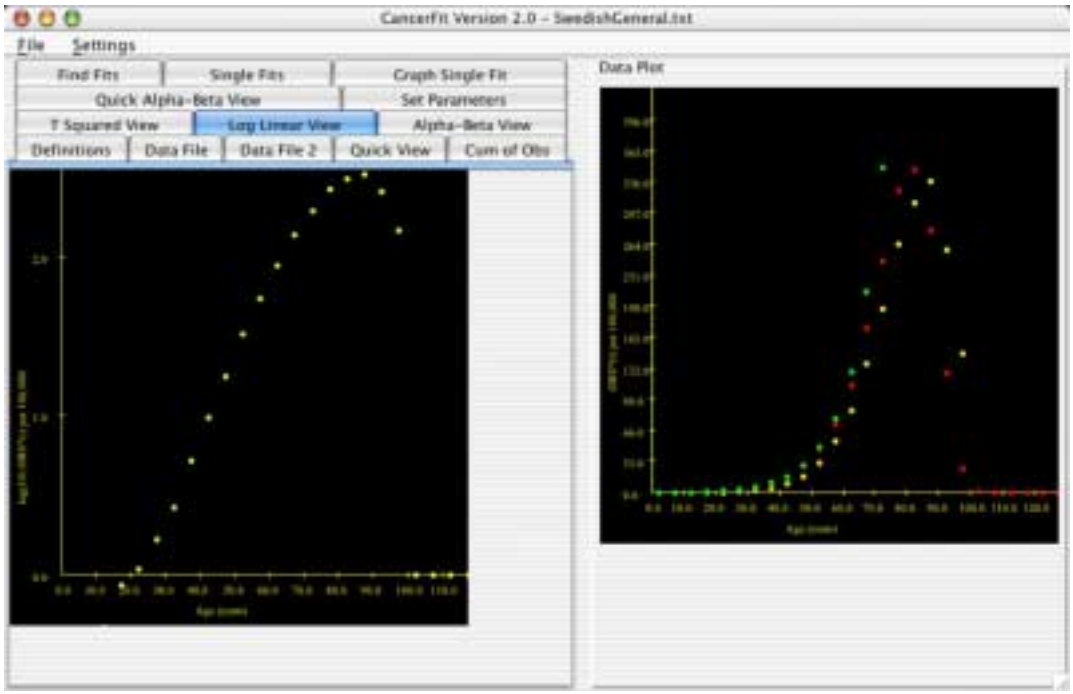

Figure 17: On the left the $log_{10}(OBS(t))$ is shown. This is the plot that Armitage and Doll [1] used as the basis for their attempts.

4 Results

The CancerFit program provides two major advantages for cancer researchers seeking to use age specific incidence or mortality data to derive estimates of parameters and to discover which parameters are robustly defined.

The first advantage is the provision of the two-stage model in a form that permits researchers to explore distinctly different hypotheses about carcinogenesis. Included among these are hypotheses about the kind and number of cells at risk of initiation, the number of events required for initiation, the age limits for risk of initiation, the number and kind of cells at risk of promotion and the number of events required for promotion. It automatically allows researchers to explore broad possibilities such as "all persons" are at essential risk of cancer or "risk for colon cancer does not confer risk of other forms of death".

The second major advantage comes from the use of the new Java based platform that permits rapid calculation of more than $6 \cdot 10^9$ trials per day ($\rangle 2.5 \cdot 10^8$ trials per hour) of combinations of the five key parameters of the two stage model each over a wide range of biologically reasonable values.

Using these advantages, it has been possible, for the first time, to discover if any of the five parameters (F, f, C_{init}, $\alpha - \beta$, r_A) are numerically defined given an age specific incidence/mortality data set and to discover the limits of computation for all five parameters.

Exhaustive computation until the set of solutions yielding residual sums of squares equal to or below the sum of variances of the data points allow four primary conclusions:

1. The value of the parameter $\alpha - \beta$ representing the growth rate of cells at risk of promotion in a preneoplastic colony is defined within a very narrow range. For instance in the case of colorectal cancer in Sweden (1958-2000 data) the estimated value lies between 0.141 and 0.145. Similar calculations from U.S. mortality data for populations born after 1870 yield similar definitions clustering around 0.143. Interestingly this is roughly equal to an approximation that the

epithelial surface of the colon would increase at about 0.9 x the growth rate of mass of a child which would be about $0.9 \cdot 0.16 = 0.144$. (W.Thilly, personal communication)

2. The term r_A, representing the promotion event rate per preneoplastic cell division assuming m=1, is defined to a lesser extent. Again using the Swedish colorectal age specific data to provide an example, the values of r_A were found to lie between $7 \cdot 10^5$ to $9 \cdot 10^5$ for the range of outcomes in which the computed residual sum of squares was equal to or less than the sum of variances for all data points containing 1000 or more colorectal cancer diagnoses, about $2 \cdot 10^{-8}$.

3. The values for F, f and C_{init} are not and cannot be uniquely defined through calculation from the data defining OBS(h,t). This is a finding directly contradicting two key publications from our group, Herrero-Jimenez et al.1998, 2000 [6, 7], in which the conclusion was reached that unique values could be and were defined by maximum likelihood approaches. This error was rooted in the comparatively slow computations afforded by an Excel based program that led the researchers to a "local minimum" without recognition of the presence of a large number of other possible maximum likelihood "fits" to their data. These errors will be reported and amended in the papers emanating from this thesis and parallel work of Prof. S. Morgenthaler.

