Paper SP08

Statistical Simulations for Sample Size Calculation with PROC IML

Iza Peszek, Merck & Co., Inc

ABSTRACT

This paper describes some techniques for sample size calculation via statistical simulation with PROC IML. The simulation approach may turn out to be the most practical way to calculate sample size when trial design is very complex. The concept is illustrated in a case example: showing how to validate a sample size calculation program from the public domain using simulation methods. The basic concepts of statistical simulation are described and corresponding IML code is provided.

INTRODUCTION

Calculation of the sample size is a typical activity performed when planning clinical trials. For simple trial designs with fixed duration, this task is often relatively simple and reduces to using some well known formulas. Complications arise when the trial design is more exotic, and especially when dealing with variable duration trials, for example, when the stopping rule depends on the number of observed clinical events.

There is a vast literature covering sample size calculation for such trials. Most described methods assume some simplifications on the underlying distribution of time-to-event, and are not well equipped to deal with any but the most simple trial designs. For example, a proportional hazard model is often assumed, even though this assumption is rarely realistic. Another example is stratified analysis. Most described methods assume uniform population and perform poorly in case of multiple stratification variables.

There are papers describing methodologies for dealing with complex trial designs, but the solutions they present are far from trivial and require substantial learning time. Statisticians asked to calculate a sample size for a trial are often hard-pressed to deliver the solution on short notice, and may not have the luxury to study the described methods in sufficient detail.

An alternative to such theoretical solutions is to use simulation techniques to assist in sample size calculation. This approach can be also used to predict a number of other characteristics in the trial (e.g., the number of clinic visits, or the amount of study drug), and allows for flexibility of assumptions and scenarios. Thus, sensitivity calculations can be performed, comparing sample sizes under a variety of assumptions. Moreover, once this technique is mastered, it can be applied to almost any design.

This paper illustrates calculation of the sample size via statistical simulation with PROC IML. We illustrate the concept by validating a published method of sample size calculation ([1]) via simulation techniques.

EXAMPLE 1: BASIC CONCEPTS

Most time-to-event trials can be described using the following concepts:

- There are several patient populations (different drug regimens, and possibly different strata).
- Each population is assumed to be homogeneous with respect to the distributions of time-to-enrollment, time-to-event, time-to-loss-to-follow-up (leaving the trial), and time-to-crossover (switching treatment group).
- Sometimes there is a time limit on the trial, so that the trial will terminate after a specified amount of time.
- Sometimes the trial will continue until a pre-specified number of clinical events are observed.
- Sometimes the trial will terminate whenever a pre-specified number of clinical events are observed, or after some pre-specified number of months, whichever comes first.

These factors will be translated into a statistical model, and simulation will be used to "run a trial".

We will start with defining some variables. There are several different populations in the trial (e.g., different treatment groups). There are N_i patients in the *i*-th population, who enroll in the trial according to distribution F_i, experience an event according to distribution G_i, and drop-out for various reasons according to distribution H_i. The time of enrollment is from time 0 (start of the trial), and the times of event and loss-to-follow-up are from the time of enrollment. For now, we will assume no crossovers.

For each of the trial populations, the simulation model will need to generate N_i variables according to distribution F_i, N_i variables according to distribution G_i, and N_i variables according to distribution H_i.
If there is no time limit on the trial, then the loss-to-follow-up time T_D (from the start of the trial) is F_i+H_i .
If there is a time limit on the trial, e.g., the trial will stop at time T (from the start of the trial), then the loss-to-follow-up time T_D= min (T, F_i+H_i). A patient experiences an event at the time F_i +G_i (as long as $F_i+G_i \leq T_D$), or is censored at the time F_i+H_i (if $F_i+G_i>T_D$).

For simplicity we assume 2 populations (2 treatment groups), having N_1=2000 and N_2 = 2000 patients respectively. We will also assume that G_i and H_i are exponential distributions with one-year event rates p_{e1} = 0.1, p_{e2} = 0.12 and one-year loss-to-follow-up rates p_{d1}=0.2, p_{d2}=0.25, and that the distribution of time of enrollment is uniform over a 0.25 year period in both populations. The study will terminate 3 years after enrollment starts. More complicated cases will be considered later.

