Power Analysis

Simulation-Based Power Analysis

Tutorial material from IQSS

1. Introduction

1.1 Basic Concepts

It is an important step to calculate statistical power in a research design. In a research design, we use statistical power to measure the probability that a null hypothesis is correctly rejected. Usually, researchers need to know the needed sample size to reject the null hypothesis at a given power level, while in other cases, people calculates the power when the sample size is fixed.

More often, in a randomized controlled trial with two groups, we can use a formula to calculate the needed sample size to reject the null hypothesis. We will use an example to show how we do this. For instance, when we plan to perform a test of a hypothesis comparing the proportions of successes of tossing coins of faces in two independent populations, we would list the following null and alternative hypothesis respectively:

$$H_0 : p_1 = p_2$$

$$H_1 : p_1 \neq p_2$$

where $p_1 = p_2$ are the proportions in the two populations for comparison. In order to make sure the test has a specific power, we can use the following formula to determine the sample sizes:

$$N = 2(\frac{z_{1-\frac{\alpha}{2}} + z_{1-\beta}}{ES})^2$$

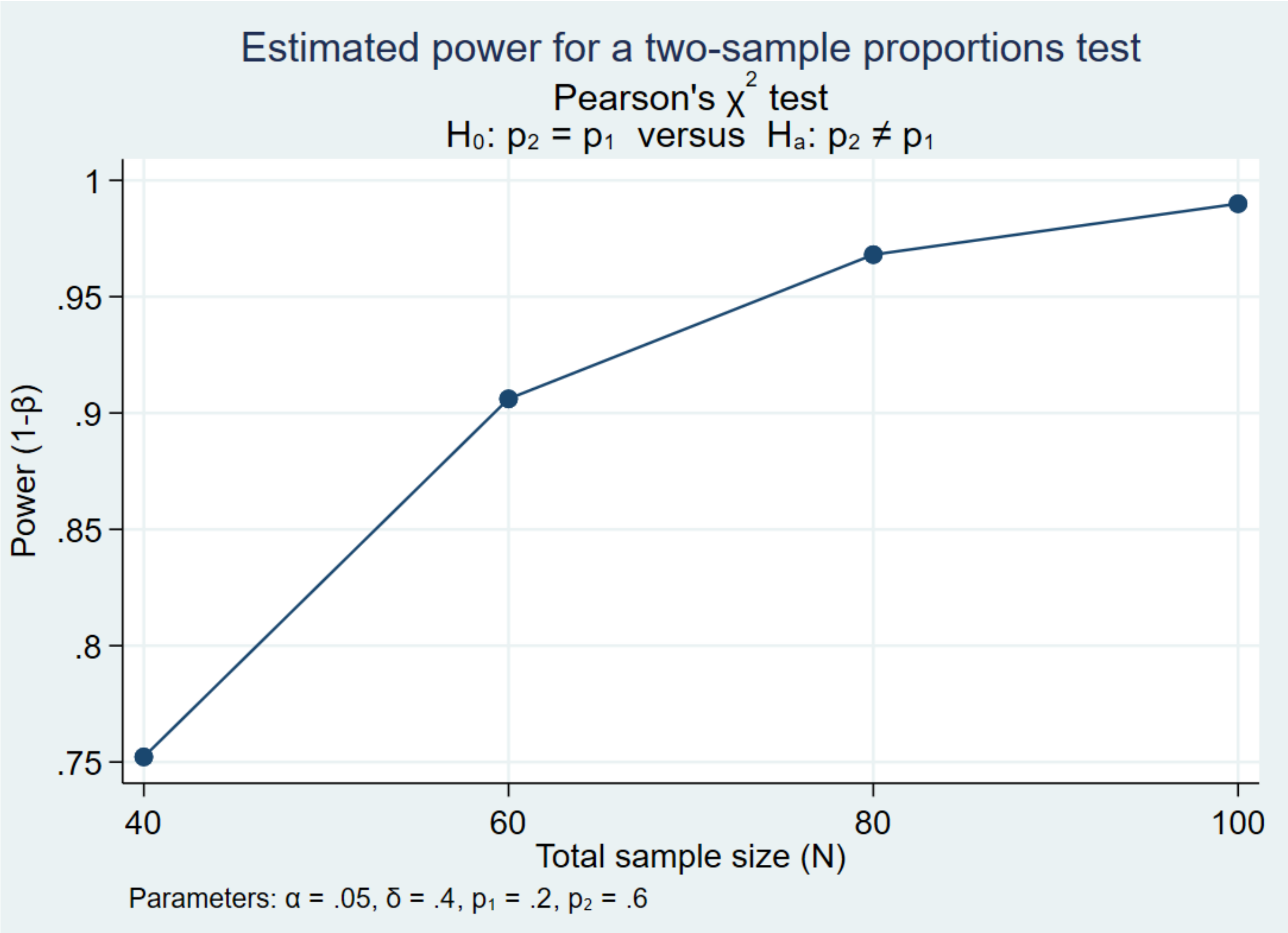
Where n_i is the sample size required in each group (i=1,2), α is the specific level of significance and $z_{1-\frac{\alpha}{2}}$ is the critical value corresponding to the significance level. $1 - \beta$ is the selected power and $z_{1-\beta}$ is the value from the standard normal distribution holding $1 - \beta$ below it. ES is the effect size, defined as follows:

$$ES = \frac{|p_1 = p_2|}{\sqrt{p(1-p)}}$$

where $|p_1 = p_2|$ is the absolute value of the proportions difference between the two groups holding under the alternative hypothesis, H_1, and p is the proportion by pooling the observations from the two comparison groups.

