Appendix 1- Calculating a distribution of lifespan gain: detailed method and full code in Matlab and Python

Principle

In order to calculate not the group mean increase in lifespan but the distribution of individual lifespan gains for people with identical cardiovascular risk status, we carried out Monte Carlo simulations. Specifically, we assumed that the occurrence of death due to a cardiovascular event is stochastic. For each individual we ran daily trials with a pre-determined probability of a fatal event. The probability was increased progressively with age. We modelled the protective effect of preventive medication by reducing the probability of fatal events but using the same stream of random numbers, so that for each individual the first fatal event with preventative medication could be on the same date or later but never earlier. Using this approach, we can carry out simulations for individuals with the same risk profile, with differences in simulated lifespan reflecting the role of chance.

Producing a daily cardiovascular and non-cardiovascular mortality for a particular risk cohort

When studying the national average cardiovascular risk profile, daily probabilities were calculated for fatal cardiovascular events and fatal non-cardiovascular events. This was done using the United Kingdom Office of National Statistics (ONS) reports of numbers of deaths in each 5-year age cohort together with the population sizes in each one-year cohort. Beyond age 95, ONS gives a single aggregate mortality. Therefore, for ages above 95 we used the Gompertz method(1) on a year-by-year basis to obtain a progressive age-related increase in mortality.

When studying a higher cardiovascular risk profile, the cardiovascular daily probabilities were scaled up as required. For example, to study a two-fold higher than average cardiovascular risk, each daily cardiovascular mortality was doubled (with non-cardiovascular left unchanged). For lower than average cardiovascular risk, the cardiovascular daily probabilities were scaled down as required.

A complete set of daily values of cardiovascular and non-cardiovascular mortality was obtained. This consisted of two separate curves that progressively rose from a low daily probability to a high daily probability at very high age.

The sum of the two daily hazards was the daily hazard of all cause mortality, with no treatment. A separate all-cause mortality curve was then calculated in a similar manner but by scaling the cardiovascular mortality by a factor of 0.7, i.e. a cardiovascular mortality hazard reduction of 0.3. This pair of curves of daily all-cause mortality, i.e. without and with treatment, was to be kept the same for every patient who was in this risk stratum.

Simulating the play of chance for many individuals with identical risk

For each level of cardiovascular risk, 10,000 individuals were simulated.
For each individual, a series of random double-precision floating-point values between 0 and 1were obtained using a software generator of numbers whose properties meet statistical criteria for randomness. One value was obtained for each simulated day from the age of 50 years to a an age chosen so high such that it would be almost certain that a fatal event would have occurred before that age. This was 150 years. The individual's entire sequence of random numbers was computed and stored.

Lifespan without treatment was calculated as follows. Starting from the first simulated day, the random number for that patient for that day was compared with the daily hazard of all-cause mortality for this risk stratum for that age without treatment. If the random number was smaller than the daily hazard, the patient was considered to have died on that day.

The same sequence of random numbers was then used to calculate lifespan with treatment. This was done exactly as in the paragraph above but using the daily hazard of all cause mortality *with* treatment, whose numerical values are smaller. Since the random sequence is the same, and the constants it is being compared to are smaller, there will always be fewer (or the same number) of occasions on which the random number is smaller than the daily hazard. This means that the date on which the patient will be considered to have died will be the same or later.

The increment in lifespan between the first run for this patient (without treatment) and the second run (with treatment) is stored as the lifespan gain for this one patient.
This process is repeated for 9,999 patients facing an identical profile of daily cardiovascular mortality.

Describing and displaying the distribution of lifespan gain

The key result from the 10,000 simulated patients with each cardiovascular risk profile was the set of 10,000 lifespan increments. The programme gives the following summary values:

- Mean lifespan gain for the entire group of 10,000
- Proportion of patients who do not gain any lifespan
- Lifespan gain for those patients who do gain any lifespan

We found that the most useful way to depict this graphically was as a bar chart which used 2-dimensions for the patients and 1-dimension for the lifespan gain. The reason to use 2-dimensions for the patient is to allow a large number of patients to be displayed without making each patient's display so narrow as to be indistinguishable. For visual convenience, the display sorts the patients into order of their lifespan gain with the greatest gain at the back. Because displaying 10,000 bars, even using a second dimension to arrange them would produce too complex an image, we programmed the system to sample the patients after the sorting process, so that a representative group of 32x32=1024 patients are shown.

On this bar chart we displayed a dotted line representing the mean lifespan gain across all 10,000 patients. We also coloured the bars representing non-zero lifespan gains to highlight the few patients whose lifespan gain was similar to the group mean lifespan gain of all 10,000. This colouring also helped visualise the fact that the minority who gained lifespan frequently gained far more than the group average.

This entire process of computation could be carried out for either sex and any desired cardiovascular risk level and any desired hazard reduction from therapy, by adjusting values in the program code.

Reader access to software and data
The entire code for the program in Matlab and the equivalent code in Python is given below, to assist readers seeking to replicate this work or use it with different parameter values.

The Matlab code works in Matlab versions 6.5 and above. The numerical aspect of this code also works in Octave which is free software available under the GNU public licence (https://www.gnu.org/software/octave/).
The Python code works in Python 3.