4. However, the progress in directly measuring nuclear gene mutations in human organs, including the colonic epithelium now for the first time offers a means to reasonably approximate C_{init} for any particular hypothesis specifying the number and age-at-risk for normal cells in a tissue. When this parameter is specified using experimentally derived values, the estimates of F and f are uniquely defined for any OBS(h,t) This finding underlines the need for such experimental definition and the potentially valuable interaction of the products of this thesis and the expected experimental determination of mutation rates in humans.

These results can be seen below in Figure 18 for the hypothesis that only stem cells are at risk and only throughout juvenile growth, n=2 and m=1 (it should be

noted that computaions become significantly more intense and subsequently take significantly longer as the value of m increases). When computations range over feasible values for all parameters and to iterate over each variable enough times to sufficiently saturate its search space, it settles on the narrow range of 0.141 - 0.145 (with the majority of results settling on 0.143) for $\alpha - \beta$ and the reasonably narrow range of $7 - 9 \cdot 10^{-5}$ for $r_A 4$. At the same time, it can be seen that a large range of F, f, C_{init} give well defined fits that all have numerically equivalent low residual sums of squares values. This result is contrary to the conclusion of Herrero-Jimenez et al. [6] in which it was postulated that any reasonably defined dataset would monotonically converge to a single unique value for F. As a result of this much faster computational tool, it has been found that Herrero-Jimenez et al. were caught in the computational trap of a local minimum that they incorrectly interpreted as a global minimum. A manuscript describing these recent findings amending the error of Herrero-Jimenez et al. is in preparation.

The fourth result can be seen in the following example. Setting C_{init} to the experimentally confined range of $1 \cdot 10^4$ to $4 \cdot 10^3$, results are given such that F is confined to range from 0.25 to 0.33 and f from 0.12 to 0.17 as seen below in Figure 19. One can then imagine if the value of C_{init} could be fixed, there would be a corresponding unique value for F that would define a global minimum.

Furthermore, application of this program permits nonmathematical cancer researchers to see that there are multiple possible scenarios of the carcinogenic process that can/cannot be mathematically excluded. That is, the model finds fits that are equally good for several possible scenarios each based on feasible choices of parameters and limits. The four specific scenarios we studied involve the variance of the number of cells at risk in the colon from only stem cells ($4 \cdot 10^7$ cells for the colon) to all cells ($8.5 \cdot 10^{10}$ cells) as well as varying the time through which cells are at risk of initiation from just through juvenile growth ($\sim$17 years of age) to throughout all of life ($\sim$100 years of age for this dataset). CancerFit allows researchers the ability to mathematically model these and any other reasonable variations of the carcinogenic process.

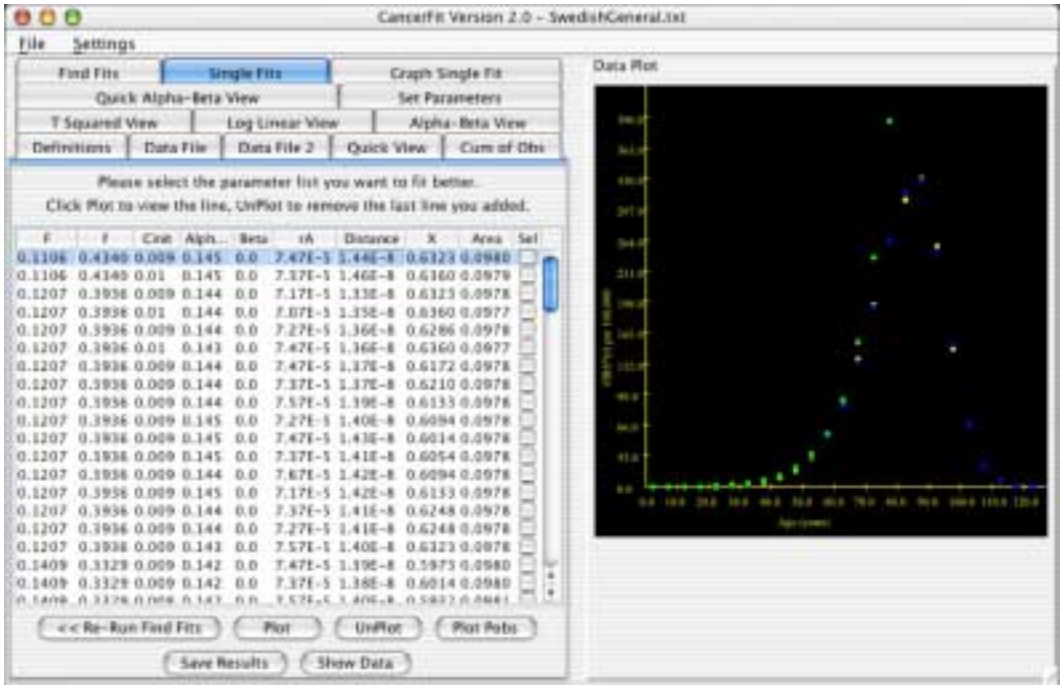

Figure 18: A partial set of results for the following parameter settings and ranges: γ = 0.9, $a_{max} = 17$, $N_{max} = 4 \cdot 10^7$, F = 0.5 - 1.0, f = 0.5 - 1.0, $C_{init} = 10^4 - 10^2$, $\alpha - \beta$ = 0.13 - 0.17, $r_A = 10^7 - 10^4$, $\beta = 0$. The iterations were set to equally saturate each search space. It should be noted that the full set of results gives equally valid solutions up through F = 1.0.