In Stata, we use the following code to calculate the sample size needed to reject the null hypothesis that $H_0 : p_1 = p_2$ ($H_1 : p_1 \neq p_2$) given $\hat{p_1} = 0.2, \hat{p_2} = 0.6$ on different fixed power levels:

```
power twoproportions 0.2 0.6, n(40(20)100) graph
```



1.2 Procedures to perform power analysis

1. Specify a hypothesis test.

Usually, there are several hypotheses in a research design, but for sample size calculation, make explicit a null and alternative hypothesis.

2. Specify the significance level of the test.

It is usually α = .05, but other values could be taken too.

3. Get the values of the parameters necessary to compute the power function.

To solve for sample size n, we need a value for standard deviation and other parameters. Need to note, sometimes we need to use a pilot dataset to get these values.

4. Specify the intended power of the test.

The power of a test is the probability of finding significance if the alternative hypothesis is true.

5. Calculate the needed sample size for a fixed power level.

2. Power Analysis with simulation

Nevertheless, formulas don't always work out to calculate the needed sample size such as in complex study designs. In these cases, simulation based power analysis stand out. The basic idea is to simulate running the experiment many times and calculate the proportion of times we reject the null hypothesis. This proportion provides an estimate of power. Generating a dataset and running an analysis for the hypothesis test is part of the simulation. One thing to mention is that randomness is usually introduced into the process through the dataset generation.

For example, say, the fixed power level is 95%, and you want to calculate the sample size on this level. You can take a “guess and check” method. With this method, firstly, you choose a sample size n_1 and run the simulation to estimate your power. If power is estimated to be lower than 95%, you need to select a new value n_2 that is larger than n_1 running the simulation again. Multiple procedures are repeated until the estimated power is roughly 95%.

As the example shows in the introduction part, for multiple commonly used statistical tests, we can use Stata’s power commands to calculate power and needed sample size. However, for complex models, such as multilevel or mixed effect models, we need to use simulations to calculate power and the needed sample size. In these scenarios, we usually use the following procedures to perform power analysis:

1.Write down the regression model of interest, including all parameters.

2.Specify the details of the covariates, such as the range of age or the proportion of females.

3.Locate or think about reasonable values for the parameters in your model.

4.Simulate a single dataset assuming the alternative hypothesis, and fit the model.

5.Write a program to create the datasets, fit the models, and use simulate to test the program.

6.Write a program to allows you to run your simulations with power.

7.Write a program that you can use numlists for all parameters.

3. Simulation-based Power Analysis in Stata

3.1 Simple linear regression

3.1.1 Write down the regression model of interest, including all parameters.

$$bpsystol = \beta_0 + \beta_1(age) + \beta_2(sex) + \beta_3(age * sex) + \epsilon$$

where the variables of interest are age, sex and the interaction of age and sex. Also, you need to estimate the coefficients for β_0, β_1, β_2, β_3.

3.1.2 Specify the details of the covariates.

You plan a study of systolic blood pressure (SBP) and you believe that there is an interaction between age and sex.

3.1.3 Locate or think about reasonable values for the parameters in your model.

```
webuse nhanes2

regress bpsystol c.age##ib1.sex
```

```
-----------------------
age             0.471***
              (28.121)

1.sex           0.000
                  (.)

2.sex         -20.458***
             (-17.557)

1.sex#c.~e      0.000
                  (.)

2.sex#c.~e      0.346***
              (15.008)

_cons         110.569***
             (130.995)
-----------------------
N               10351
r2              0.255
r2_a            0.255
F            1180.868
-----------------------
t statistics in parentheses
* p<0.1, ** p<0.05, *** p<0.01
```

Using the data data from the National Health and Nutrition Examination Survey (NHANES), we can estimate β_0 =110.6, β_1 =0.47, β_2 =-20.46, β_3 =0.35.

3.1.4 Simulate a single dataset assuming the alternative hypothesis, and fit the model.

Next, we create a simulated dataset based on our assumptions about the model under the alternative hypothesis.

```
clear
set seed 150
set obs 400
generate age = runiformint(18,65)
generate female = rbinomial(1,0.5)
generate interact = age*female
generate e = rnormal(0,20)
generate sbp = 110 + 0.5*age + (-20)*female + 0.35*interact  + e
```

We can then test the null hypothesis that the interaction term equals zero using a likelihood-ratio test.

```
regress sbp age i.female c.age#i.female
estimates store full
```

```
regress sbp age i.female
estimates store reduced

Likelihood-ratio test                                 LR chi2(1)  =      13.38
(Assumption: reduced nested in full)                  Prob > chi2 =     0.0003
```

The test yields a p-value of 0.0003.

```
return list

scalars:
r(p) =  .0002540647000293
r(chi2) =  13.38189649447986
r(df) =  1

local reject = (r(p)<0.05)
```

3.1.5 Write a program to create the datasets, fit the models, and use simulate to test the program.

Next, let's write a program that creates datasets under the alternative hypothesis.

```
capture program drop simregress
program simregress, rclass
    version 16
    // DEFINE THE INPUT PARAMETERS AND THEIR DEFAULT VALUES
    syntax, n(integer)          /// Sample size
          [ alpha(real 0.05)    /// Alpha level
            intercept(real 110) /// Intercept parameter
            age(real 0.5)       /// Age parameter
            female(real -20)    /// Female parameter
            interact(real 0.35) /// Interaction parameter
            esd(real 20) ]      //  Standard deviation of the error
    quietly {
        // GENERATE THE RANDOM DATA
        clear
        set obs `n'
        generate age = runiformint(18,65)
        generate female = rbinomial(1,0.5)
        generate interact = age*female
        generate e = rnormal(0,`esd')
        generate sbp = `intercept' + `age'*age + `female'*female + ///
           `interact'*interact  + e
        // TEST THE NULL HYPOTHESIS
        regress sbp age i.female c.age#i.female
        estimates store full
        regress sbp age i.female
        estimates store reduced
        lrtest full reduced
```

```
    }
    // RETURN RESULTS
    return scalar reject = (r(p)<`alpha')
end
```