It is a well known fact that to generate a distribution according to cdf F, we calculate $F^{-1}(X)$, where X has a uniform distribution over the interval (0,1). For the exponential distribution, F(t) = 1-exp(-λt), and $F^{-1}(t)$ =-ln(1-t)/λ. The rate of events at unit time is 1-exp(-λ), so λ=-ln(1-p), where p is a one-year rate of events. Thus, to generate an exponential distribution corresponding to one-year rate of p, we will calculate -ln(1-t)/ λ or ln(1-t)/ln(1-p), where t is uniform over (0,1) and can be easily generated by a UNIFORM function.

Here is PROC IML code to implement this scenario:

```
n1 = 1000;  * group 1 sample size;
n2 = 1000;  * group 2 sample size;

pe1 = 0.1;  * prob of event in group 1 after 1 year;
pe2 = 0.12; * prob of event in group 2 after 1 year;
pd1 = 0.20; * prob of loss-to-follow-up in group 1 after 1 year;
pd2 = 0.25; * prob of loss-to-follow-up in group 2 after 1 year;

studyduration  = 3;

* initialize a dummy used for random number generation;
  myran1=j(n1, 1, 0);
  myran2=j(n2, 1, 0);


* generate enrollment time according to uniform distribution over 0.25 years;
  enroll = 0.25*(uniform(myran1))// 0.25*(uniform(myran2));

* generate event time according to exponential distribution;
* this is time from enrollment;
  event_  = (log(1-uniform(myran1))/log(1-pe1))
          //(log(1-uniform(myran2))/log(1-pe2));
* calculate event time from time 0 (study start);
  event = enroll+event_;

* generate loss-to-follow-up time according to exponential distribution;
* this is time from enrollment;
  loss_  = (log(1-uniform(myran1))/log(1-pd1))
         //(log(1-uniform(myran2))/log(1-pd2));
* calculate loss-to-follow-up time from time time 0 (study start);
  loss = enroll+loss_;
* in addition, administrative censoring occurs at specified time
  after study started;
  censor = (loss<studyduration)#loss+studyduration*(loss>=studyduration);

* the following matrix has time of event or loss-to-follow-up or censoring -
    whichever comes first - in the first column ,
    an indicator of censoring
    (1 if this is time of loss-to-follow-up /censor, 0 otherwise)
    in the 2nd column,
    and treatment group (0/1) in the 3rd column;
    status = (event||censor)[,><]||(censor<event)||(myran1//(1+myran2));

 * output data with log-rank test statistics values;
   create status from status [colname={'time' 'censor' 'trt'}];
   append from status;
```

Once we have run this code, we will have a simulated data set STATUS with trial results. For simplicity, we will assume that the final statistical analysis will compare distributions of time-to-event via log-rank statistics. We can run a log-rank test on this data set and determine whether we should reject the null hypothesis that the distribution of time-to-event is the same in both groups. By repeating this process a number of times, and calculating the percent of rejections of the null hypothesis, we will estimate the power of the trial. Typically, 5000 iterations give a reasonable estimate of the power.

There are several ways to run the log-rank test. We can leave PROC IML and conduct the test via PROC PHREG, or we can write an IML module for the log-rank test and run it within PROC IML. The latter solution seems to be more practical, as the CPU time is significantly shorter. The module for log-rank statistics is not annotated below, as it is not the focus of this paper.

