**Death Clock**

**Ari Sosnovsky - 212076923**

**Kevin Gomes - 212132718**

**Peter Le - 210845428**

**Vicki Tran - 212202719**

**MATH 3090**

**Assignment 3**

**Dec. 5th, 2014**

**Introduction**

At one point or another, the thought of death has altered our reality. Although there should be no motivation to live life to the fullest, the idea of losing existence is quite thought provoking. Posing the question, “if you knew when you would die, would you live differently?” Would there be a little more motivation towards getting out of bed, taking a vacation or essentially burning a hole into your apprehensiveness (uneasy state of mind)? You would assume that with poor living habits, your longevity is shortened, and with ideal living behavior, a longer life is sustained. Despite being impossible to wholeheartedly predict, using personal data and vital statistics, one can actually estimate the remaining time of their mortality. Essentially, no application or process can really accurately determine your life expectancy, but a “death clock” can statistically try. Using personal information such as age, gender, and certain routines, the death clock gives you a reminder that death is imminent.

People seek a sense of security, which may be the next most significant goal after obtaining food, clothing, and shelter. Person(s) with economic security can satisfy their needs in the present and the future. These needs include food, shelter, medical care, etc. The possibility of losing economic security is called economic risk (which is typically referred to as risk). Risk derives from the variation from the expected outcome. With relation to the death clock, the risk of dying (or losing life security) derives from the variation from the expected outcome. To be more specific, the final prediction for a person’s age of death is a variation from the expected outcome for all causes (variables) that apply to this person.

**The Mathematics**

The death clock establishes an approximate death through probability, statistics and actuarial mathematics. It is not a guess, but rather a calculated approximate. To begin the mathematics behind the death clock, the variables are named:

* tlx: the ***number*** of ***survivals*** from age *x* to *x+t*
* tdx: the ***number*** of ***deaths*** from age *x* to *x+t*
* tqx: the ***probability*** of ***dying*** from age *x* to *x+t*
* tpx: the ***probability*** of ***surviving*** from age *x* to *x+t*
* T(x): the ***random variable*** representing a life aged *x*
* ex = Exp[T(x)]: ***expectation*** of the random variable

*Assumption:*

* T(x) is of an ***exponential*** family (ie. fT(x)(t) = A(x)·e-B(x)t+C(x))

*Observe that:*

* tqx = P(T(x) ≤ t) = dx+t / lx = FT(x)(t)
* tpx = P(T(x) > t) = lx+t/lx = 1 - FT(x)(t)

For convenience, 1px = px and 1qx = qx.

*Expectation:*

* Since the assumption of T(x) is of exponential family then:

E[T(x)] = Σt=0 to ∞t·fT(x)(t)

= t·f’T(x)(t) ]t=0 to ∞ + Σt=0 to ∞F’T(x)(t)

= Σt=0 to ∞ tPx

This is where the selected lifestyles (or variables) come into play:

* tqx’(i): the ***probability*** of ***dying*** between the age of *x* to *x+t* due to ***cause*** (or variable) ***i***
* tqxT: the ***total probability*** of ***death*** between the age of *x* to *x+t*

For this model, tqx’(i) and tqx’(j) for all i \ne j are ***independent***.

Furthermore, tpxT = \prodi=1 to ∞ tpx’(i).

*Age:*

The ***final approximation*** for a person’s age is:

x + ex = x + Σt=0 to ∞\prod(1-tpx’(i))

for all causes *i* that apply to this individual.

*Premium*:

The ***final approximation*** for a ***premium*** is computed a little differently. Recall, A = P·a(t) where:

* A = final payment
* P = premium payment
* a(t) = accumulation function

Observe, a(t) = (1+i)t where *i* is the ***interest rate***.

Now let *t* be a ***random variable*** representing a ***person’s age*** at ***death***:

A = P·(1+i)T(x)

So,

E[A] = E[P·(1+i)T(x)]

= P·E[(1+i)T(x)]

= P·Σt=0 to ∞(1+i)t·fT(x)(t)

*Example:*

Assume individual is male, white (Caucasian), fifty-years old, drinks a lot of alcohol, smokes often, and never swims.