Below, we use simulate to run simregress 200 times and summarize the variable reject. The results indicate that we would have 74% power to detect an interaction parameter of 0.35 given a sample of 400 participants and the other assumptions about the model.

```
Simulations (200)
----+--- 1 ---+--- 2 ---+--- 3 ---+--- 4 ---+--- 5
..................................................    50
..................................................   100
..................................................   150
..................................................   200

summarize reject

    Variable |        Obs        Mean    Std. dev.       Min        Max
-------------+---------------------------------------------------------
      reject |        200        .735    .4424407          0          1
```

3.1.6 Write a program called power_cmd_simregress, which allows you to run your simulations with power.

Next, let's write a program called power_cmd_simregress so that we can integrate simregress into Stata's power command.

```
capture program drop power_cmd_simregress
program power_cmd_simregress, rclass
    version 17
    // DEFINE THE INPUT PARAMETERS AND THEIR DEFAULT VALUES
    syntax, n(integer)          /// Sample size
          [ alpha(real 0.05)    /// Alpha level
            intercept(real 110) /// Intercept parameter
            age(real 0.5)       /// Age parameter
            female(real -20)    /// Female parameter
            interact(real 0.35) /// Interaction parameter
            esd(real 20)        /// Standard deviation of the error
            reps(integer 100)]  //   Number of repetitions

    // GENERATE THE RANDOM DATA AND TEST THE NULL HYPOTHESIS
    quietly {
        simulate reject=r(reject), reps(`reps'):              ///
             simregress, n(`n') age(`age') female(`female')    ///
                         interact(`interact') esd(`esd') alpha(`alpha')
```

```
        summarize reject
    }
    // RETURN RESULTS
    return scalar power = r(mean)
    return scalar N = `n'
    return scalar alpha = `alpha'
    return scalar intercept = `intercept'
    return scalar age = `age'
    return scalar female = `female'
    return scalar interact = `interact'
    return scalar esd = `esd'
end
```

3.1.7 Write a program called power_cmd_simregress_init.

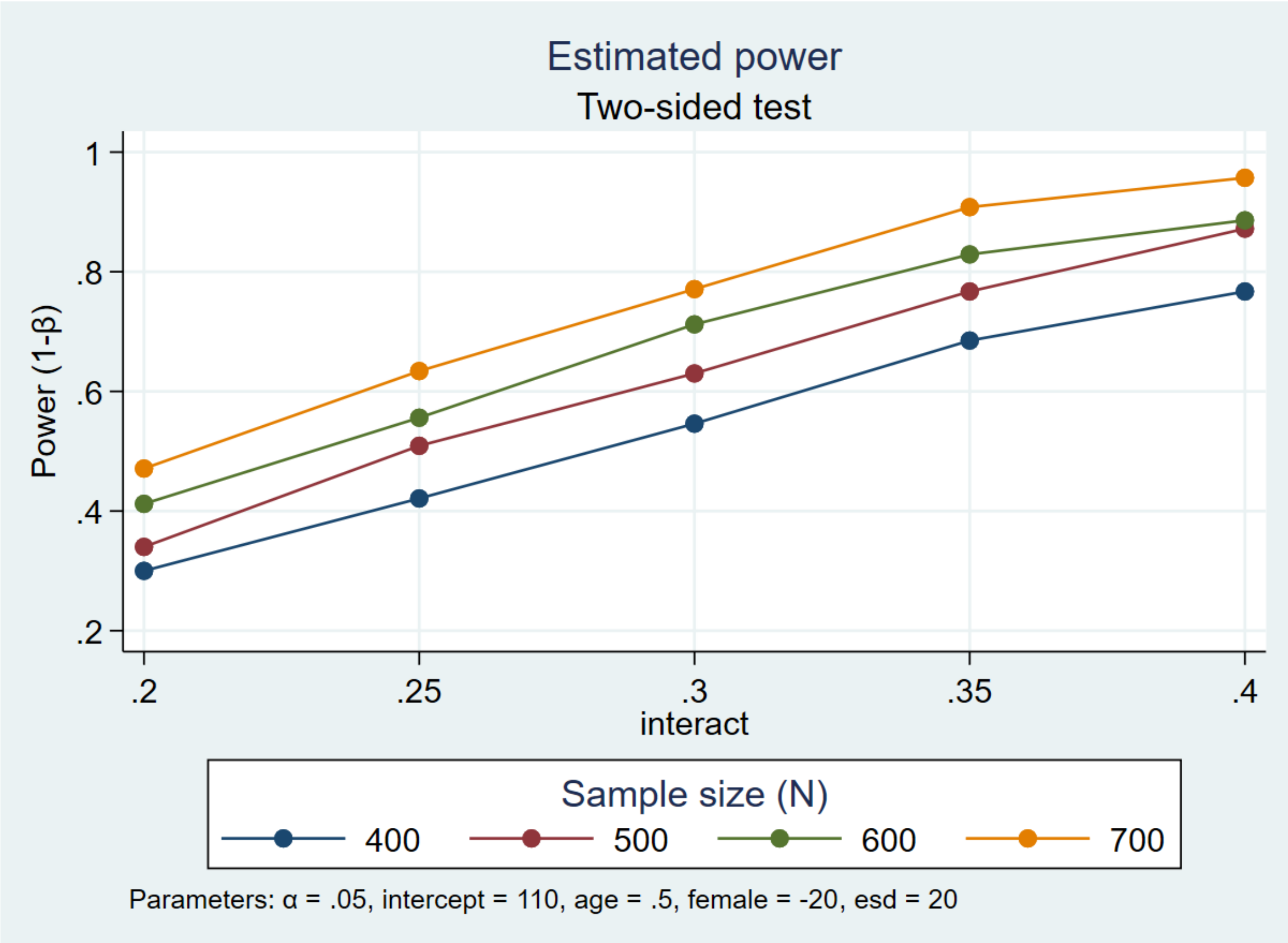
Run power simregress for a range of input parameter values, including the parameters listed in double quotes.

```
capture program drop power_cmd_simregress_init
program power_cmd_simregress_init, sclass
    sreturn local pss_colnames "intercept age female interact esd"
    sreturn local pss_numopts  "intercept age female interact esd"
end
```