Here is the final code for this simple example:

```
proc iml;

* define a module to calculate log-rank statistics;
* takes a matrix as input
* in first column are event times (death or censoring)
* in second column are censoring indicators (1 if censored, 0 otherwise)
* in third column are treatment codes(1/0);

start logrank(times);
   * sort matrix;
   b=times;
   times[rank(times[,1]),]=b;
   times=times||j(nrow(times),1,1);
   trt1id=times[,4]#(times[,3]=0);
      trt2id=times[,4]#(times[,3]=1);

      ntrt1 = sum(trt1id);
      ntrt2 = sum(trt2id);
      * indicator if death occurred;
      death1 = (trt1id)#(1-times[,2]);
      death2 = (trt2id)#(1-times[,2]);
      death=death1+death2;

      cumgone1 = cusum(trt1id);
      cumgone2 = cusum(trt2id);
      cumgone1_={0}//cumgone1[1:nrow(cumgone1)-1];
      cumgone2_={0}//cumgone2[1:nrow(cumgone2)-1];
      * number at risk before each event time point;
      atrisk1 = ntrt1-cumgone1_;
      atrisk2 = ntrt2-cumgone2_;
      atrisk=atrisk1+atrisk2;

      disp = atrisk||atrisk1||atrisk2||death||death1||death2||times[,1];
      create tmpdata from disp;
      append from disp;
      summary class {col7} var {col1 col2 col3} stat{max} opt{noprint save};
      summary class {col7} var {col4 col5 col6} stat{sum} opt{noprint save};
      close ;
      call delete(tmpdata);
      atriskf =  col1||col2||col3||col4||col5||col6||col7;
      small = loc(col1<2);

      if nrow(small)>0 then do;
        min0 = min(small)-1;
        atriskf = atriskf[1:min0,1:5];
      end;

      w = atriskf[,5]-(atriskf[,2]#atriskf[,4])/atriskf[,1];
      v = (atriskf[,2]#atriskf[,3]#atriskf[,4]#(atriskf[,1]-
          atriskf[,4]))/(atriskf[,1]#atriskf[,1]#(atriskf[,1]-1));
      sum_w=sum(w);
      sum_v=sum(v);
      chisq = sum_w*sum_w/sum_v;
      return(chisq);
   finish logrank;

   *-----------------------------------------------------------------------------;
   * start simulation
   *-----------------------------------------------------------------------------;

   n1 = 100; * group 1 sample size;
   n2 = 100; * group 2 sample size;
```

```
pe1 = 0.1; * prob of event in group 1 after 1 year;
pe2 = 0.12;* prob of event in group 2 after 1 year;
pd1 = 0.20;* prob of loss-to-follow-up in group 1 after 1 year;
pd2 = 0.25;* prob of loss-to-follow-up in group 2 after 1 year;

studyduration  = 3;

* initialize a dummy used for random number generation;
  myran1=j(n1, 1, 0);
  myran2=j(n2, 1, 0);

* initialize matrix to which we will append values of chi-square statistics
            from logrank test;
  result = j(1,2,0);


do i=1 to 5000 by 1;

* generate enrollment times, event times and loss-to-follow-up times
  enroll = 0.25*(uniform(myran1)) // 0.25*(uniform(myran2));
  event_ = (log(1-uniform(myran1))/log(1-pe1))
         //(log(1-uniform(myran2))/log(1-pe2));
  event  = enroll+event_;
  loss_  = (log(1-uniform(myran1))/log(1-pd1))
         //(log(1-uniform(myran2))/log(1-pd2));
  loss   = enroll+loss_;
  censor = (loss<studyduration)#loss+studyduration*(loss>=studyduration);

  status = (event||censor)[,><]||(censor<event)||(myran1//(1+myran2));

* run logrank test and append test statistics to result matrix;
  chisq = logrank(status);
  result = result//(chisq||i);

end;

* output data with logrank test statistics values;
  create result from result [colname={'chisq' 'runid'}];
  append from result;

quit;

* calculate percent of rejections of null hypothesis by logrank test;
  data result;
  set result (where=( runid>0));
    prob = 1-probchi(chisq,1);
    if prob<0.05 then reject=1; else reject=0;
  run;

proc freq data=result;
  tables reject;
  title "Percent of rejections of null hypothesis";
run;
```

EXAMPLE 2

The paper by Shih ([1]) describes the use of a SAS® program to calculate sample size in complex time-to-event trials. We want to validate the results of this SAS program via simulations. The author of the paper describes several examples and gives the results generated by their program. The first example is as follows: a hypothetical 2-arm trial with 2 years of average-follow up. The probability of having an event in the control group after 2 years is 30%. The hazard rate is piecewise constant such that λ_1: λ_2: λ_3: λ_4 = 1: 2: 2: 2, where study is divided into four 6-months periods and λ_i is the hazard rate in the *i*-th period. The ratio of the hazard in the treatment group to that in the control group is assumed constant and equal to 0.65. The program described in the paper calculates a total sample of 683 to achieve 80% power.