Program code in Matlab

```
% DSB Calculation of *distribution* of lifespan gain from cardiovascular preventative
intervention
%
% This is the Matlab script used for the attached paper. It requires Matlab version
6.5 or upwards
% A corresponding Python script is also available.

clear all

% Number of patients
n_patients=10000;

% Choose whether to calculate for 'male' or 'female'
gender = 'male';

% Default value of 1 below will give calculations for a person with average risk. 2 is
double risk, 0.5 is half risk, etc.
baseline_CVD_hazard_ratio=1;

% Starting age, and effect size, of intervention. For example, a hazard reduction of
0.3 means a hazard ratio of 0.7.
% Below the start age, patients never have the intervention. From the start age
onwards, each patient is "run" with one set of chance, twice - once without
intervention (control) and once with (treated)
hazard_reduction=0.3;
start_age_years= 50;

% Age at which simulation censors, i.e. does not count lifespan extension beyond that
time point. This can be earlier or alter than the highest age in the mortality table.
% If it is younger than the oldest age in the mortality table, the later years in the
mortality table are not used.
% If it is older than the oldest age in the mortality table, for the later years the
mortality of the oldest tabulated year is continued until the age of censoring
% In practice this should be set so high that it is almost certain that all simulated
patients will have had a fatal event then, even with treatment.
max_age_years=150;
max_age_days=max_age_years*365;

% Mortality tables
% Separate mortality tables are entered here for male and female, using 2012 UK
statistics published by the office of national statistics up to age 90, and then
extending beyond that using the Gompertz method with an annual relative increase of
10%.
% First column is the age at the start of the year
% Second column is the IHD mortality of the standard population (remember the
"baseline_CVD_hazard_ratio" allows us to simulate people at higher or lower risk than
this)
% Third column is the non-IHD mortality
% It is mandatory that the first column starts at 0 and increments by 1. Any simulated
patients surviving to the end of the mortality table are considered to continue to
experience the final year's level of risk for the remainder of their simulated life,
until the simulation censors (which may be at a much later age).
mortality_rate_table_male=[
0 0.00000 0.00474
1 0.00000 0.00018
2 0.00000 0.00018
3 0.00000 0.00018
4 0.00000 0.00018
5 0.00000 0.00008
6 0.00000 0.00008
7 0.00000 0.00009
8 0.00000 0.00009
9 0.00000 0.00009
10 0.00000 0.00011
11 0.00000 0.00011
12 0.00000 0.00011
13 0.00000 0.00010
14 0.00000 0.00010
15 0.00000 0.00033
16 0.00000 0.00032
17 0.00000 0.00031
18 0.00000 0.00032
19 0.00000 0.00030
20 0.00001 0.00049
```

21 0.00001 0.00048
22 0.00001 0.00046
23 0.00001 0.00046
24 0.00001 0.00047
25 0.00002 0.00054
26 0.00002 0.00055
27 0.00002 0.00054
28 0.00002 0.00054
29 0.00002 0.00055
30 0.00004 0.00070
31 0.00004 0.00070
32 0.00004 0.00069
33 0.00004 0.00069
34 0.00004 0.00072
35 0.00010 0.00102
36 0.00010 0.00103
37 0.00010 0.00102
38 0.00010 0.00100
39 0.00010 0.00099
40 0.00026 0.00152
41 0.00025 0.00146
42 0.00024 0.00142
43 0.00025 0.00145
44 0.00024 0.00142
45 0.00043 0.00203
46 0.00042 0.00199
47 0.00042 0.00200
48 0.00042 0.00199
49 0.00042 0.00201
50 0.00072 0.00262
51 0.00073 0.00267
52 0.00075 0.00277
53 0.00079 0.00288
54 0.00080 0.00295
55 0.00116 0.00422
56 0.00121 0.00439
57 0.00125 0.00455
58 0.00130 0.00473
59 0.00130 0.00472
60 0.00197 0.00767
61 0.00203 0.00791
62 0.00202 0.00786
63 0.00198 0.00771
64 0.00192 0.00750
65 0.00266 0.00997
66 0.00245 0.00917
67 0.00319 0.01194
68 0.00329 0.01232
69 0.00330 0.01238
70 0.00455 0.01635
71 0.00513 0.01844
72 0.00575 0.02066
73 0.00556 0.01997
74 0.00564 0.02026
75 0.00790 0.02737
76 0.00843 0.02918
77 0.00898 0.03109
78 0.00956 0.03311
79 0.01051 0.03640
80 0.01436 0.04739
81 0.01510 0.04983
82 0.01641 0.05418
83 0.01825 0.06026
84 0.02084 0.06880
85 0.02279 0.07418
86 0.02568 0.08357
87 0.02975 0.09684
88 0.03565 0.11603
89 0.04302 0.14001
90 0.04722 0.15216
91 0.05181 0.16517
92 0.05684 0.17906
93 0.06235 0.19384
94 0.06836 0.20951
95 0.07494 0.22607
96 0.08212 0.24348
97 0.08995 0.26170

```
98 0.09848 0.28066
99 0.10778 0.30026
100 0.11790 0.32038
101 0.12890 0.34086
102 0.14084 0.36152
103 0.15378 0.38212
104 0.16779 0.40240
105 0.18294 0.42206
106 0.19928 0.44076
107 0.21688 0.45812
108 0.23579 0.47376
109 0.25607 0.48726
110 0.27775 0.49821
111 0.30087 0.50622
112 0.32546 0.51091
113 0.35150 0.51196
114 0.37898 0.50912
115 0.40788 0.50224
116 0.43811 0.49125
117 0.46958 0.47623
118 0.50217 0.45734
119 0.53571 0.43491
120 0.57000 0.40935
121 0.60480 0.38119
122 0.63984 0.35102
123 0.67480 0.31948
124 0.70936 0.28723
125 0.74314 0.25493
126 0.77578 0.22318
127 0.80692 0.19256
128 0.83620 0.16355
129 0.86331 0.13658
130 0.88798 0.11198
131 0.91000 0.08998
132 0.92926 0.07074
133 0.94572 0.05428
134 0.95944 0.04056
135 0.97056 0.02944
136 0.97931 0.02069
137 0.98596 0.01404
138 0.99084 0.00916
139 0.99427 0.00573
];

mortality_rate_table_female=[
0 0.00002 0.00377
1 0.00000 0.00014
2 0.00000 0.00014
3 0.00000 0.00015
4 0.00000 0.00015
5 0.00000 0.00007
6 0.00000 0.00008
7 0.00000 0.00008
8 0.00000 0.00008
9 0.00000 0.00008
10 0.00000 0.00009
11 0.00000 0.00009
12 0.00000 0.00008
13 0.00000 0.00008
14 0.00000 0.00008
15 0.00000 0.00015
16 0.00000 0.00015
17 0.00000 0.00015
18 0.00000 0.00015
19 0.00000 0.00014
20 0.00001 0.00021
21 0.00000 0.00020
22 0.00000 0.00020
23 0.00000 0.00020
24 0.00000 0.00020
25 0.00001 0.00026
26 0.00001 0.00026
27 0.00001 0.00026
28 0.00001 0.00026
29 0.00001 0.00026
30 0.00002 0.00039
31 0.00002 0.00039
```