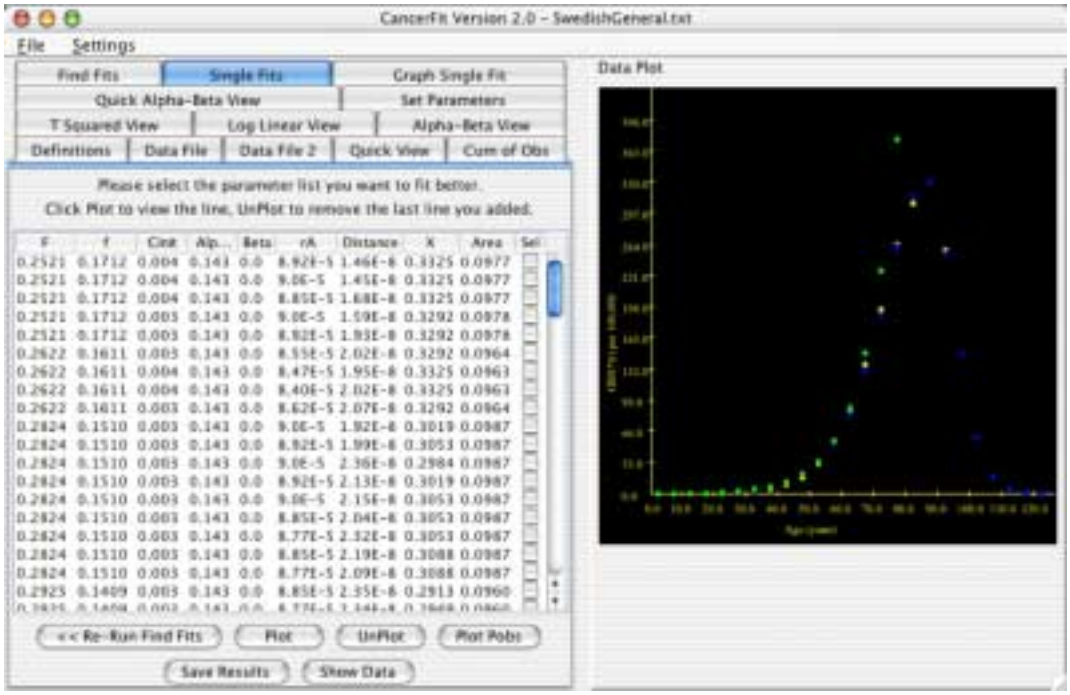

Figure 19: A partial set of results for the following parameter settings and ranges: γ = 0.9, $a_{max} = 17$, $N_{max} = 4 \cdot 10^7$, F = 0.5 - 1.0, f = 0.5 - 1.0, $C_{init} = 10^4 - 4 \cdot 10^3$, $\alpha - \beta$ = 0.143, $r_A = 6 \cdot 10^5 - 9 \cdot 10^5$, $\beta = 0$. The iterations were set to equally saturate each search space. The full set of results contains valid solutions from F = 0.25 through F = 0.33.

5 Future Work

CancerFit also has the capability to analyze otherkinds of age-specific disease data sets such as diabetes and vascular disease. It may also be modified to compare multiple datasets, primarily through using the ratio of two datasets. Work in this area is just beginning, but through this functionality cancer researchers will be able to explore risk within familial groups to estimate heritability as in Hemminki [5] and across historical cohorts to observe the affects of historical changes on the individual parameters.