We will validate the results of this example via IML code. First, we need to translate the problem into the language of a statistical model. Constant hazard rate implies exponential distribution, so time-to-event is piece-wise exponential. In order to determine the parameters of these exponential distributions, note that the assumptions imply that in the control group, $1-(1-P_1)*(1-P_2)^3 = 0.3$, where $P_1= 1-\exp(-\lambda_1)$ is the probability of event in period 1 and $P_2 = 1-\exp(-2\lambda_1)$

is the probability of event in periods 2-4, respectively. Hence, $0.7 = \exp(-\lambda_1)(\exp(-2\lambda_1))^3$ and $\lambda_1=-\ln(0.7)/7$. In periods 2-4, the hazard of event in the control group is $2\lambda_1$. In the treatment group, hazard in each period is 0.65 of the hazard in the control group. All patients enter the study simultaneously and there are no losses to follow-up. Study duration is 2 years. In the previous example, our time unit was year. In this example, it is more convenient to use 6-months as the time unit, so study duration is 4. Based on a total sample size of 683, we set N_1 = 342 and N_2=341.

Now we can write PROC IML statements to implement this statistical model.
For brevity, we quote only the statements to simulate one repetition of the study outcome. Calculating log-rank statistics and running iterations can be done in the same way as in Example 1.

```
n  = 683; * total sample size;
n1 = 342; * control sample size;
n2 = 341; * treatment sample size;


hctrl1 = -1*log(0.7)/7;  * hazard, period 1, control grp;
hctrl2 = 2*hctrl1;       * hazard, periods 2-4, conrol grp;
htrt1  = 0.65*hctrl1;    * hazard, period1, trt grp;
htrt2  = 0.65*hctrl2;    * hazard, period2, trt grp;


* initialize a dummy used for random number generation;
  myran1=j(n1, 1, 0);
  myran2=j(n2, 1, 0);

* create treatment groups and study duration matrices;
  trt=j(n2, 1, 1);
  control=j(n1,1,0);
  studyduration=j(n,1,4);
  p1duration = j(n,1,1);  * duration of period 1


* generate time of event according to 2-stage distribution;

* time of event according to initial distribution (in period1);
  period1 = -1#log(1-uniform(myran1)/hctrl1;
         // -l#log(1-uniform(myran2)/htrt1;

* time of event according to 2nd stage distribution,
     applies only if no event in period1 (conditional on no event in period1);
  period2 = p1duration +(-1#log(1-uniform(myran1)/hctrl2
                      //(-1#log(1-uniform(myran2)/htrt2);
* finally, “absolute” time of event;
  teventBX = period1#(period1<=p1duration)+period2#(period1>p1duration);

* time of event is censored if it occurs study ended;
  censor = (studyduration<teventBX);

* final time of event is the smaller of study ending time and event time;
  event = studyduration#censor+teventBX#(1-censor);

* final matrix has time of event/censoring in 1st column, censoring indicator in the
  second column and treatment group indicator in the third column;

  times0 = event||censor||(control//trt);
```

EXAMPLE 3

Let us now assume that in the previous example the study duration is 2.5 years, participants enter the study uniformly for 1 year and are followed for a minimum of 1.5 years. In addition, 10% of each group are crossovers (switch to another group) and 10% are lost to follow up after 2 years. For loss-to-follow-up and for crossover from control to treatment group, the hazard rate is constant. For crossover from treatment to control group, the hazard rate is piecewise constant such that λ_1: λ_2: λ_3: λ_4 : λ_5 = 1: 1: 2: 2: 2 when the study is divided into five 6-months intervals. We assume that patients who crossed over to another treatment group immediately experience the hazard of clinical event specific to this group. Hence we need to simulate the following: time of entry into the study, time of lost to follow up, time of event conditional on no crossover, time of crossover, and time of event after crossover (conditional on no event and no lost to follow up before crossover).

To simulate time of loss-to-follow-up and time of crossover from treatment to control group, note that the assumptions imply that $0.9= \exp(-\beta)\exp(-\beta)\exp(-\beta)\exp(-\beta)$, where β is a hazard rate for lost to follow up. Hence the (constant) hazard of lost to follow, $\beta=-\ln(0.9)/4$. This is also the hazard of crossover from treatment to control. For crossovers from control to treatment group, $0.9 = \exp(-\gamma)\exp(-\gamma)\exp(-2\gamma)\exp(-2\gamma)$, so $\gamma = -\ln(0.9)/6$.