Now, we're ready to use power simregress! The output below shows the simulated power when the interaction parameter equals 0.2 to 0.4 in increments of 0.05 for samples of size 400, 500, 600, and 700.

```
power simregress, n(400(100)700) intercept(110)                 ///
                  age(0.5) female(-20) interact(0.2(0.05)0.4)    ///
                  reps(1000) table graph(xdimension(interact)    ///
                  legend(rows(1)))

Estimated power
Two-sided test


  +--------------------------------------------------------------------+
  |   alpha   power       N intercept     age  female interact      esd |
  |--------------------------------------------------------------------|
  |     .05      .3     400       110      .5     -20       .2       20 |
  |     .05    .421     400       110      .5     -20      .25       20 |
  |     .05    .546     400       110      .5     -20       .3       20 |
  |     .05    .685     400       110      .5     -20      .35       20 |
  |     .05    .767     400       110      .5     -20       .4       20 |
  |     .05     .34     500       110      .5     -20       .2       20 |
  |     .05    .509     500       110      .5     -20      .25       20 |
  |     .05     .63     500       110      .5     -20       .3       20 |
  |     .05    .767     500       110      .5     -20      .35       20 |
  |     .05    .872     500       110      .5     -20       .4       20 |
  |     .05    .412     600       110      .5     -20       .2       20 |
```

```
  |       .05      .556      600        110        .5      -20        .25        20 |
  |       .05      .712      600        110        .5      -20         .3        20 |
  |       .05      .829      600        110        .5      -20        .35        20 |
  |       .05      .886      600        110        .5      -20         .4        20 |
  |       .05      .471      700        110        .5      -20         .2        20 |
  |       .05      .634      700        110        .5      -20        .25        20 |
  |       .05      .771      700        110        .5      -20         .3        20 |
  |       .05      .908      700        110        .5      -20        .35        20 |
  |       .05      .957      700        110        .5      -20         .4        20 |
  +---------------------------------------------------------------------------------+
```



3.2 Mixed Effect models

3.2.1 Write down the regression model of interest, including all parameters.

$$weight_{it} = \beta_0 + \beta_1(age_{it}) + \beta_2(female_i) + \beta_3(age_{it} * female_i) + \mu_{0i} + \mu_{1i}(age) + \epsilon_{it}$$

where i stands for children, t for age, and we assume $\mu_{0i} \sim N(0, \tau_0)$, $\mu_{1i} \sim N(0, \tau_1)$, $\epsilon_{it} \sim N(0, \sigma)$.

The covariates are weight, age, female, and the interaction term age*female. Also, we need to estimate the coefficients for β_0 (Intercept), β_1 (Coefficient for age), β_2 (Coefficient for the female comparing with the male), β_3 (Coefficient for the interaction),μ_{1i} (randome effect of age).

We also need to think about the covariates in our model. This is a longitudinal study, so we need to specify the starting age, the length of time between measurements, and the total number of measurements. We also need to consider the proportion of males and females in our study. Are we likely to sample 50% females and 50% males?

Let's assume that we will measure the childrens' weight every 6 months for 2 years beginning at age 12. And let's also assume that the sample will be 50% female. The interaction term age×female is easy to calculate once we create variables for age and female.

3.2.2 Specify the details of the covariates, such as the range of age or the proportion of females.

Let's assume that we will measure the children's weight every 4 months for 4 years beginning at age 10. Also, in the sample, the proportion of female is equal to that of male. It's not difficult to calculate the iteration term when generate the variable for age and female.

3.2.3 Locate or think about reasonable values for the parameters in your model.

In this step, we use an external data set measuring Asian kid's data to estimate the coefficients for the above regression model, and we get β_0=5.35, β_1=3.59, β_2=-0.47, β_3=-0.24,τ_0=0.24,τ_1 -0.57and σ=1.17.

```
mixed weight c.age##i.girl || id: age, stddev
-----------------------
weight
age              3.588***
              (18.963)

0.girl           0.000
                   (.)

1.girl          -0.468
              (-1.585)

0.girl#c~e       0.000
                   (.)

1.girl#c~e      -0.240
              (-0.896)

_cons            5.351***
              (25.770)
-----------------------
lns1_1_1
_cons           -0.569***
              (-2.640)
-----------------------
lns1_1_2
_cons           -1.434
              (-0.847)
-----------------------
lnsig_e
_cons            0.153**
```

```
              (2.343)
-----------------------
N                  198
r2
r2_a
F
-----------------------
t statistics in parentheses
* p<0.1, ** p<0.05, *** p<0.01
```