A Source Code

```
/*
 * Fit.java
 *
 * Created on April 7, 2002, 4:53 PM
 */

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;

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();
    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();

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

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

    //Global Math variables
```

```
    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[] 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[][] 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 brth, deth, rA, rB, rC, rX;
    double c, ninit, minit, survivalFraction;
    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 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;

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 = "2.0" ;

boolean newEquation = true ;

int extension = 10 ;

double tempArea = 0.0 ;

/** Creates new Fit */
public Fit() {
    System.out.println("Loading....");

    //Do all the math initialization
    survivalFraction = 0.0 ;
    ninit = 2;
    minit = 1;
    ngrid = 200;
    igrid = 200;
    pobsline = -12 ;

    setParametersWindow = new SetParametersWindow();

    //Set the defualt values for all the parameters
    Frl = 0.05;
    Fru = 1.0;
    fl = 0.05;
    fu = 1.0;
    cl = 0.1;
    cu = 10;
    alminbel = .1;
    alminbeu = .3;
    betal = 9;
    betau = 9;
    rAl = Math.pow(10, -10);
    rAu = Math.pow(10, -6);
    alpha = 9.143 ;
```

```
        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;

        ////////////////////////////////////////////
        //**************Build Menu Bar**************//
        ////////////////////////////////////////////

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

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

        //New
        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);


        ////////////////////////////////////////
        //******Build Tabbed Pane and Graph*******//
        ////////////////////////////////////////

        this.getContentPane().setLayout(new BorderLayout());
        //Build the JTabbed Pane
        tabbedPane.addTab("Definitions", null, definitionsWindow) ;
        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("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("Find Fits", null, bestFitWindow);
        tabbedPane.addTab("Single Fits", null, selectFitWindow);
        tabbedPane.addTab("Graph Single Fit", null, graphSingleFitWindow);
        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);

    }

    ////////////////////////////////////////////////////////////////////////
    /************************** Listeners ****************************/
    ////////////////////////////////////////////////////////////////////////

    /* 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){
                graph.removeExcessLines(dataLine);
            } 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("SetScaleRatio.Plot") == 0){
```

```
                dataFileView2.plotRatio();
            } else if(e.getActionCommand().compareTo("DefinitionsWindow.Next") == 0){
                tabbedPane.setSelectedIndex(1);
            } else if(e.getActionCommand().compareTo("EstimateWindow.Previous") == 0) {
                tabbedPane.setSelectedIndex(1);
            } else if(e.getActionCommand().compareTo("EstimateWindow.Next") == 0){
                tabbedPane.setSelectedIndex(8);
            } else if(e.getActionCommand().compareTo("EstimateLogWindow.Estimate") == 0) {
                estimateLogWindow.calculateMU(estimateLogWindow.entryTable.getSelIndex());
            } else if(e.getActionCommand().compareTo("EstimateLogWindow.Next") == 0) {
                tabbedPane.setSelectedIndex(9);
            } 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) {
                ninit = Double.parseDouble(setParametersWindow.nNum.getText()) ;
                minit = Double.parseDouble(setParametersWindow.mNum.getText()) ;
                survivalFraction = Double.parseDouble(setParametersWindow.survivalNum.getText());
                tabbedPane.setSelectedIndex(10);
            }  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();
                tabbedPane.setSelectedIndex(10);
            } 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() ;
            }
        }
    }


    // 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());
        }
    }
}
////////////////////////////////////////////////////////////////////
/************************ Internal Classes ************************/
////////////////////////////////////////////////////////////////////


/** 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";
fileText += "\n" ;
fileText += "Neoplastic Colony:     Cancerous colony that leads to death";
fileText += "\n" ;
fileText += "Initiation:            Creation of preneoplastic colonies";
fileText += "\n" ;
fileText += "Promotion:             Creation of neoplastic colonies";
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 at lifetime risk (F = GE) ";
fileText += "\n" ;
fileText += "f:    Sub-fraction expected to die of related forms of mortality";
fileText += "\n" ;
fileText += "S:    The percentage of people who survive a specific cancer";
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 at age a";
fileText += "\n" ;
fileText += "Nmax:    The maximum number of cells at risk";
fileText += "\n" ;
```

```
fileText += "Cinit:   Constant of initiation";
fileText += "\n" ;
fileText += "Alpha-Beta:    Net growth rate of preneoplastic colonies";
fileText += "\n" ;
fileText += "Beta:     Rate of cell death";
fileText += "\n" ;
fileText += "rA:       Rate of required promotion events";
fileText += "\n" ;
fileText += "n:        Number of required initiation events";
fileText += "\n" ;
fileText += "m:        Number of required promotion events" ;
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);
        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];
    }
}

double thisAge = t[numlines-1] ;
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)");
graph.setYLabel("OBS*(t) per 100,000");
int yInterval = Math.round(Math.round(maxcases/10)) ;
graph.setEnd(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);
    }
}

/*****TSquaredView******/
//Initialize and plot the graphTSquared
graphTSquared.setStart(0,0);
graphTSquared.setXLabel("Age^2 (years)");