Here is the IML code to simulate the study data.

```
hctrl1 = -1*log(0.7)/7;  * hazard for event, period 1, control grp;
hctrl2 = 2*hctrl1;       * hazard for event, periods 2-5, control grp;
 htrt1 = 0.65*hctrl1;    * hazard for event, period 1, trt grp;
 htrt2 = 0.65*hctrl2;    * hazard for event, period2, trt grp;
 hloss = -1*log(0.9)/4;  * hazard for loss to follow up;

* hazards of crossover:;

htc12 = -1*log(0.9)/6;   * treatment to control, periods 1 and 2;
htc34 =  2*htc12;        * treatment to control, periods 3,4,5;
  hct = -1*log(0.9)/4;   * control to treatment, all periods;

* initialize a dummy used for random number generation;
myran1=j(n1, 1, 0);
myran2=j(n2, 1, 0);

* create duration matrices;
studyduration_c = j(n1,1,5); * study duration for control;
studyduration_t = j(n2,1,5); * study duration for treatment group;
   p1duration_c = j(n1,1,1); * period 1 duration – control;
   p1duration_t = j(n2,1,1); * period 1 duration – treatment group;
  p12duration_c = j(n1,1,2); * period 1 + period 2  duration – control;
  p12duration_t = j(n2,1,2); * period 1 + period 2  duration – treatment group;


* for control group;

     enroll_c = 2#uniform(myran1);* enrollment time – uniform over priods 1,2;

 timeofloss_c = enroll_c - log(1-uniform(myran1))/hloss;
timesofstop_c = timeofloss_c||studyduration_c;
      tloss_c = timesofstop_c[,><];  * minumum of lost-to-follow and study stop;

period1_c = -1#log(1-uniform(myran1))/hctrl1; * time of event in period 1;
period2_c = p1duration_c - log(1-uniform(myran1))/hctrl2;
      * time of event in periods 2-5, conditional on no event in period 1;

teventBX_c =   enroll_c + period1_c#(period1_c<=p1duration_c) +
                          period2_c#(period1_c>p1duration_c);
         * the unconditional time of event, from start of study, if no crossovers,
           control group;

tcross_c = enroll_c-log(1-uniform(myran1))/hct;
          * time of crossover from control to trt, from study start;

* generate time of event after noncompliance started;
period1X_c = -1#log(1-uniform(myran1))/htrt1;   * time of event in period 1;
period2X_c = p1duration_c - log(1-uniform(myran1))/htrt2;
             * time of event in periods 2-5, conditional on no event in period 1;

teventX0_c = tcross_c + period1X_c#(period1X_c<=p1duration_c) +
                        period2X_c#(period1X_c>p1duration_c);
             * unconditional time of event from start of study, control group;

tevent_c = teventBX_c#(teventBX_c<=tcross_c)+teventX_c#(teventBX_c>tcross_c);
          * observed event time is teventBX_c of it occurs before x-over or
            teventX_c otherwise;

   censor_c = (tloss_c<tevent_c);
eventtime_c = tevent_c#(1-censor_c)+tloss_c#censor_c;
              * event is observed only if it occurs before loss to follow and before
                study end;
```

```
* for treatment group;

     enroll_t = 2#uniform(myran2);* enrollment time;

 timeofloss_t = enroll_t - log(1-uniform(myran2))/hloss;
timesofstop_t = timeofloss_t||studyduration_t;
      tloss_t = timesofstop_t[,><];  * minumum of lost-to-follow and study stop;

period1_t = -1#log(1-uniform(myran2))/htrt1; * time of event in period 1;
period2_t = p1duration_t - log(1-uniform(myran2))/htrt2;
    * time of event in periods 2-5, conditional on no event in period 1;
teventBX_t = enroll_t + period1_t#(period1_t<=p1duration_t) +
                        period2_t#(period1_t>p1duration_t);
             * the unconditional time of event, from start of study, if no
               crossovers, control group;

tcross12_t = -1#log(1-uniform(myran2))/htc12; * time of x-over in periods 1-2;
tcross34_t = p12duration_t - log(1-uniform(myran2))/htc34;
           * time of x-over in periods 3-5, conditional on no x-over in periods 1-2;
  tcross_t = enroll_t + tcross12_t#(tcross12_t<=p12duration_t) +
                        tcross34_t#(tcross34_t>p12duration_t);
             * time of x-over from treatment to control group, from study start;

* generate time of event after noncompliance started;
period1X_t = -1#log(1-uniform(myran2))/hctrl1;
             * time of event according to initial distribution;
period2X_t = p1duration_t - log(1-uniform(myran2))/hctrl2;
             * time of event in periods 2-5, conditional on no event in period 1;

teventX_t = tcross_t + period1X_t#(period1X_t<=p1duration_t) +
                       period2X_t#(period1X_t>p1duration_t);
          * unconditional time of event from start of study, treatment group;

 tevent_t = teventBX_t#(teventBX_t<=tcross_t)+teventX_t#(teventBX_t>tcross_t);
          * observed event time is teventBX_c of it occurs before x-over
             or teventX_c otherwise;

   censor_t = (tloss_t<tevent_t);
eventtime_t = tevent_t#(1-censor_t) + tloss_t#censor_t;
          * event is observed only if it occurs before loss to follow and before
            study end;

* create matrix for logrank routine;
finaldata = (eventtime_c||censor_c||(myran1))//(eventtime_t||censor_t||(1+myran2));
chisq=logrank(finaldata);
```