32 0.00002 0.00039
33 0.00002 0.00038
34 0.00002 0.00040
35 0.00005 0.00061
36 0.00005 0.00062
37 0.00005 0.00061
38 0.00005 0.00060
39 0.00005 0.00058
40 0.00008 0.00099
41 0.00008 0.00095
42 0.00008 0.00091
43 0.00008 0.00093
44 0.00008 0.00091
45 0.00014 0.00149
46 0.00013 0.00147
47 0.00013 0.00146
48 0.00013 0.00146
49 0.00013 0.00147
50 0.00023 0.00207
51 0.00023 0.00212
52 0.00024 0.00219
53 0.00025 0.00228
54 0.00025 0.00234
55 0.00041 0.00332
56 0.00043 0.00344
57 0.00044 0.00357
58 0.00046 0.00369
59 0.00045 0.00366
60 0.00074 0.00567
61 0.00076 0.00580
62 0.00075 0.00575
63 0.00073 0.00560
64 0.00072 0.00547
65 0.00115 0.00720
66 0.00107 0.00665
67 0.00137 0.00857
68 0.00141 0.00879
69 0.00141 0.00882
70 0.00218 0.01193
71 0.00242 0.01324
72 0.00267 0.01460
73 0.00257 0.01406
74 0.00258 0.01412
75 0.00474 0.02073
76 0.00497 0.02169
77 0.00518 0.02263
78 0.00542 0.02370
79 0.00580 0.02534
80 0.01035 0.03893
81 0.01058 0.03980
82 0.01101 0.04138
83 0.01168 0.04392
84 0.01287 0.04837
85 0.01854 0.06552
86 0.02024 0.07151
87 0.02214 0.07824
88 0.02474 0.08742
89 0.02787 0.09847
90 0.03061 0.10745
91 0.03362 0.11715
92 0.03692 0.12762
93 0.04054 0.13888
94 0.04450 0.15099
95 0.04884 0.16396
96 0.05359 0.17782
97 0.05879 0.19259
98 0.06447 0.20827
99 0.07069 0.22485
100 0.07748 0.24231
101 0.08489 0.26062
102 0.09297 0.27970
103 0.10178 0.29947
104 0.11137 0.31982
105 0.12180 0.34059
106 0.13313 0.36161
107 0.14542 0.38265
108 0.15875 0.40347

```
109 0.17317 0.42376
110 0.18874 0.44320
111 0.20553 0.46142
112 0.22360 0.47802
113 0.24301 0.49261
114 0.26379 0.50476
115 0.28599 0.51406
116 0.30965 0.52014
117 0.33476 0.52265
118 0.36133 0.52131
119 0.38933 0.51594
120 0.41872 0.50644
121 0.44942 0.49284
122 0.48131 0.47527
123 0.51427 0.45400
124 0.54811 0.42942
125 0.58261 0.40201
126 0.61753 0.37234
127 0.65258 0.34102
128 0.68744 0.30870
129 0.72175 0.27603
130 0.75516 0.24363
131 0.78730 0.21208
132 0.81780 0.18190
133 0.84633 0.15354
134 0.87258 0.12737
135 0.89630 0.10368
136 0.91733 0.08266
137 0.93557 0.06443
138 0.95102 0.04898
139 0.96378 0.03622
];

if strcmp(gender,'male'),
    mortality_rate_table=mortality_rate_table_male;
elseif strcmp(gender,'female'),
    mortality_rate_table=mortality_rate_table_female;
else
    error('Error: unknown gender')
end

% The seed approach below gives reproducible sets of dice across versions of Matlab
from v4 upwards
% If you want to obtain a different, but again reproducible, set of dice, change the 1
to a different integer
% Or set the seed to, for example, 100*sum(clock) to obtain an unreproducible set of
dice
rand('seed',1);

% For testing equivalence between software in different languages it may be desirable
to use a generator that is less sophisticated
% but easy to implement identically in different languages. For this purpose we have
the linear congruential generator from Numerical Recipes
% http://en.wikipedia.org/wiki/Linear_congruential_generator
gen_modulus=2^31;
gen_a=1664525;
gen_c=1013904223;
gen_seed=1;

lifespan_days_treated=[];
lifespan_days_control=[];
lifespan_days_gained=[];

age_years_in_table=mortality_rate_table(:,1);
if or( not(all(diff(age_years_in_table))==1) , min(age_years_in_table)~=0 ),
    error('The first column of the mortality table should be the integers starting
from 0 and counting upwards in steps of 1.')
end
yearly_ihd_mortality_UK_av=mortality_rate_table(:,2);
yearly_nonihd_mortality=mortality_rate_table(:,3);

day=(1:max_age_days)';
year=day/365;
hazard_ratio_treated = ones(size(year)) - hazard_reduction*(year>=start_age_years);

row_number_in_mortality_table=min(floor(year)+1, length(yearly_nonihd_mortality));
```

```
daily_ihd_mortality_UK_av=1-(1-
yearly_ihd_mortality_UK_av(row_number_in_mortality_table)).^(1/365);
daily_ihd_mortality= baseline_CVD_hazard_ratio*daily_ihd_mortality_UK_av;
daily_nonihd_mortality=1-(1-
yearly_nonihd_mortality(row_number_in_mortality_table)).^(1/365);

p_death_daily_untreated= daily_nonihd_mortality + daily_ihd_mortality;
p_death_daily_treated  = daily_nonihd_mortality +
daily_ihd_mortality.*hazard_ratio_treated;

% Patients are alive to start_age_years
p_death_daily_untreated(1:start_age_years*365)=0;
p_death_daily_treated  (1:start_age_years*365)=0;


tic
for patient=1:n_patients,
    % Each patient has a long sequence of "dice" thrown, one for each potential day of
life up to the maximum simulated age (defined at the top of the program).

    % Standard version as used in paper
    one_patients_lifetime_of_dice = rand(max_age_days,1);
    % Alternative version below, only for testing equivalence between different
software designed to implement the same algorithm
    % This is slower and less random, but useful because it can be made to be
identical between different operating systems and languages
    %
    % for day=1:max_age_days
    %    gen_seed=mod(gen_a*gen_seed+gen_c,gen_modulus);
    %    one_patients_lifetime_of_dice(day)=gen_seed/gen_modulus;
    % end

    % Although we are colloquially calling these "dice", they are in fact pseudorandom
numbers uniformly distributed between 0 and 1.