The distributions are tqx’(1), tqx’(2), and tqx’(3) respectively (drinking, smoking, and swimming) for all t \in \!\, [1, 100-x]. For total probability of death of the individual with (1) and (2) (since swimming or (3) doesn’t apply):

tqxT = 1 - tpxT = 1 - (tpx’(1) · tpx’(2))

Since the individual is 50, the expected age of death is:

50 + e50 = 50 + Σt=1 to 50 + tp50T

The MATLAB Death Clock Application compiles these equations, and comes to a conclusion of the total probability of death and expected age of death for an individual.

**The Code**

The death clock MATLAB program is based on the *Mathematics* section of this paper. The original plans were to use non-linear regression models (regression analysis modeled by a function which is a nonlinear combination), but following the advice of an accredited York University professor, it is easier to simply use the values at hand instead of estimating a probability distribution function. The significant consequence or drawback of this method is that there are no error bounds. On the other hand, the advantage is that there is no requirement to making an estimate. Therefore, the solution achieved is more accurate.

The MATLAB code is divided into two parts: the Graphical User Interface (GUI), and the computation function. In this report, the main focus is on the computation function, as the GUI is not relevant to the mathematics. The function itself is defined as follows:

function[results] = computelife(tables,PARAMS)

The “results”, “tables”, and “PARAMS” are part of the MATLAB structure. In relation to other structures, using this class is much more convenient in ordering data. The structure “tables” contains all life tables (statistically achieved) pre-loaded into the GUI before initiating the function. A sample table would look as follows:

tables.drugs =

Columns 1 through 6

0.0010 0.0008 0.0006 0.0005 0.0001 0.0001

Each table consists of either 6 or 18 columns (depending on specifications), and is loaded from csv files specifically made for this application. Content of these columns will be elaborated in a later paragraph.

The function starts by finding a maximum age and determining the given age of the person by doing the following:

%% Approximating the Age

agelist = [1,4:5:100,100]; %>The data is given in intervals of 5

[ageid ageid] = min(abs(agelist-PARAMS.age));

age = agelist(ageid); %>approximated age

It is important to note that the age needs to be “approximated” as the data is given in intervals of 5, so the approximated age is just the closet rounded age.

%% Determine Max Age and Show Error if sizes are 0 or do not match

disp('determining max age...');

maxage = 0; %>Set base age to 0

for i = 1:numel(types)

if maxage <= length(tables.(types{i})); %>save the largest of values

maxage = length(tables.(types{i}));

else

if i > 1 %>let us know if the sizes don't match or if there is a 0 length table

error(message(strcat('Not all tables have the same length, eg:',types{i},' and ', types{i-1})));

else

error(message(strcat('The following has length 0:',types{1})));

end

end

end

Error handlers are used to ensure smooth processing, and keep the code devoid from any kinks. Some of these handlers include disallowing males from being pregnant or reading tables that are of zero length or do no match, which therefore leads to the next part involving the cleaning of tables for computational purposes.

As noted before the loaded tables will either have 6 or 18 columns. 6 columns tables will have columns labeled as follows:

“l(Male)”, “l(Female)”, “d(Male)”, “d(Female)”, “q(Male)”, “q(Female)”

The l, d, and q are the same as described in the mathematics portion of this report. Similarly for the 18 columns tables, the same data is displaying with exception that the data is also divided into three ethnicities. So in addition to gender, it will contain columns such as “l(White Male)” or “q(Black Female)”. Hence, there are 18 columns (6 gendered columns ✕ 3 ethnicities). The reasoning behind this is that data was found that was believed to be important for this computation, but was not available for various ethnic groups.

The next portion of the code is an internal function typed out to clean up the table into a structure that is easier to work with (mainly applying to 18 column tables).