3.2.4 Simulate a single dataset assuming the alternative hypothesis, and fit the model.

Next, we create a simulated dataset based on our assumptions about the model under the alternative hypothesis. We will simulate 5 observations at 4-month increments for 200 children.

```
set seed 16
clear
set obs 200
generate child = _n
generate female = rbinomial(1,0.5)
generate u_0i = rnormal(0,0.25)
generate u_1i = rnormal(0,0.60)
expand 5
bysort child: generate age = (_n-1)*0.5
generate interaction = age*female
generate e_ij = rnormal(0,1.2)
generate weight = 5.35 + 3.6*age + (-0.5)*female + (-0.25)*interaction ///
    + u_0i + age*u_1i + e_ij
```

Our dataset includes the random deviations that we would not observe in a real dataset. We can then use mixed to fit a model to our simulated data.

```
mixed weight age i.female c.age#i.female || child: age , stddev nolog noheader
estimates store full
mixed weight age i.female || child: age , stddev nolog noheader
estimates store reduced
lrtest full reduced
```

We can then test the null hypothesis that the interaction term equals zero using a likelihood-ratio test.

```
 lrtest full reduced

Likelihood-ratio test                                 LR chi2(1)  =      8.23
(Assumption: reduced nested in full)                  Prob > chi2 =    0.0041
```

The p-value for our test is 0.0041, so we would reject the null hypothesis that the interaction term equals zero.

3.2.5 Write a program to create the datasets, fit the models, and use simulate to test the program.

Next, let's write a program that creates datasets under the alternative hypothesis, fits mixed models, tests the null hypothesis of interest, and uses simulate to run many iterations of the program.

```
capture program drop simmixed
program simmixed, rclass
    version 16
    // PARSE INPUT
    syntax, n1(integer)              ///
            n(integer)               ///
          [ alpha(real 0.05)         ///
            intercept(real 5.35)     ///
            age(real 3.6)            ///
            female(real -0.5)        ///
            interact(real -0.25)     ///
            u0i(real 0.25)           ///
            u1i(real 0.60)           ///
            eij(real 1.2) ]

    // COMPUTE POWER
    quietly {
        drop _all
        set obs `n'
        generate child = _n
        generate female = rbinomial(1,0.5)
        generate u_0i = rnormal(0,`u0i')
        generate u_1i = rnormal(0,`u1i')
        expand `n1'
        bysort child: generate age = (_n-1)*0.5
        generate interaction = age*female
        generate e_ij = rnormal(0,`eij')
        generate weight = `intercept' + `age'*age + `female'*female + ///
           `interact'*interaction  + u_0i + age*u_1i + e_ij

        mixed weight age i.female c.age#i.female || child: age, iter(200)
        local conv1 = e(converged)
        estimates store full
        mixed weight age i.female || child: age, iter(200)
        local conv2 = e(converged)
        estimates store reduced
        lrtest full reduced
        local reject = cond(`conv1' + `conv2'==2, (r(p)<`alpha'), .)
    }
    // RETURN RESULTS
    return scalar reject = `reject'
    return scalar conv = `conv1'+`conv2'
end
```

We then use simulate to run simmixed 10 times using the default parameter values for 5 observations on each of 200 children.

```
simulate reject=r(reject) converged=r(conv), reps(10) seed(12345):
             simmixed, n1(5) n(200)

      command:  simmixed, n1(5) n(200)
       reject:  r(reject)
    converged:  r(conv)

Simulations (10)
----+--- 1 ---+--- 2 ---+--- 3 ---+--- 4 ---+--- 5
```

simulate saved the results of the hypothesis tests to a variable named reject. The mean of reject is our estimate of the power to test the null hypothesis that the age×sex interaction term equals zero, assuming that the weight of 200 children is measured 5 times.

3.2.6 Write a program called power_cmd_mymethod, which allows you to run your simulations with power.

We could stop with our quick simulation if we were interested only in a specific set of assumptions. But it's easy to write an additional program named power_cmd_simmixed that will allow us to use Stata's power command to create tables and graphs for a range of sample sizes.

```
capture program drop power_cmd_simmixed
program power_cmd_simmixed, rclass
    version 16
    // PARSE INPUT
    syntax, n1(integer)             ///
            n(integer)              ///
          [ alpha(real 0.05)        ///
            intercept(real 5.35)    ///
            age(real 3.6)           ///
            female(real -0.5)       ///
            interact(real -0.25)    ///
            u0i(real 0.25)          ///
            u1i(real 0.60)          ///
            eij(real 1.2)           ///
            reps(integer 1000) ]

    // COMPUTE POWER
    quietly {
        simulate reject=r(reject), reps(`reps'):                        ///
        simmixed, n1(`n1') n(`n') alpha(`alpha') intercept(`intercept')     ///
                  age(`age') female(`female') interact(`interact')        ///
                  u0i(`u0i') u1i(`u1i') eij(`eij')
        summarize reject
    }
```

```
    // RETURN RESULTS
    return scalar power = r(mean)
    return scalar n1 = `n1'
    return scalar N = `n'
    return scalar alpha = `alpha'
    return scalar intercept = `intercept'
    return scalar age = `age'
    return scalar female = `female'
    return scalar interact = `interact'
    return scalar u0i = `u0i'
    return scalar u1i = `u1i'
    return scalar eij = `eij'
end
```