    % There is a simple way of using such numbers to simulate events occurring with a
certain probability.
    % For example, the probability of such a "die" value being less than 0.5 is itself
0.5.
    % Thus testing whether the "die" value is less than 0.5 is a simple way of
simulating an event that has probability 0.5.
    % As another example, the probability of such a "die" having a value less than 0.1
is itself 0.1 again.
    % This works for any probability. Just test whether the die has a value lower than
the probability we desire, and we create an event that has that probability of
occurring.

    % The sequence of dice values for this patient is then evaluated twice.
    % First, with no treatment.

    % Find the dates on which the dice would throw a value below the daily probability
of dying.
    % Those are the dates on which a fatality is considered to occur, if the patient
was alive on that date.
    fatal_dates_untreated =
find(one_patients_lifetime_of_dice<p_death_daily_untreated);
    % We only want the first such date, since the patient is considered to die on that
date.
    if length (fatal_dates_untreated)>0,
control_life_days(patient)=fatal_dates_untreated(1);
    % If there is no date on which a fatal event occurs in this patient, then the
simulation censors at a fixed date.
    else control_life_days(patient)=length(max_age_days);
    end

    % The *same* sequence of dice values as used immediately above, is then used
again.
    % This second time it is *with* treatment.
    % Because the probabilities of fatality are slightly smaller, there may be fewer
occasions on which a fatal event would potentially occur.
    fatal_dates_treated = find(one_patients_lifetime_of_dice<p_death_daily_treated);
    if length (fatal_dates_treated)>0,
treated_life_days(patient)=fatal_dates_treated(1);
    else treated_life_days(patient)=length(max_age_days);
    end
end

extra_life_days = treated_life_days - control_life_days;
```

```
p_benefitting = sum(extra_life_days>0)/n_patients;
disp(' ')
disp(sprintf('%g %s patients',n_patients, gender));
disp(sprintf('Cardiovascular event rate = %g times standard UK
rate',baseline_CVD_hazard_ratio));
disp(sprintf('Proportion not benefitting = %0.5g%%',(1-p_benefitting)*100));
disp(sprintf('Mean extension = %0.5g days',mean(extra_life_days)));
disp(sprintf('Mean extension for those benefitting = %0.5g
days',mean(extra_life_days)/p_benefitting));
disp(sprintf('Time taken = %0.5g seconds',toc));


side_3d=32;
area_3d=side_3d^2;
patients_we_will_plot= area_3d;
if n_patients>patients_we_will_plot,
    i=round(linspace(n_patients/(patients_we_will_plot*2),n_patients-
n_patients/(patients_we_will_plot*2),area_3d));
else
    i=round(linspace(1,n_patients,area_3d));
end
sorted_extra_life_days=sort(extra_life_days);
sample_extra= sorted_extra_life_days(i);
sample_extra_3d= reshape(sample_extra,side_3d,side_3d);

% Figure showing features of the simulation (not needed for publication)
figure (1)
clf
subplot(2,2,1)
plot(year,100*(1-(1-p_death_daily_untreated).^365),'r');
hold on
plot(year,100*(1-(1-p_death_daily_treated  ).^365),'g');
axis([0 100 0 20]);
ylabel('Annual mortality (%)')
xlabel('Age (years)')
title('Mortality')
legend('Untreated','Treated');
subplot(2,2,2)
bar(sort(extra_life_days))
subplot(2,2,3)
dezerod_extra_life_days = extra_life_days(extra_life_days ~=0);
hist(dezerod_extra_life_days)
xlabel('Days of extra life')
ylabel(sprintf('Number of patients out of %g',n_patients))
subplot(2,2,4)
bar3(sample_extra_3d)
axis_bar3=axis;


% Figure for publication
figure(2)
clf
% Rough graph without the colouring we want
h=bar3((12/365)*sample_extra_3d);
a=gca;
set(a,'xtick',[])
set(a,'xticklabel','')
set(a,'ytick',[])
set(a,'yticklabel','')

% Draw blue line at the back of the graph to show the mean gain
mean_extra_days=mean(extra_life_days);
mean_extra_months=mean_extra_days*12/365;
hold on;
l=line([0 side_3d],[0 0],[mean_extra_months mean_extra_months]);
set(l,'color','b','linestyle','--');

% Now apply the colouring we want for the bars
% Patients gaining between 0.75 and 1.5 of the mean gain, are coloured blue,
%    which is specified in the third row of Red Green Blue components in the colormap
below.
% Those gaining something, but less, are yellow (second row)
% Those gaining nothing are white (top row)
% Those gaining more are green (bottom row)
colormap([
    1 1 1
```

```
    1 1 0.5
    0 0 1
    0.5 1 0.5
])

for i = 1:size(sample_extra_3d,2)
    zdata = [];
    for j = 1:size(sample_extra_3d,1)
        if sample_extra_3d(j,i)==0,
            color_status_of_this_bar=1;
        elseif sample_extra_3d(j,i)<mean_extra_days*0.75,
            color_status_of_this_bar=2;
        elseif sample_extra_3d(j,i)>mean_extra_days*1.5,
            color_status_of_this_bar=4;
        else
            color_status_of_this_bar=3;
        end
        zdata = [zdata; ones(6,4)*color_status_of_this_bar];
    end
    set(h(i),'Cdata',zdata)
end
zlabel('Extra lifespan (months)')

t=text(side_3d/2,0,mean_extra_months,'Group mean lifespan gain');
set(t,'fontname','Verdana','verticalalignment','bottom','horizontalalignment','center'
,'color','b','rotation',21)
```