graphTSquared.setYLabel("OBS*(t) per 100,000");
graphTSquared.setEnd(Math.pow(maxt,2) + 200, maxcases + 20);
BufferedImage imgSquared = new BufferedImage(5,5, BufferedImage.TYPE_INT_RGB);
Graphics2D g2Squared = imgSquared.createGraphics();
g2Squared.setColor(Color.yellow);
g2Squared.fill(new java.awt.geom.Ellipse2D.Float(0,0, 5, 5));
dataLineSquared = graphTSquared.addLine(imgSquared);  //Set the Global dataLine
graphTSquared.setOffset(45,45);
graphTSquared.setInterval(1000, 10);
graphTSquared.calibrate();
graphTSquared.drawAxis();


for(int i =0; i<t.length; i++) {
    if(t[i] == 0) {
        break;
    } else {
        //normalize incidents per 100,000
        normalizedIncidents = cases[i]*(100000.00/pop[i]) ;
        graphTSquared.addPoint(dataLineSquared, Math.pow(t[i], 2), normalizedIncidents);
    }
}
```

```
            /*****LogLinearView******/
            //Initialize and plot the graphLogLinear
            graphLogLinear.setStart(0,0);
            graphLogLinear.setXLabel("Age (years)");
            graphLogLinear.setYLabel("log(10) OBS*(t) per 100,000");
            graphLogLinear.setEnd(maxt + 20, Math.log(maxcases + 20)/Math.log(10));
            BufferedImage imgLogLinear = new BufferedImage(5,5, BufferedImage.TYPE_INT_RGB);
            Graphics2D g2LogLinear = imgLogLinear.createGraphics();
            g2LogLinear.setColor(Color.yellow);
            g2LogLinear.fill(new java.awt.geom.Ellipse2D.Float(0,0, 5, 5));
            dataLineLogLinear = graphLogLinear.addLine(imgLogLinear);  //Set the Global dataLine
            graphLogLinear.setOffset(45,45);
            graphLogLinear.setInterval(10, 1);
            graphLogLinear.calibrate();
            graphLogLinear.drawAxis();

            for(int i =0; i<t.length; i++) {
                if(t[i] == 0) {
                    break;
                } else {
                    //normalize incidents per 100,000
                    normalizedIncidents = Math.log((cases[i]*(100000.00/pop[i])))/ Math.log(10) ;
                    graphLogLinear.addPoint(dataLineLogLinear, t[i], normalizedIncidents);
                }
            }


            /*****Mu View******/
            double loggedMin = 0.0;
            double loggedMax = 0.0;

            for(int i =0; i<t.length; i++) {
                if((t[i] == 0) || (t[i] > 85)) {
                    break;
                } else if (t[i] < 2.5){
                    //do nothing
                } else {
                    if (i == 0 )
                        loggedArray[i] = 0.0;
                    else
                        loggedArray[i] = Math.log(((cases[i]*(100000.00/pop[i])) - (cases[i-1]*
(100000.00/pop[i-1])))/5)/ Math.log(2) ;
                }
                loggedMin = Math.min(loggedArray[i], loggedMin);
                loggedMax = Math.max(loggedArray[i], loggedMax);
            }

            //Initialize and plot the graphLogLinear2
            graphLogLinear2.setStart(0, 0);
            graphLogLinear2.setXLabel("Age (years)");
            graphLogLinear2.setYLabel("log(2) delta OBS*(t)/ delta t per 100,000");
            graphLogLinear2.setEnd(maxt + 20, loggedMax+(1.5*Math.abs(loggedMin)));
            BufferedImage imgLogLinear2 = new BufferedImage(5,5, BufferedImage.TYPE_INT_RGB);
            Graphics2D g2LogLinear2 = imgLogLinear2.createGraphics();
            g2LogLinear2.setColor(Color.yellow);
            g2LogLinear2.fill(new java.awt.geom.Ellipse2D.Float(0,0, 5, 5));
            dataLineLogLinear2 = graphLogLinear2.addLine(imgLogLinear2);  //Set the Global
dataLine
            graphLogLinear2.setOffset(45,45);
            graphLogLinear2.setInterval(10, 1);
            graphLogLinear2.calibrate();
            graphLogLinear2.drawAxis();

            for(int i =0; i<t.length; i++) {
                if((t[i] == 0) || (t[i] > 85)) {
                    break;
                } else if (t[i] < 2.5){
                    //do nothing
```

```
            } else {
                graphLogLinear2.addPoint(dataLineLogLinear2, t[i]-2.5, loggedArray[i]+
Math.abs(loggedMin));
            }
        }


        double logLogMin = 0.0;
        double logLogMax = 0.0;

        for(int i =0; i<t.length; i++) {
            if((t[i] == 0) || (t[i] > 95)) {
                break;
            } else if (t[i] < 2.5){
                //do nothing
            } else {
                if (i == 0 )
                    logLogArray[i] = 0.0;
                else
                    logLogArray[i] = Math.log(cases[i]*(100000.00/pop[i]))/ Math.log(10) ;
            }
            logLogMin = Math.min(logLogArray[i], logLogMin);
            logLogMax = Math.max(logLogArray[i], logLogMax);
        }

        //Initialize and plot the graphLogLog
        graphLogLog.setStart(0, 0);
        graphLogLog.setXLabel("log(10) Age (years)");
        graphLogLog.setYLabel("log(10) delta OBS*(t) per 100,000");
        graphLogLog.setEnd(Math.log(maxt + 20)/Math.log(10), logLogMax+(1.5*Math.abs(
logLogMin)));
        BufferedImage imgLogLog = new BufferedImage(5,5, BufferedImage.TYPE_INT_RGB);
        Graphics2D g2LogLog = imgLogLog.createGraphics();
        g2LogLog.setColor(Color.yellow);
        g2LogLog.fill(new java.awt.geom.Ellipse2D.Float(0,0, 5, 5));
        dataLineLogLog = graphLogLog.addLine(imgLogLog);  //Set the Global dataLine
        graphLogLog.setOffset(45,45);
        graphLogLog.setInterval(.25, 1);
        graphLogLog.calibrate();
        graphLogLog.drawAxis();

        for(int i =0; i<t.length; i++) {
            if((t[i] == 0) || (t[i] > maxt)) {
                break;
            } else if (t[i] < 2.5){
                //do nothing
            } else {
                graphLogLog.addPoint(dataLineLogLog, Math.log(t[i]-2.5)/Math.log(10),
 logLogArray[i]+Math.abs(logLogMin));
            }
        }

        //Begin setup anf initialization for graphCumObs
        double maxCum = 0;
        for (int i =0; i<t.length; i++) {
            normalizedIncidents = cases[i]*(100000.00/pop[i]) ;
            if ((Double.toString(normalizedIncidents)).equals("NaN"))
                break ;
            maxCum = maxCum + normalizedIncidents ;
        }

        //Initialize and plot the graph
        graphCumObs.setStart(0, 0);
        graphCumObs.setXLabel("Age (years)");
        graphCumObs.setYLabel("Cum of OBS*(t) per 100,000");
        int yIntervalCum = Math.round(Math.round(maxCum/100)) ;