3.2.7 Write a program called power_cmd_mymethod_init so that you can use numlists for all parameters.

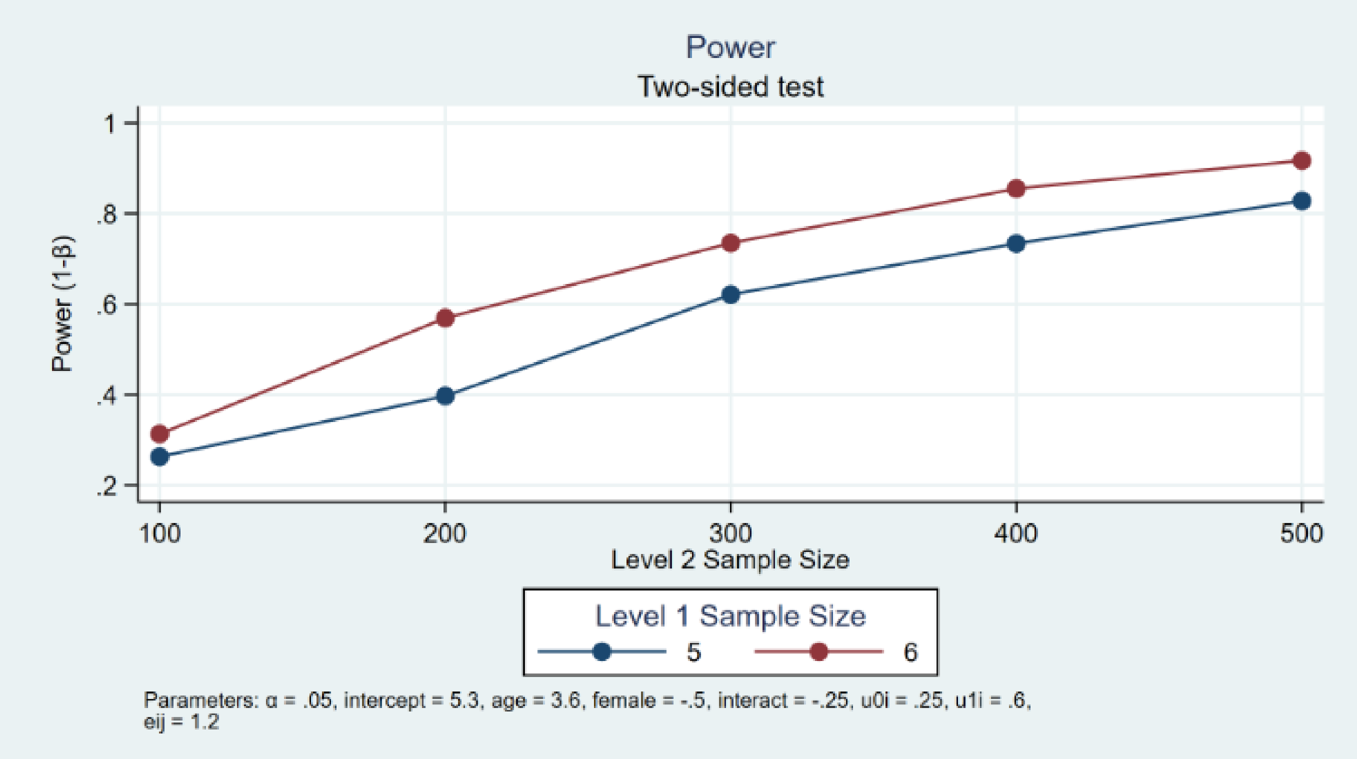
It's also easy to write a program named power_cmd_simmixed_init that will allow us to simulate power for a range of values for the parameters in our model.

```
capture program drop power_cmd_simmixed_init
program power_cmd_simmixed_init, sclass
      version 16
      sreturn clear
      // ADD COLUMNS TO THE OUTPUT TABLE
      sreturn local pss_colnames "n1 intercept age female interact u0i u1i eij"
      // ALLOW NUMLISTS FOR ALL PARAMETERS
      sreturn local pss_numopts  "n1 intercept age female interact u0i u1i eij"
end
```

Now, we can use power simmixed to simulate power for a variety of assumptions. The example below simulates power for a range of sample sizes at both levels 1 and 2. Level 2 sample sizes range from 100 to 500 children in increments of 100. At level 1, we consider 5 and 6 observations per child.

```
power simmixed, n1(5 6) n(100(100)500) reps(1000)
               table(n1 N power)
               graph(ydimension(power) xdimension(N) plotdimension(n1)
               xtitle(Level 2 Sample Size) legend(title(Level 1 Sample Size))
xxxxxxxxxxxxxxxxxxxxxxxxxxx
Estimated power
Two-sided test

  +-------------------------+
  |      n1        N   power |
  |-------------------------|
  |       5      100   .2629 |
  |       6      100    .313 |
  |       5      200    .397 |
```

```
|          6       200      .569 |
|          5       300      .621 |
|          6       300      .735 |
|          5       400      .734 |
|          6       400      .855 |
|          5       500      .828 |
|          6       500      .917 |
+-------------------------------+
```



4. Simulatiion-based Power Analysis in Python

After elaborating how to implement simulation based power analysis in Stata, let's move to the codes in Python.

4.1 Linear Model Simulation

4.1.1 Import Libraries

First, let's import the libraries used for simulation based power analysis in Python which include numpy, pandas, LinearRegression, statsmodels.api, scipy and matplotlib.pyplot.

```
import random
import numpy as np
import pandas as pd

from sklearn.linear_model import LinearRegression
import statsmodels.api as sm
import scipy
```

```
import matplotlib.pyplot as plt
```

4.1.2 Setting Seeds

Set the seed to 1024.

```
np.random.seed(1024)
```

4.1.3 Setting Parameters

Here, we use the same regression model as the example from Stata.

```
def generate_dataset(sample_size, interact_coef):

    data_set = []

    for i in range(sample_size):

        _id = i
        age = np.random.randint(18,66)
        female = np.random.choice([0, 1])
        interact = age * female
        e = np.random.normal(0, 20)

        sbp = 110 + 0.5*age + (-20)*female + interact_coef*interact + e

        data_set.append([_id, age, female, interact, e, sbp])

    data_set = pd.DataFrame(data_set)
    data_set.columns = ["_id", "age", "female", "interact", 'e', "sbp"]

    return data_set
```