Program code in Python

```
import numpy as np
import time
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

# Number of patients
n_patients=10000;

# Choose whether to calculate for 'male' or 'female'
gender = 'male';

# Default value of 1 below will give calculations for a person with average risk. 2 is
double risk, 0.5 is half risk, etc.
baseline_CVD_hazard_ratio=1;

# Starting age, and effect size, of intervention. For example, a hazard reduction of
0.3 means a hazard ratio of 0.7.
# Below the start age, patients never have the intervention. From the start age
onwards, each patient is "run" with one set of chance, twice - once without
intervention (control) and once with (treated)
hazard_reduction=0.3;
start_age_years= 50;

# Age at which simulation censors, i.e. does not count lifespan extension beyond that
time point. This can be earlier or alter than the highest age in the mortality table.
# If it is younger than the oldest age in the mortality table, the later years in the
mortality table are not used.
# If it is older than the oldest age in the mortality table, for the later years the
mortality of the oldest tabulated year is continued until the age of censoring
# In practice this should be set so high that it is almost certain that all simulated
patients will have had a fatal event then, even with treatment.
max_age_years=150;
max_age_days=max_age_years*365;

# Mortality tables
# Separate mortality tables are entered here for male and female, using 2012 UK
statistics published by the office of national statistics up to age 90, and then
extending beyond that using the Gompertz method with an annual relative increase of
10%.
# First column is the age at the start of the year
# Second column is the IHD mortality of the standard population (remember the
"baseline_CVD_hazard_ratio" allows us to simulate people at higher or lower risk than
this)
# Third column is the non-IHD mortality
# It is mandatory that the first column starts at 0 and increments by 1. Any simulated
patients surviving to the end of the mortality table are considered to continue to
experience the final year's level of risk for the remainder of their simulated life,
until the simulation censors (which may be at a much later age).
mortality_rate_table_male=np.array([
[0,0,0.00474],
[1,0,0.00018],
[2,0,0.00018],
[3,0,0.00018],
[4,0,0.00018],
[5,0,0.00008],
[6,0,0.00008],
[7,0,0.00009],
[8,0,0.00009],
[9,0,0.00009],
[10,0,0.00011],
[11,0,0.00011],
[12,0,0.00011],
[13,0,0.0001],
[14,0,0.0001],
[15,0,0.00033],
[16,0,0.00032],
[17,0,0.00031],
[18,0,0.00032],
[19,0,0.0003],
[20,0.00001,0.00049],
[21,0.00001,0.00048],
[22,0.00001,0.00046],
[23,0.00001,0.00046],
[24,0.00001,0.00047],
```

```
[25,0.00002,0.00054],
[26,0.00002,0.00055],
[27,0.00002,0.00054],
[28,0.00002,0.00054],
[29,0.00002,0.00055],
[30,0.00004,0.0007],
[31,0.00004,0.0007],
[32,0.00004,0.00069],
[33,0.00004,0.00069],
[34,0.00004,0.00072],
[35,0.0001,0.00102],
[36,0.0001,0.00103],
[37,0.0001,0.00102],
[38,0.0001,0.001],
[39,0.0001,0.00099],
[40,0.00026,0.00152],
[41,0.00025,0.00146],
[42,0.00024,0.00142],
[43,0.00025,0.00145],
[44,0.00024,0.00142],
[45,0.00043,0.00203],
[46,0.00042,0.00199],
[47,0.00042,0.002],
[48,0.00042,0.00199],
[49,0.00042,0.00201],
[50,0.00072,0.00262],
[51,0.00073,0.00267],
[52,0.00075,0.00277],
[53,0.00079,0.00288],
[54,0.0008,0.00295],
[55,0.00116,0.00422],
[56,0.00121,0.00439],
[57,0.00125,0.00455],
[58,0.0013,0.00473],
[59,0.0013,0.00472],
[60,0.00197,0.00767],
[61,0.00203,0.00791],
[62,0.00202,0.00786],
[63,0.00198,0.00771],
[64,0.00192,0.0075],
[65,0.00266,0.00997],
[66,0.00245,0.00917],
[67,0.00319,0.01194],
[68,0.00329,0.01232],
[69,0.0033,0.01238],
[70,0.00455,0.01635],
[71,0.00513,0.01844],
[72,0.00575,0.02066],
[73,0.00556,0.01997],
[74,0.00564,0.02026],
[75,0.0079,0.02737],
[76,0.00843,0.02918],
[77,0.00898,0.03109],
[78,0.00956,0.03311],
[79,0.01051,0.0364],
[80,0.01436,0.04739],
[81,0.0151,0.04983],
[82,0.01641,0.05418],
[83,0.01825,0.06026],
[84,0.02084,0.0688],
[85,0.02279,0.07418],
[86,0.02568,0.08357],
[87,0.02975,0.09684],
[88,0.03565,0.11603],
[89,0.04302,0.14001],
[90,0.04722,0.15216],
[91,0.05181,0.16517],
[92,0.05684,0.17906],
[93,0.06235,0.19384],
[94,0.06836,0.20951],
[95,0.07494,0.22607],
[96,0.08212,0.24348],
[97,0.08995,0.2617],
[98,0.09848,0.28066],
[99,0.10778,0.30026],
[100,0.1179,0.32038],
[101,0.1289,0.34086],
```

```
[102,0.14084,0.36152],
[103,0.15378,0.38212],
[104,0.16779,0.4024],
[105,0.18294,0.42206],
[106,0.19928,0.44076],
[107,0.21688,0.45812],
[108,0.23579,0.47376],
[109,0.25607,0.48726],
[110,0.27775,0.49821],
[111,0.30087,0.50622],
[112,0.32546,0.51091],
[113,0.3515,0.51196],
[114,0.37898,0.50912],
[115,0.40788,0.50224],
[116,0.43811,0.49125],
[117,0.46958,0.47623],
[118,0.50217,0.45734],
[119,0.53571,0.43491],
[120,0.57,0.40935],
[121,0.6048,0.38119],
[122,0.63984,0.35102],
[123,0.6748,0.31948],
[124,0.70936,0.28723],
[125,0.74314,0.25493],
[126,0.77578,0.22318],
[127,0.80692,0.19256],
[128,0.8362,0.16355],
[129,0.86331,0.13658],
[130,0.88798,0.11198],
[131,0.91,0.08998],
[132,0.92926,0.07074],
[133,0.94572,0.05428],
[134,0.95944,0.04056],
[135,0.97056,0.02944],
[136,0.97931,0.02069],
[137,0.98596,0.01404],
[138,0.99084,0.00916],
[139,0.99427,0.00573]
])

mortality_rate_table_female=np.array([
[0,0.00002,0.00377],
[1,0,0.00014],
[2,0,0.00014],
[3,0,0.00015],
[4,0,0.00015],
[5,0,0.00007],
[6,0,0.00008],
[7,0,0.00008],
[8,0,0.00008],
[9,0,0.00008],
[10,0,0.00009],
[11,0,0.00009],
[12,0,0.00008],
[13,0,0.00008],
[14,0,0.00008],
[15,0,0.00015],
[16,0,0.00015],
[17,0,0.00015],
[18,0,0.00015],
[19,0,0.00014],
[20,0.00001,0.00021],
[21,0,0.0002],
[22,0,0.0002],
[23,0,0.0002],
[24,0,0.0002],
[25,0.00001,0.00026],
[26,0.00001,0.00026],
[27,0.00001,0.00026],
[28,0.00001,0.00026],
[29,0.00001,0.00026],
[30,0.00002,0.00039],
[31,0.00002,0.00039],
[32,0.00002,0.00039],
[33,0.00002,0.00038],
[34,0.00002,0.0004],
[35,0.00005,0.00061],
```

```
[36,0.00005,0.00062],
[37,0.00005,0.00061],