%% Clean Table Function

function[res] = clean\_table(table,gender,ethnic,maxage)

com = [1 3 5; 2 4 6]; %>These values are standardized table values

res.lx = table(1:maxage,com(gender+1,ethnic+1)+6); %>Number of people alive

res.dx = table(1:maxage,com(gender+1,ethnic+1)); %>Number of people who died during this period

res.qx = table(1:maxage,com(gender+1,ethnic+1)+12); %>Probability to die in one more year

end

Then the program runs the cleaning algorithm on all the tables, and selects those that fit the gender and ethnic criteria of our computation:

%% Clean Tables and Select Tables

disp('Cleaning Tables...');

for i = 1:numel(types)

disp(strcat(num2str(i),'/',num2str(numel(types)),' tables cleaned'));

if ~(strcmp(types{i},'general')||size(tables.(types{i}),2)<18)%needs to be cleaned differently

tables.(types{i}) = clean\_table(tables.(types{i}),PARAMS.gender,PARAMS.ethnic,maxage);%saves a clean table for the i'th table

end

if ~(strcmp(types{i},'general')||size(tables.(types{i}),2)~=6)%needs to be cleaned differently, no ethnic groups

table = tables.(types{i});

tables.(types{i}).lx = table(1:maxage,PARAMS.gender+1); %Number of people alive

tables.(types{i}).dx = table(1:maxage,PARAMS.gender+3); %Numbe of people who died in this period

tables.(types{i}).qx = table(1:maxage,PARAMS.gender+5); %Probability to die in one more year

end

end

Next is setting up the multiple life table by selecting only the tables that apply to the individual prospect:

types = fieldnames(tables);

disp('Selecting tables based on category');

for i = 1:numel(types)

if PARAMS.(types{i}) == 0

tables=rmfield(tables,types{i});

end

end

A multiple life table is now constructed, but first an empty structure is defined to store the data:

%% Multiple Life table

types = fieldnames(tables);

disp('using the following tables:');

disp(types);

disp('Building Multiple Life Table...');

table=struct(

'lx',tables.general(1:maxage,1),

'dx',tables.general(1:maxage,2),

'px',(1-tables.general(1:maxage,3))

); %>Define an 'empty' multiple life table as the general

%Please note that 'px' is the probability of a person age 'x' surviving 5 more year

Next is to loop through every select table, and apply the mathematics model:

for i = 1:numel(types)

if ~(strcmp(types{i},'general'))%already defined

for j = 1:maxage

table.lx(j) = table.lx(j) + tables.(types{i}).lx(j); %>add all lives

table.dx(j) = table.dx(j) + tables.(types{i}).dx(j); %>add all deaths

table.px(j) = table.px(j)\*(1-tables.(types{i}).qx(j)); %>multiple all probabilities (independence)

end

end

end

With the appropriate table(s), the expected age can now be computed:

%% Compute expected age

disp('Computing expected age...');

expected\_age = sum( 5 \* table.px(ageid:maxage) )+ age;%Sum of all probabilities to survive 5 years at given age

And also the expected return:

%% Compute expected return

disp('Computing expected return...');

% Computation

expected\_return = PARAMS.premium\*(1+PARAMS.intrest)^expected\_age;

% Stylze

expected\_return = sprintf('$%.2f', expected\_return);

expected\_return(2, length(expected\_return) - 6:-3:2) = ',';

expected\_return = transpose(expected\_return(expected\_return ~= char(0)));

Please note that unlike in the mathematics section, there is no annuity so it is sufficient to use the expected age to compute the return. Finally, the results are stored “results” structure:

%% Results

results.table = table;

results.tables = tables;

results.expected\_age = expected\_age;

results.expected\_return = expected\_return;

disp('Done!');

disp(results);

This concludes the computation function. In the other significant part of the program, the GUI application is used to run the following to load the tables:

TABLES.DIR = strcat(mfilename('fullpath'),'/csv/'); %>Finds path of files

TABLES.files = dir(strcat(TABLES.DIR,'\*.csv')); %> Loads a list of files

TABLES.files = {TABLES.files.name}; %> Return the file names

disp(strcat('retrieving tables from ', TABLES.DIR));

disp('Loading files...');

types = TABLES.files;

for i = 1:numel(types)

disp(strcat(num2str(i),'/',num2str(numel(types)),' files loaded'));

if ~(strcmp(types{i},'age')||strcmp(types{i},'ethnic')||strcmp(types{i},'gender'))