        graphCumObs.setEnd(maxt + 20, maxCum + yIntervalCum*10);
        BufferedImage imgCumObs = new BufferedImage(5,5, BufferedImage.TYPE_INT_RGB);
```

```
            Graphics2D g2CumObs = imgCumObs.createGraphics();
            g2CumObs.setColor(Color.yellow);
            g2CumObs.fill(new java.awt.geom.Ellipse2D.Float(0,0, 5, 5));
            cumObsLine = graphCumObs.addLine(img);  //Set the Global dataLine
            graphCumObs.setOffset(45,45);
            graphCumObs.setInterval(10,yInterval*5);
            graphCumObs.calibrate();
            graphCumObs.drawAxis();

            double currentTotal = 0.0 ;
            for(int i =0; i<t.length; i++) {
                if(t[i] == 0) {
                    break;
                } else {
                    //normalize incidents per 100,000
                    normalizedIncidents = cases[i]*(100000.00/pop[i]) ;
                    currentTotal = currentTotal + normalizedIncidents ;
                    graphCumObs.addPoint(cumObsLine, t[i], currentTotal);
                }
            }


        }

        public void next(){
            Vector vec = estimate(t, cases, pop);
            double slp[] = (double[])vec.get(0);
            double slope[] = (double[])vec.get(1);
            double deta[] = (double[])vec.get(2);
            int tempIndex = ((String)(vec.get(3) + "")).lastIndexOf("E") ;
            int tempIndex2 = ((String)(vec.get(3) + "")).lastIndexOf(".") ;
            if (tempIndex > 0)
                estimateWindow.maxSlope.setText(((String)(vec.get(3) + "")).substring(0, 4)
 + ((String)(vec.get(3) + "")).substring(tempIndex));
            else if (tempIndex2 >  0)
                estimateWindow.maxSlope.setText(((String)(vec.get(3) + "")).substring(0,
 tempIndex2));
            else
                estimateWindow.maxSlope.setText(((String)(vec.get(3) + "")).substring(0, 6));
            estimateWindow.maxDeta.setText((String)(vec.get(4) + "").substring(0,4));
            estimateWindow.maxSlope.setCaretPosition(0);
            estimateWindow.maxDeta.setCaretPosition(0);
            estimateWindow.entryTable.updateTable(slp, slope, deta);

            //Setup the estimateLogWindow
            estimateLogWindow.entryTable.updateTable(t, loggedArray);
        }
    }



/** 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 ;

        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");
    tempButton.setActionCommand("DataFileView2.Plot");
    tempButton.addActionListener(tabbedPaneListener);
    tempPanel.add(tempButton);

    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);

    }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());
        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] ;
        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(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("SetScaleRatio.Plot");
        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") ;
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();

double normalizedIncidents1, normalizedIncidents2 ;
double minRatio = 10000.0 ;
double maxRatio = 0.0 ;
for(int i =0; i<t2.length; i++) {
        if(t2[i] == 0)
            break;
        else {
            normalizedIncidents1 = cases[i]*(100000.00/pop[i]) ;
            normalizedIncidents2 = cases2[i]*(100000.00/pop2[i]) ;
            ratioGraph.addPoint(lineNum, t2[i], normalizedIncidents2/normalizedIncidents1);
            if ((t2[i] > 40) && (t2[i] < 90)){
                minRatio =Math.min(minRatio, normalizedIncidents2/normalizedIncidents1);
                if (minRatio == normalizedIncidents2/normalizedIncidents1)
                    maxRatio = minRatio ;
                maxRatio =Math.max(maxRatio, normalizedIncidents2/normalizedIncidents1);
            }
        }
    }
System.out.println("The minimum ratio is " + minRatio) ;
System.out.println("The maximum ratio is " + maxRatio) ;

//Calculate the area under the first curve
int i = 0 ;
while((i < cases.length) && (cases[i] != 0.0)){
    area1 = area1 + (cases[i]*(100000.00/pop[i])) * 5.0 ;
    i++ ;
}
double slope = ((cases[i-1]*(100000.00/pop[i-1])) -(cases[i-2]*(100000.00/pop[i-2])))/5;
double height = cases[i-1]*(100000.00/pop[i-1]) ;
double base =  Math.abs(height / slope) ;
area1 = area1 + (0.5* base *height) ;
area1 = area1 * Math.pow(10, -5);

//Calculate the area under the second curve
i = 0 ;
while(((i < cases2.length) && (cases2[i] != 0.0)) || (i < 5)){
    area2 = area2 + (cases2[i]*(100000.00/pop2[i])) * 5.0 ;
    i++ ;
}
slope = ((cases2[i-1]*(100000.00/pop2[i-1])) -(cases2[i-2]*(100000.00/pop2[i-2])))/5;
height = cases2[i-1]*(100000.00/pop2[i-1]) ;
```

```
        base =  Math.abs(height / slope) ;
        area2 = area2 + (0.5* base *height) ;
        area2 = area2 * Math.pow(10, -5);

        area2 = 0.144 ;
        ratioOfAreas = area2/area1 ;
        System.out.println("Area of red curve " + area2) ;
        System.out.println("Area of yellow curve " + area1) ;
        System.out.println("ratioOfAreas is " + ratioOfAreas) ;

        double guess = 0.0 ;
        //an iterative solution
        double loopingF = 0.0 ;
        int looper = 0 ;
        while (looper < 99){
            looper  = looper + 1 ;
            loopingF = (double) looper / 100 ;
            guess = Math.log(1-minRatio*loopingF) / Math.log(1-loopingF) ;
            if (((ratioOfAreas*.99) <= guess) && ((ratioOfAreas*1.01) >= guess))
                break;
            System.out.println("For loopingF = " + loopingF + " guess = " + guess) ;
        }
        System.out.println("F = " + loopingF);


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

        }
}

    /** This is the second step in estimating alpha-beta.  Takes in the data
     * from the "Quick Mu View" window, and allows the user to select a set of points
     * to use to calculate alpha - beta. */

    class EstimateLogWindow extends JPanel {
        JPanel directions = new JPanel();
        JTable entryView = null;
        EntryTable entryTable = new EntryTable();
        JButton estimate = new JButton("Calculate Alpha - Beta");
        JButton next = new JButton("Next >>");
        DoubleNumberField aminusb = new DoubleNumberField(0, 5);
        DoubleNumberField sdMU = new DoubleNumberField(0, 5);
//two standard deviations of the MU estimate

        int checkcount = 0;

        EstimateLogWindow() {