[38,0.00005,0.0006],
[39,0.00005,0.00058],
[40,0.00008,0.00099],
[41,0.00008,0.00095],
[42,0.00008,0.00091],
[43,0.00008,0.00093],
[44,0.00008,0.00091],
[45,0.00014,0.00149],
[46,0.00013,0.00147],
[47,0.00013,0.00146],
[48,0.00013,0.00146],
[49,0.00013,0.00147],
[50,0.00023,0.00207],
[51,0.00023,0.00212],
[52,0.00024,0.00219],
[53,0.00025,0.00228],
[54,0.00025,0.00234],
[55,0.00041,0.00332],
[56,0.00043,0.00344],
[57,0.00044,0.00357],
[58,0.00046,0.00369],
[59,0.00045,0.00366],
[60,0.00074,0.00567],
[61,0.00076,0.0058],
[62,0.00075,0.00575],
[63,0.00073,0.0056],
[64,0.00072,0.00547],
[65,0.00115,0.0072],
[66,0.00107,0.00665],
[67,0.00137,0.00857],
[68,0.00141,0.00879],
[69,0.00141,0.00882],
[70,0.00218,0.01193],
[71,0.00242,0.01324],
[72,0.00267,0.0146],
[73,0.00257,0.01406],
[74,0.00258,0.01412],
[75,0.00474,0.02073],
[76,0.00497,0.02169],
[77,0.00518,0.02263],
[78,0.00542,0.0237],
[79,0.0058,0.02534],
[80,0.01035,0.03893],
[81,0.01058,0.0398],
[82,0.01101,0.04138],
[83,0.01168,0.04392],
[84,0.01287,0.04837],
[85,0.01854,0.06552],
[86,0.02024,0.07151],
[87,0.02214,0.07824],
[88,0.02474,0.08742],
[89,0.02787,0.09847],
[90,0.03061,0.10745],
[91,0.03362,0.11715],
[92,0.03692,0.12762],
[93,0.04054,0.13888],
[94,0.0445,0.15099],
[95,0.04884,0.16396],
[96,0.05359,0.17782],
[97,0.05879,0.19259],
[98,0.06447,0.20827],
[99,0.07069,0.22485],
[100,0.07748,0.24231],
[101,0.08489,0.26062],
[102,0.09297,0.2797],
[103,0.10178,0.29947],
[104,0.11137,0.31982],
[105,0.1218,0.34059],
[106,0.13313,0.36161],
[107,0.14542,0.38265],
[108,0.15875,0.40347],
[109,0.17317,0.42376],
[110,0.18874,0.4432],
[111,0.20553,0.46142],
[112,0.2236,0.47802],
```

```
[113,0.24301,0.49261],
[114,0.26379,0.50476],
[115,0.28599,0.51406],
[116,0.30965,0.52014],
[117,0.33476,0.52265],
[118,0.36133,0.52131],
[119,0.38933,0.51594],
[120,0.41872,0.50644],
[121,0.44942,0.49284],
[122,0.48131,0.47527],
[123,0.51427,0.454],
[124,0.54811,0.42942],
[125,0.58261,0.40201],
[126,0.61753,0.37234],
[127,0.65258,0.34102],
[128,0.68744,0.3087],
[129,0.72175,0.27603],
[130,0.75516,0.24363],
[131,0.7873,0.21208],
[132,0.8178,0.1819],
[133,0.84633,0.15354],
[134,0.87258,0.12737],
[135,0.8963,0.10368],
[136,0.91733,0.08266],
[137,0.93557,0.06443],
[138,0.95102,0.04898],
[139,0.96378,0.03622]
])


if gender=='male':
    mortality_rate_table=mortality_rate_table_male;
elif gender=='female':
    mortality_rate_table=mortality_rate_table_female;
else:
    raise Exception('Error: unknown gender')

# The seed approach below gives reproducible sets of dice
# If you want to obtain a different, but again reproducible, set of dice, change the 1
to a different integer
# Or set the seed to None to obtain an unreproducible set of dice
np.random.seed(1);

# For testing equivalence between software in different languages it may be desirable
to use a generator that is less sophisticated
# but easy to implement identically in different languages. For this purpose we have
the linear congruential generator from Numerical Recipes
# http://en.wikipedia.org/wiki/Linear_congruential_generator
gen_modulus=2**31
gen_a=1664525
gen_c=1013904223
gen_seed=1

lifespan_days_treated=[];
lifespan_days_control=[];
lifespan_days_gained=[];

age_years_in_table=mortality_rate_table[:,0]
if not(all(np.diff(age_years_in_table))==1) or min(age_years_in_table)!=0:
    raise Exception('The first column of the mortality table should be the integers
starting from 0 and counting upwards in steps of 1.')

yearly_ihd_mortality_UK_av=mortality_rate_table[:,1]
yearly_nonihd_mortality=mortality_rate_table[:,2]

day=np.array([_ for _ in range(1,max_age_days+1)])
year=day/365
hazard_ratio_treated = np.ones(np.size(year)) -
hazard_reduction*(year>=start_age_years);

row_number_in_mortality_table=np.minimum(np.floor(year).astype('int')+0,
np.size(yearly_nonihd_mortality)-1);
# In Python the above constants are 0, 0 and -1, rather than the Matlab values of
1,1,0, respectively,
# because Python has indices starting at 0 rather than 1.

daily_ihd_mortality_UK_av=1-(1-
```

```
yearly_ihd_mortality_UK_av[row_number_in_mortality_table])**(1/365);
daily_ihd_mortality= baseline_CVD_hazard_ratio*daily_ihd_mortality_UK_av;
daily_nonihd_mortality=1-(1-
yearly_nonihd_mortality[row_number_in_mortality_table])**(1/365);

p_death_daily_untreated= daily_nonihd_mortality + daily_ihd_mortality;
p_death_daily_treated  = daily_nonihd_mortality +
daily_ihd_mortality*hazard_ratio_treated;

# Patients are alive to start_age_years
p_death_daily_untreated[0:start_age_years*365]=0;
p_death_daily_treated  [0:start_age_years*365]=0;


# Create empty arrays for survival times in the two arms of the simulation
control_life_days = np.NaN*np.ones(n_patients)
treated_life_days = np.NaN*np.ones(n_patients)

start_time = time.time()

for patient in range(0,n_patients):
    # Each patient has a long sequence of "dice" thrown, one for each potential day of
life up to the maximum simulated age (defined at the top of the program).

    # Standard version as used in paper
    one_patients_lifetime_of_dice = np.random.random(max_age_days)
    #
    # Alternative version below, only for testing equivalence between different
software designed to implement the same algorithm
    # This is slower and less random, but useful because it can be made to be
identical between different operating systems and languages
    #
    # for day in range(0,max_age_days):
    #    gen_seed=np.mod(gen_a*gen_seed+gen_c,gen_modulus)
    #    one_patients_lifetime_of_dice[day]=gen_seed/gen_modulus


    # Although we are colloquially calling these "dice", they are in fact pseudorandom
numbers uniformly distributed between 0 and 1.
    # There is a simple way of using such numbers to simulate events occurring with a
certain probability.
    # For example, the probability of such a "die" value being less than 0.5 is itself
0.5.
    # Thus testing whether the "die" value is less than 0.5 is a simple way of
simulating an event that has probability 0.5.