4.1.4 Simulation

Following the previous step, we run simulations in Python.

```
def cal_power(sample_size, interact_coef, simiu_cnt, alpha):

    power_list = []

    for i in range(simiu_cnt):

        dataset = generate_dataset(sample_size, interact_coef)
```

```
        y1 = dataset['sbp']
        x1 = dataset[['age', 'female', 'interact']]
        x1 = sm.add_constant(x1)
        full_model = sm.OLS(y1, x1).fit()
        full_ll = full_model.llf

        y2 = dataset['sbp']
        x2 = dataset[['age', 'female']]
        x2 = sm.add_constant(x2)
        reduced_model = sm.OLS(y2, x2).fit()
        reduced_ll = reduced_model.llf

        LR_statistic = -2*(reduced_ll-full_ll)
        power = scipy.stats.chi2.sf(LR_statistic, 1)

        if power<=alpha:
            power_list.append(1)
        else:
            power_list.append(0)

    mean_power = sum(power_list)/len(power_list)

    return [sample_size, interact_coef, mean_power]
```

4.1.5 Results

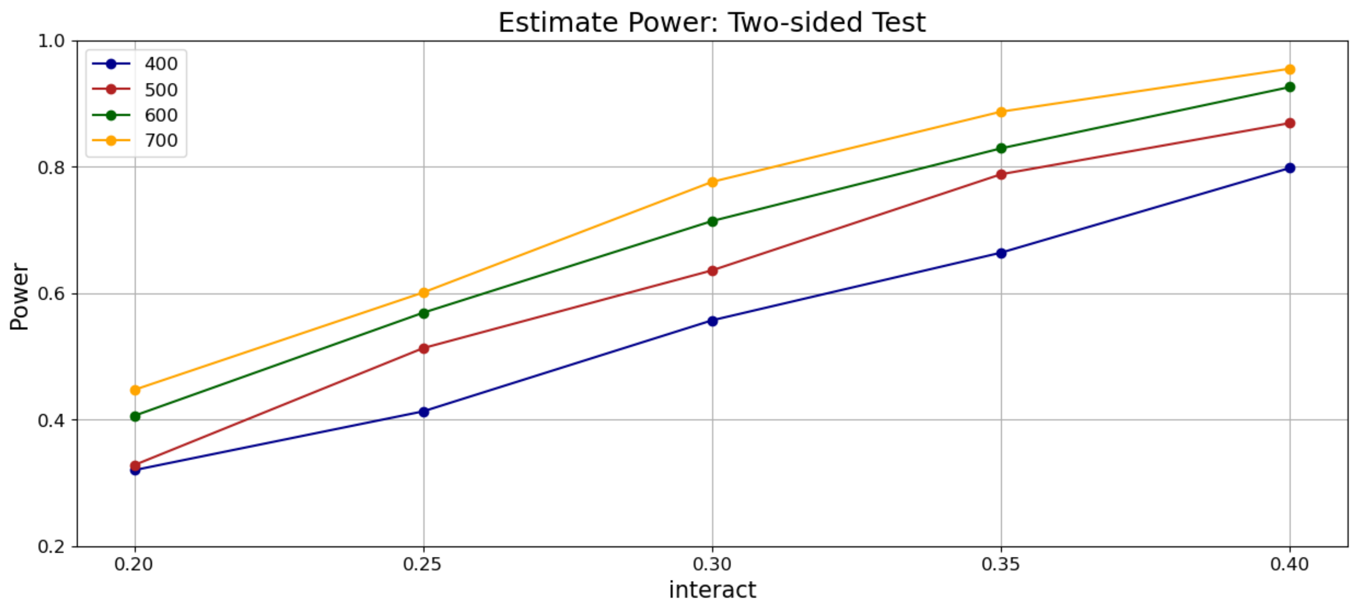
In this part, we export the results of the simulations which include two parts: a table and a graph showing the results from the simulations. It should be noted that the graph from Python simulation is a little bit different from that in Stata, and this is mainly caused by different simulation process within Stata and Python.

```
result = []

for i in range(400, 800, 100):
    for j in [0.2, 0.25, 0.3, 0.35, 0.4]:
        result.append(cal_power(sample_size = i, interact_coef = j, simiu_cnt = 
1000, alpha = 0.05))

result = pd.DataFrame(result)
result.columns = ['N', 'interact_coef', 'Power']
result
```

```
    N   interact_coef   Power
0   400 0.20     0.320
1   400 0.25     0.413
2   400 0.30     0.557
3   400 0.35     0.664
4   400 0.40     0.798
5   500 0.20     0.328
```

```
6   500 0.25    0.513
7   500 0.30    0.636
8   500 0.35    0.788
9   500 0.40    0.869
10  600 0.20    0.406
11  600 0.25    0.569
12  600 0.30    0.714
13  600 0.35    0.829
14  600 0.40    0.926
15  700 0.20    0.447
16  700 0.25    0.601
17  700 0.30    0.776
18  700 0.35    0.887
19  700 0.40    0.955
```

```
n_list = result['N'].unique()
color_list = ['darkblue', 'firebrick', 'darkgreen', 'orange']

plt.figure(figsize=(15,6))

for i in range(len(n_list)):
    n = n_list[i]
    c = color_list[i]
    plt.plot(result[result['N']==n]['interact_coef'], result[result['N']==n]
['Power'], 'o-', color = c)

plt.grid()

plt.xticks([0.2, 0.25, 0.3, 0.35, 0.4], fontsize = 12)
plt.yticks([0.2, 0.4, 0.6, 0.8, 1], fontsize = 12)

plt.xlabel('interact', fontsize = 15)
plt.ylabel('Power', fontsize = 15)

plt.legend(result['N'].unique(), fontsize = 12)
plt.title('Estimate Power: Two-sided Test', fontsize = 18)

plt.show()
```




```