% We don't have or need tables for these

if exist(strcat(TABLES.DIR,types{i}),'file')% If the table exists load it, else show error

disp(strcat('Loading \',TABLES.DIR,types{i},'\ ...'));

tables.(strrep(types{i},'.csv','')) = csvread(strcat(TABLES.DIR,types{i}),1,1);

else

disp(strcat('Sorry but \',TABLES.DIR,types{i},'\ does not exists'));

end

end

end

And to make use of the output of the computation Function, the GUI does as follows:

computedlife = computelife(tables,PARAMS);=

%Plot Prob. to Die

scatter(handles.dying\_axes,[1,4:5:100,100],1-computedlife.table.px)

%Plot Prob. to Survive

scatter(handles.surviving\_axes,[1,4:5:100,100],computedlife.table.px)

%Plot No. of People Alive

scatter(handles.peoplealive\_axes,[1,4:5:100,100],computedlife.table.lx)

%Plot No. of People Died

scatter(handles.peopledead\_axes,[1,4:5:100,100],computedlife.table.dx)

%Results

set(handles.deathoutput,'String', computedlife.expected\_age); %display expected age

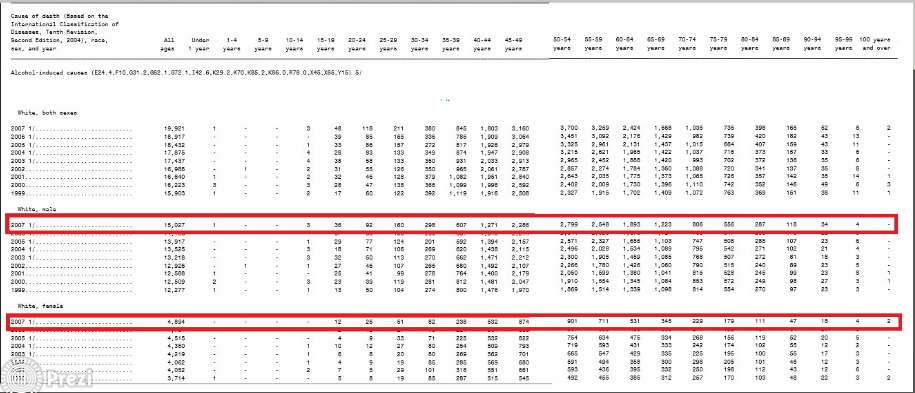
set(handles.payoutoutput,'String',computedlife.expected\_return); %display expected output

In conclusion, the death clock program combines the best of computing and mathematics to create an application designed to estimate the timely death of an individual (with help from mortality tables). The full code is included in the attachments.

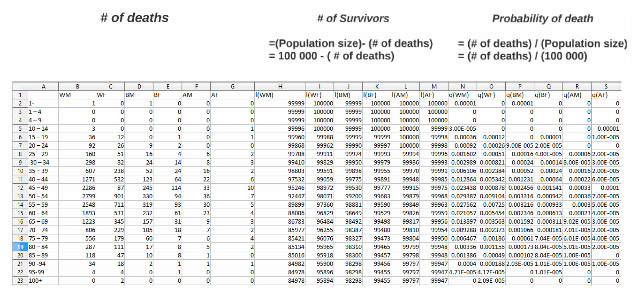
**Variables (Causes & Life Tables)**

The death clock computes its prediction by taking into account various variables relevant to life. Age, gender and tobacco use (smoking) are arguably the more significant variables, but ethnicity, chance of pregnancy, being a driver, having asthma, alcohol use, swimming, travel and cleaning frequency are also factors of life (and death). As previously stated in the *Mathematics* section, there are mortality tables pertaining to each one of these variables calculated by actuaries. Mortality tables (better known as life tables) show the probability that a person will die with respect to age. From this, the probability of surviving any particular year of age or the remaining life expectancy for people at different ages can be derived.