    # As another example, the probability of such a "die" having a value less than 0.1
is itself 0.1 again.
    # This works for any probability. Just test whether the die has a value lower than
the probability we desire, and we create an event that has that probability of
occurring.

    # The sequence of dice values for this patient is then evaluated twice.
    # First, with no treatment.

    # Find the dates on which the dice would throw a value below the daily probability
of dying.
    # Those are the dates on which a fatality is considered to occur, if the patient
was alive on that date.
    fatal_dates_untreated =
np.nonzero(one_patients_lifetime_of_dice<p_death_daily_untreated);
    # We only want the first such date, since the patient is considered to die on that
date.
    if np.size (fatal_dates_untreated)>0:
        control_life_days[patient]=fatal_dates_untreated[0][0]+1;
    # For Python the first [0] extracts the array of day indices and the second [0]
extracts the first day index in the array
    # And 1 was added to match the convention used in the Matlab program, namely that
if death occurred in on the very first day of simulation, this is considered one day
of survival

    # If there is no date on which a fatal event occurs in this patient, then the
simulation censors at a fixed date.
    else:
        control_life_days[patient]=max_age_days;

    # The *same* sequence of dice values as used immediately above, is then used
again.
```

```
    # This second time it is *with* treatment.
    # Because the probabilities of fatality are slightly smaller, there may be fewer
occasions on which a fatal event would potentially occur.
    fatal_dates_treated =
np.nonzero(one_patients_lifetime_of_dice<p_death_daily_treated);
    if np.size (fatal_dates_treated)>0:
        treated_life_days[patient]=fatal_dates_treated[0][0]+1;
    else:
        treated_life_days[patient]=max_age_days;

extra_life_days = treated_life_days - control_life_days;

p_benefitting = sum(extra_life_days>0)/n_patients;

print(' ')
print('{:g} {:s} patients'.format(n_patients,gender))
print('Cardiovascular event rate = {:g} times standard UK
rate'.format(baseline_CVD_hazard_ratio));
print('Proportion not benefitting = {:0.5g}%'.format((1-p_benefitting)*100));
print('Mean extension = {:0.5g} days'.format(np.mean(extra_life_days)));
print('Mean extension for those benefitting = {:0.5g}
days'.format(np.mean(extra_life_days)/p_benefitting));
print('Time taken = {:0.5g} seconds'.format(time.time()-start_time));


side_3d=31;
area_3d=side_3d**2;
patients_we_will_plot= area_3d;
if n_patients>patients_we_will_plot:
    indices=np.round(np.linspace(n_patients/(patients_we_will_plot*2),n_patients-
n_patients/(patients_we_will_plot*2),area_3d))-1
    # Minus 1 needed because Python indices start at 0, unlike Matlab which starts at
1
else:
    indices=np.round(np.linspace(1,n_patients,area_3d))-1
indices=indices.astype('int')

sorted_extra_life_days=np.sort(extra_life_days);
sample_extra= sorted_extra_life_days[indices];
sample_extra_3d= np.reshape(sample_extra,(side_3d,side_3d));

# Figure showing features of the simulation (not needed for publication)
plt.figure (1)
plt.clf()
plt.subplot(2,2,1)
plt.plot(year,100*(1-(1-p_death_daily_untreated)**365),'r');
plt.plot(year,100*(1-(1-p_death_daily_treated  )**365),'g');
plt.axis([0,100,0,20]);
plt.ylabel('Annual mortality (%)')
plt.xlabel('Age (years)')
plt.title('Mortality')
plt.legend('Untreated','Treated');
plt.subplot(2,2,2)
plt.bar( np.arange(0,np.size(extra_life_days)),np.sort(extra_life_days))
plt.subplot(2,2,3)
dezerod_extra_life_days = extra_life_days[extra_life_days !=0];
plt.hist(dezerod_extra_life_days)
plt.xlabel('Days of extra life')
plt.ylabel('Number of patients out of {:g}'.format(n_patients))
#plt.subplot(2,2,4)
#plt.bar3(sample_extra_3d)
#axis_bar3=axis;


# Figure for publication
fig=plt.figure(2)
plt.clf()
ax = fig.add_subplot(111, projection='3d')

xpos=np.mod(np.arange(0,area_3d),side_3d)
ypos=np.arange(0,area_3d) // side_3d
zpos=np.zeros(area_3d)

dx=np.ones(area_3d)
dy=np.ones(area_3d)
dz=sample_extra*(12/365)
```

```
ax.w_xaxis.set_ticks([])
ax.w_yaxis.set_ticks([])
ax.set_zlabel('Extra lifespan (months)')


# Draw blue line at the back of the graph to show the mean gain
mean_extra_days=np.mean(extra_life_days);
mean_extra_months=mean_extra_days*12/365;
ax.plot([0,0],[0,side_3d],[mean_extra_months,mean_extra_months],color='b',linestyle='-
-')

# Now apply the colouring we want for the bars
# Patients gaining between 0.75 and 1.5 of the mean gain, are coloured blue,
#    which is specified in the third row of Red Green Blue components in the colormap
below.
# Those gaining something, but less, are yellow (second row)
# Those gaining nothing are white (top row)
# Those gaining more are green (bottom row)
colormap=np.array([
    [1, 1, 1],
    [1, 1, 0.5],
    [0, 0, 1],
    [0.5, 1, 0.5],
])

color_data = np.ones((area_3d,4));
i_bar=0
for j in range(0,side_3d):
    for i in range(0,side_3d):
        if sample_extra_3d[j,i]==0:
            color_status_of_this_bar=1;
        elif sample_extra_3d[j,i]<mean_extra_days*0.75:
            color_status_of_this_bar=2;
        elif sample_extra_3d[j,i]>mean_extra_days*1.5:
            color_status_of_this_bar=4;
        else:
            color_status_of_this_bar=3;
        # Matlab requires each of the 6 facets of the bar to have its colour specified
        # whereas Python's pyplot 3d module requires the transparency to be specified
        color_data[i_bar] = np.hstack((colormap[color_status_of_this_bar-1],[1]));
        i_bar+=1

ax.bar3d(xpos, ypos, zpos, dx, dy, dz, color=color_data)
plt.title('3d bars in Python''s matplotlib not quite right: front layers may be
falsely invisible due to z-order problem')

#t=text(side_3d/2,0,mean_extra_months,'Group mean lifespan gain');
#set(t,'fontname','Verdana','verticalalignment','bottom','horizontalalignment','center
','color','b','rotation',21)
```