ADDITIONAL REMARKS

We illustrated the simulation of a clinical trial via some simple examples. In these examples, we started with a predetermined sample size and estimated the power of the study via simulations. To determine a sample size for a specified power, a search algorithm should be implemented, trying different sample sizes until the desired power is achieved. A practical way would be to select a reasonable initial estimate using a simple formula based on the log-rank test and recursively refine the initial estimate until the specified power is achieved. One way to speed up the simulation time would be to conduct a "search for N" with a small number of repetitions (say, 200). This will quickly produce a rough estimate of the power. Once we determine the sample size which gives a result close to the desired power, we can verify the accuracy by increasing the number of repetitions to a more reasonable number (5000 repetitions or so).

One of several search algorithms is as follows:

1. Calculate the initial sample size N based on the log-rank formula:
   $D = 4(z_{\alpha/2} + z_{\beta})^2/\theta^2$, where D denotes required number of events, z_{α} is an upper α-th percentile of standard normal distribution and θ is a log-hazard ratio. Given D, sample sizes in treatment groups 1 and 2 can be approximated from the formula $D=N_1{*}P_1+N_2{*}P_2$, where N_1, N_2 denote sample sizes, and P_1, P_2 denote expected rates of events over an average study duration.
2. Perform low-repetition simulations to obtain power estimate.

3. If estimated power is greater than desired power, N_{new} = N+Δ, where Δ=0.5*N. If estimated power is less than desired power, N_{new} = N-Δ. Retain Δ. In the next step, Δ=0.5Δ.
4. Continue incrementing (or decrementing) sample size by Δ (halving Δ in each step) until reasonable proximity to the desired power is achieved. Once this happens, perform high-repetition simulations to refine the sample size.

It is easy to extend this approach to calculate additional characteristics of the trial. For example, if it is desired to calculate the number of clinic visits, then we would develop a model which would generate random variables for each clinic visit and incorporate these calculations into IML code. Typically, we would assume that the time of the visit is distributed uniformly over some number of days centered at the protocol-prescribed time. Of course, more complicated assumptions are also possible, although the law of diminishing returns will start working here. That is, it may not be worthwhile to over-complicate your modeling assumptions.

The simulation approach may be the most practical way to calculate sample size when trial design is very complex. Since powerful computers are readily available, the simulation approach is much more feasible nowadays than it was a decade ago. Once this technique is mastered, it is easily extended to any trial design. Usually the hardest task will be translating the language of clinical assumptions into a statistical model. Once this task is accomplished, implementation of simulations for the statistical model is relatively straightforward.

REFERENCES

1. "Sample Size Calculation for Complex Clinical Trials with Survival Endpoints" by Joanna H. Shih. Controlled Clinical Trials 16:395-407 (1995).
2. "Modelling Survival Data in Medical Research" by D. Collet. Chapman and Hall, 1994.

ACKNOWLEDGMENTS

The author thanks Dr. J. Shih for permission to quote examples from her paper, and John Troxell for a careful review of this publication and valuable comments.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Iza Peszek
Merck & Co., Inc
126 Lincoln Ave
Clinical Biostatistics RY 34-A320
Rahway NJ 07065-0900 Address
Work Phone: 594 732 3623
Email: izabella_peszek@merck.com