            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("Alpha-Beta"), aminusb));
            temp.add(createEastWestPanel(new JLabel("2 SD of Alpha-Beta"), sdMU));
            this.add(temp);

            temp = new JPanel();
            estimate.addActionListener(tabbedPaneListener);
            estimate.setActionCommand("EstimateLogWindow.Estimate");
            temp.add(estimate);
            next.addActionListener(tabbedPaneListener);
            next.setActionCommand("EstimateLogWindow.Next");
```

```
            temp.add(next);
            this.add(temp);
            estimate.setEnabled(false);
        }

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

        /** Estimates the value of MU based on the chosen values
         * (denoted by pnt), and on that same data also calculates
         * two standard deviations from the mean**/
        void calculateMU(Point pnt) {
            //Calculate the mean (estimated MU)
            double estimatedMU = (loggedArray[(int) pnt.getY() + 2] - loggedArray[(int)
pnt.getX()+2]) / ((pnt.getY()-pnt.getX())*5) ;
            aminusb.setText(truncString(String.valueOf(estimatedMU), 4));

            //Calculate the Standard Deviation
            double sum = 0.0 ;
            int n = 0;
            for (int i = (int) pnt.getX() + 2; i < (int) pnt.getY() + 2; i++){
                sum = sum + Math.pow((((loggedArray[i+1] - loggedArray[i])/5)- estimatedMU), 2);
                n++ ;
            }
            double twiceSD = 2.0 * Math.sqrt(sum/(n-1)) ;
            sdMU.setText(truncString(String.valueOf(twiceSD), 4));

            //set the bound of MU to estimated+-twiceSD
            alminbel = estimatedMU - twiceSD ;
            alminbeu = estimatedMU + twiceSD ;
            bestFitWindow.setMU(alminbel, alminbeu) ;

        }

        //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");
                columnNames.add("Select");
            }

            public void updateTable(double[] age, double[] values) {
                data = new Object[values.length][3];
                int i = 0 ;
                while((i+2 < values.length) && (age[i+2] != 0) && (values[i+2] != 0)){
                    data[i][0] = String.valueOf(t[i+2]-2.5);
                    data[i][1] = String.valueOf(values[i+2]);
                    data[i][2] = new Boolean(false);
                    i++;
                }
                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;
        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 == 2) {
```

```
                return true;
            } else
                return false;
        }
    }
}


/** 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)*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;
    }
  }
}



/** 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);
            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));
            this.add(temp);

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

            next.setActionCommand("EstimateWindow.Next");
            previous.setActionCommand("EstimateWindow.Previous");
            temp.add(previous);
            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("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][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);
    JCheckBox maleCheckBox = 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(survivalFraction , 3);


    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: (Maximun 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(9, 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();
            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;

        //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("rA                  "));
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, rA");
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);
}

/** 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);
            tabbedPane.setSelectedIndex(11);
            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,
    //should check to see if the user wants to overwrite the file
    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();

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];
}

/*legacy from when we were experimenting with age dependent
```

```
        parameters.  Left in in case we would like to experiemtn with
        them again latter*/
        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("rA 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 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);
            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);
    //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*****************//

        //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);
            } else if(action.compareTo("SelectFitWindow.unPlot") == 0) {
                graph.removeLastLine();
            } 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("rA");
                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 ;

                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];
    }

 /*
  * 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] = 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);
        }
    }

}

/*********************************************************************/
/////////////////////////////////////////////////////////////////////
/************************** Functions ****************************/
/////////////////////////////////////////////////////////////////////
/*********************************************************************/

/*************************    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;
        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++){
        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;
}

/*main math procedure: "twostage" signifies the two stage process(initiation and promotion)
   this is the porcedure that ends up generating the values for Pobs and Vobs
*/
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;
}

/*  The is teh term that drives the initiation function of the equation.
    Has been modified to deal with the notion of a variable N[a] and
    a maxiumum age of initiation.
*/
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 ;
}

/* The Morgenthaler exact solution that deals with survival and the promotion term