Appendix 2 - Script of Structured Survey

We are research doctors doing a survey with Imperial College London

We are finding out whether people prefer the certainty of a definite small benefit or the chance of a larger benefit

Imagine you have a choice between 2 treatments:

Treatment A would give you *certainty* of *1* extra year of healthy life (pause)
Treatment B would give you a *2%/ 5%/ 10%/ 20%/ 50% chance* of an extra 10 years of healthy life

Which would you prefer out of A or B?

----- *If they answer B*

To you, the *2%/ 5%/ 10%/ 20%/ 50%* chance of 10 years is more attractive.
If the certain increase in extra life was not just one year, it might be more attractive.
How large would the certain increase in life need to be just as attractive as what you chose which was *2%/ 5%/ 10%/ 20%/ 50%* chance of 10 years.

Demographics

Do you mind telling us how old you are?

Have you ever had a heart attack or a stroke?

Male or Female?

Appendix 3 – Characteristics of survey respondents

Table 3(a)- Influence of age on survey responses

	Probability offered for the 10 extra years option				
	2%	**5%**	**10%**	**20%**	**50%**
Youngest age quartile (18 to 26 years, 110 respondents)					
Percentage of respondents choosing *certainty* of 1 extra year	62	58	54	40	12
Percentage of respondents choosing *chance* of 10 extra years	38	42	46	60	88
Second age quartile (27 to 34 years, 89 respondents)					
Percentage of respondents choosing *certainty* of 1 extra year	67	55	52	60	23
Percentage of respondents choosing *chance* of 10 extra years	33	45	48	40	77
Third age quartile (35 to 50 years, 98 respondents)					
Percentage of respondents choosing *certainty* of 1 extra year	84	64	65	42	12
Percentage of respondents choosing *chance* of 10 extra years	16	36	35	58	88
Oldest age quartile (51 to 88 years, 99 respondents)					
Percentage of respondents choosing *certainty* of 1 extra year	59	45	50	32	25
Percentage of respondents choosing *chance* of 10 extra years	41	55	50	68	75

Table 3(b)- Influence of gender on survey responses

	Probability offered for the 10 extra years option				
	2%	**5%**	**10%**	**20%**	**50%**
Female					
Percentage of respondents choosing *certainty* of 1 extra year	73	67	61	45	10
Percentage of respondents choosing *chance* of 10 extra years	27	33	39	55	90
Male					
Percentage of respondents choosing *certainty* of 1 extra year	64	49	47	40	23
Percentage of respondents choosing *chance* of 10 extra years	36	51	53	60	78

References

1. Makeham WM. On the Law of Mortality and the Construction of Annuity Tables. J Inst Actuaries and Assur Mag. 1860;8:301-310.