Before coding the death clock, the probabilities of the causes (variables) of death are retrieved. Life tables are easily accessible tables of information found through national databases. Depending on data of need, the tables can be as broad or specific as required. For the purpose of the assignment, the tables used are specific to age, gender, race, and the cause(s) of death, allowing the application to have very accurate information. In addition to the specifics mentioned above, all life tables are given as a sample size (the total number of people in the test group). In the case of the death clock program, the number of deaths is given as per 100,000 people. Once the tables are collected (one for each cause of death), the data can be compiled. To do this, simply place the information given (number of deaths) into an excel spreadsheet. Next, in order to find the number of people who survived, subtract the number of people who died from our test group (i.e 100,000 - #dead). Finally, retrieve the probabilities of death by simply dividing the number of deaths by the test group (i.e #dead/#alive). This process is repeated until the probabilities of death are compiled for every cause chosen. Thus, with these probabilities compiled, the *mathematics* can now be computed for the death clock.



***Figure 1 -*** Original mortality table from Society of Actuaries.



***Figure 2 -*** Custom compiled life tables based on probabilities.

**Application**

Contrary to popular belief, the death clock is applicable to real life situations, and not just for recreation. Similarly to the death clock of this assignment, the “Deadline” app on iOS (iPhone Operating System) uses personal statistical data from another app (HealthKit) such as blood pressure, sleep patterns, along with vital statistics (age, gender, height, etc.) to determine an age of death. Deadline actually provides a countdown clock to your accurate prediction of expiration. The application isn’t meant to frighten you, but rather motivate a healthier lifestyle. The app actually sends reminders users (“You can often change the date by living a healthier lifestyle. Watch the ticker…”) to promote positive living standards. Likewise, the death clock isn’t meant to scare people, but rather motivate better lives.

Another real life circumstance where the death clock is applicable is through health/life insurance. Before receiving health or life insurance, the insurer determines the cost of insurance using mortality tables calculated by actuaries. The death clock similarly determines life expectancy using mortality tables to statistically calculate the age of extinction. While the death clock looks into more variables of life, health and life insurance typically depend on age, gender and use of tobacco. Mortality tables based on these variables are essentially the root of the mathematics in both insurance and the death clock. While the death clock might predict death to an optimistic age, insurance companies predict death to a pessimistic view so that a profit can be made. The death clock is arguably a more precise and accurate prediction than insurance estimates. Like how mortality tables can predict expected ages of death, they can also determine lifespans of dinosaurs!

**Conclusion**

With a great deal of mathematics and MATLAB computing, the death clock can statistically and accurately determine the expected age of death of certain individuals with vital personal information and variables towards death. Despite a fully operational application, there is room for improvement. Such methods for improvement include more possible causes for death, ways to improve expected age of death, and making the program aesthetically pleasing. The idea of the death clock is to inform users of their imminent death, motivate them to live better lifestyles, provide insurance estimates, determine lifespans, and so much more. The death clock is a significant part of society, and continues to grow from just an idea to an everyday application. The death clock promotes the idea of living life to the fullest. The future may not be as long as many hope, but the idea is to die young as late as possible.

**References**

Anderson, Judy F. & Brown, Robert L. *Risk and Insurance.* Society of Actuaries, 2005. Print.

Borreli, Lizette. “The Death Clock: HealthKit’s ‘Deadline’ App Predicts Datte and Time You’ll Kick The Bucket.” *Medical Daily.* IBT Media Inc, 2014. Web. 29 Nov. 2014. <http://www.medicaldaily.com/death-clock-healthkits-deadline-app-predicts-date-and-time-youll-kick-bucket-309602>

Bowers, Newton L. et. al. *Actuarial Mathematics.* 2nd Ed. Society of Actuaries, 1997. Print.

“Complete life table, male, Canada, 2009 to 2011.” *Statistics Canada.* Government of Canada, 2013. Web. 29 Nov. 2014. <http://www.statcan.gc.ca/pub/84-537-x/2013005/tbl/tbl1a-eng.htm>

Dickson, David C.M., Hardy, Mary R., and Water, Howard R. *Actuarial Mathematics for Life Contingent Risks.* 2nd Ed. Cambridge University Press, 2013. Print.

“Mortality and Other Rate Tables.” *Society of Actuaries*. n.p., n.d. Web. 29 Nov. 2014. <http://mort.soa.org/>

“Mortality Tables.” *Centers for Disease Control and Prevention.* National Vital Statistics System, 2011. Web. 29 Nov. 2014. <http://www.cdc.gov/nchs/data/dvs/LCWK1\_2011.pdf>