    if statement deals with change in value of "m".  Gets very, very slow as m increases
*/
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
*/
```

```
    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
            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]);
        }

        //legacy code from switch from approximation by old Pobs to new
        //exact Vobs
        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++){
            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;
    }

    /*  The procedure that does the looping procedure to find the optimal fits in
        the defined search space
    */
    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;

        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);
                            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;


                                    for (int ii = nobs - 1; ii >= 0; ii--) {
                                        if(breakOut == true) {
                                            bestFitWindow.setProgress(100);
                                            return;
                                        }

                                        if (newEquation){
                                            fit = Pobs[ii] * Frac/ (Frac+ ((double)1 - Frac) *
 Math.exp((1-survivalFraction)*intPobs[ii] / f));
                                        }
                                        fit = Pobs[ii] * Frac / (Frac + ((double)1 - Frac) *
 Math.exp((1-survivalFraction)*intPobs[ii] / f));
                                        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 there is a age based weighting it should be
normalized
                                            //this methodology is still under development in
collaboration with
                                            //Prof. S. Morgenthaler at ETH
                                            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[ii] / (Math.pow(pop[ii], (double)2))));
                                            }
                                            else if (ii < ageIndexl)
                                                dist = dist + (Math.pow(fit-obs[ii], (double)2)
/ (cases[ageIndexl] / (Math.pow(pop[ageIndexl], (double)2))));
                                            else if (ageIndexu < nobs)
                                                dist = dist + (Math.pow(fit-obs[ii], (double)2)
/ (cases[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.
*/

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 sored 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 peserve 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
double[] temp = new double[9] ;
for (int i = nfits ; i > 0 ; i--)
    for (int j = 0; j < i -1; j++)
        if (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.  (LEGACY from old experimentation)
 */
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){
        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 ;
        }
        /*
        for (int j = 0; j<20; j++)
            System.out.println(N[j]) ;*/
    }


    /* 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 nubmer 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)*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)*intPobs[ii] / f)) * population;
            tempArea = tempArea + fit ;
            graph.addPoint(lineNum, t[ii], fit);
        }

        //Also graph Pobs
        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();
    }

    //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)*intPobs[ii] / f)) * population;
            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
        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++){
            fit = Pobs[i] * Frac/ (Frac+ (1.0 - Frac) * Math.exp((1-survivalFraction)*
intPobs[i] / f));
            area = area + fit;
        }

        return area ;
        }

    /** @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("rA:  " + 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("rA:  " + bestFitWindow.rAnum.getText()+"\n") ;

                fileWriter.write("\nThe Fits"+"\n") ;
                fileWriter.write("F           f          Cinit         Alpha-Beta     Beta
     rA        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. Armitage and R. Doll. The age distribution of cancer and a multi-stage theory of carcinogenesis. *British Journal of Cancer*, 1954.

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

[3] A. Dewanji, E.G. Luebeck, and S.H. Moolgavkar. A stochastic two-stage model for cancer risk assessment. ii. the number and size of premalignant clones. *Risk Analysis*, 1989.

[4] A. Dewanji, S.H. Moolgavkar, and E.G. Luebeck. Two-mutation model for carcinogenesis: Joint analysis of premalignant and malignant lesions. *Mathematical Biosciences*, 1991.

[5] K. Hemminki, X. Li, K. Plna, C. Granstrom, and P. Vaittinen. The nationwide swedish family-cancer database: updated structure and familial rates. *Acta Oncol*, 2001.

[6] P. Herrero-Jimenez, G. Thilly, P.J. Southam, A. Tomita-Mitchell, S. Morgenthaler, E.E. Furth, and W.G. Thilly. Mutation, cell kinetics, and subpopulations at risk for colon cancer in the united states. *Mutation Research*, May 1998.

[7] Pablo Herrero-Jimenez, Stephan Morgenthaler, and William G. Thilly. Analysis of lung cancer mortality rates and cigarette use in the united states. *Mutation Research*, January 2000.

[8] A.G. Jr. Knudson. Mutation and cancer: Statistical study of retinoblastoma. *Proceedings of the National Academy of Science of the United States of America*, April 1971.

[9] S.H. Moolgavkar, A. Dewanji, and D.J. Venzon. A stochastic two-stage model for cancer risk assessment. i. the hazard function and the probability of tumor. *Risk Analysis*, 1988.

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

[11] S.H. Moolgavkar and E.G. Luebeck. Two-event model for carcinogenesis: Biological, mathematical, and statistical considerations. *Risk Analysis*, 1990.

[12] S.H. Moolgavkar and E.G. Luebeck. Multistage carcinogenesis: Population-based model for colon cancer. *Journal of the National Cancer Institute*, 1992.

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

[14] C.O. Nordling. A new theory on the cancer-inducing mechanism. *British Journal of Cancer*, 1953.

[15] F. Vogel. Genetics of retinoblastoma. *Human Genetics*, 1979.