```

4.2 Mixed Model Simulation

4.2.1 Import Libraires

We import the same libraries as in section 4.1.

```
import random
import numpy as np
import pandas as pd

from sklearn.linear_model import LinearRegression
import statsmodels.api as sm
import scipy

import matplotlib.pyplot as plt
```

4.2.2 Setting Seeds

Again, we set seed to 1024.

```
np.random.seed(1024)
```

4.2.3 Setting Parameters

We set a fixed effect model here too that is similar to 3.2.1. Please refer to that section for the interpretation of the parameters.

```
def generate_dataset(sample_size, obser_cnt):

    data_set = []

    for i in range(sample_size):
        child_id = i
        female_origin = np.random.choice([0, 1])
        u_0i_origin = np.random.normal(0, 0.25)
        u_1i_origin = np.random.normal(0, 0.60)

        for j in range(obser_cnt):

            child = child_id
            female = female_origin
            age = 0.5*j
            u_0i = u_0i_origin
            u_1i = u_1i_origin
            interaction = age * female
            e_ij = np.random.normal(0, 1.2)
            weight = 5.35 + 3.6*age + (-0.5)*female + (-0.25)*interaction + u_0i +
age*u_1i + e_ij

            data_set.append([child, female, age, u_0i, u_1i, interaction, e_ij,
weight])

    data_set = pd.DataFrame(data_set)
    data_set.columns = ["child_id", "female", "age", "u_0i", "u_li",
"interaction", "e_ij", "weight"]

    return data_set
```

4.2.4 Simulation

Run simulations.

```
def cal_power(sample_size, obser_cnt, simiu_cnt, alpha):

    power_list = []

    for i in range(simiu_cnt):

        dataset = generate_dataset(sample_size, obser_cnt)

        y1 = dataset['weight']
        x1 = dataset[['female', 'age', 'interaction']]
        x1 = sm.add_constant(x1)
```

```
        full_model = sm.OLS(y1, x1).fit()
        full_ll = full_model.llf

        y2 = dataset['weight']
        x2 = dataset[['female', 'age']]
        x2 = sm.add_constant(x2)
        reduced_model = sm.OLS(y2, x2).fit()
        reduced_ll = reduced_model.llf

        LR_statistic = -2*(reduced_ll-full_ll)
        power = scipy.stats.chi2.sf(LR_statistic, 1)

        if power<=alpha:
            power_list.append(1)
        else:
            power_list.append(0)

    mean_power = sum(power_list)/len(power_list)

    return [obser_cnt, sample_size, mean_power]
```

4.2.5 Results

The last precedure is to export the results which contain a table and a graph.

```
result = []

for i in range(100, 600, 100):
    for j in range(5, 7):
        result.append(cal_power(sample_size = i, obser_cnt = j, simiu_cnt = 1000,
alpha = 0.05))

result = pd.DataFrame(result)
result.columns = ['n1', 'N', 'Power']
result
```

```
n1  N   Power
0   5   100 0.290
1   6   100 0.398
2   5   200 0.491
3   6   200 0.632
4   5   300 0.655
5   6   300 0.798
6   5   400 0.779
7   6   400 0.917
8   5   500 0.857
```

```
9   6   500 0.940
```

```
n1_list = result['n1'].unique()
color_list = ['darkblue', 'firebrick', ]

plt.figure(figsize=(15,6))

for i in range(len(n1_list)):
    n = n1_list[i]
    c = color_list[i]
    plt.plot(result[result['n1']==n]['N'], result[result['n1']==n]['Power'], '-o',
color = c)

plt.grid()

plt.xticks([100, 200, 300, 400, 500], fontsize = 12)
plt.yticks([0.2, 0.4, 0.6, 0.8, 1], fontsize = 12)

plt.xlabel('Level 2 Sample Size', fontsize = 15)
plt.ylabel('Power', fontsize = 15)

plt.legend(result['n1'].unique(), fontsize = 12)
plt.title('Power: Two-sided Test', fontsize = 18)

plt.show()
```

5. Simulatiion-based Power Analysis in R

After introducing simulation based power analysis in Stata and Python, we introduce how to do the same simulation in R.

5.1 Linear Model Simulation

5.1.1 Import Library

We mainly use lmtest and ggplot2 libraries in R.

```
library(lmtest)
library(ggplot2)
```

5.1.2 Setting Seeds

We set seed to 1024 as that in Python.

```
set.seed(1024)
```

5.1.3 Simulation

We run simulation based on a regression model exactly as that in Stata and Python simulations.

```
sample_cnt <- c(400, 500, 600, 700)
interact_coef <- c(0.2, 0.25, 0.3, 0.35, 0.4)
repect_cnt <- seq(1, 1000)

power_list <- data.frame(sample_cnt=double(), interact_coef=double(),
power=double())[-1,]

for (s in sample_cnt) {

  for (i in interact_coef){

    results <- c()

    for (r in repect_cnt){

      age = ceiling(runif(s, 18, 65))
      female = rbinom(s, 1, 0.5)
      interact = age * female
      e = rnorm(s, 0, 20)
      sbp = 110 + 0.5*age + (-20)*female + i*interact + e

      full_model = lm(sbp ~ age + female + interact)
      reduced_model = lm(sbp ~ age + female)
      prob = lrtest(full_model, reduced_model)$Pr[2]
      reject = ifelse((prob<=0.05), 1, 0)

      results <- rbind(results, reject)
```

```
    }

    power_list = rbind(power_list, data.frame(sample_cnt=s, interact_coef=i,
power=mean(results)))

  }
}

power_list
```

5.1.4 Output

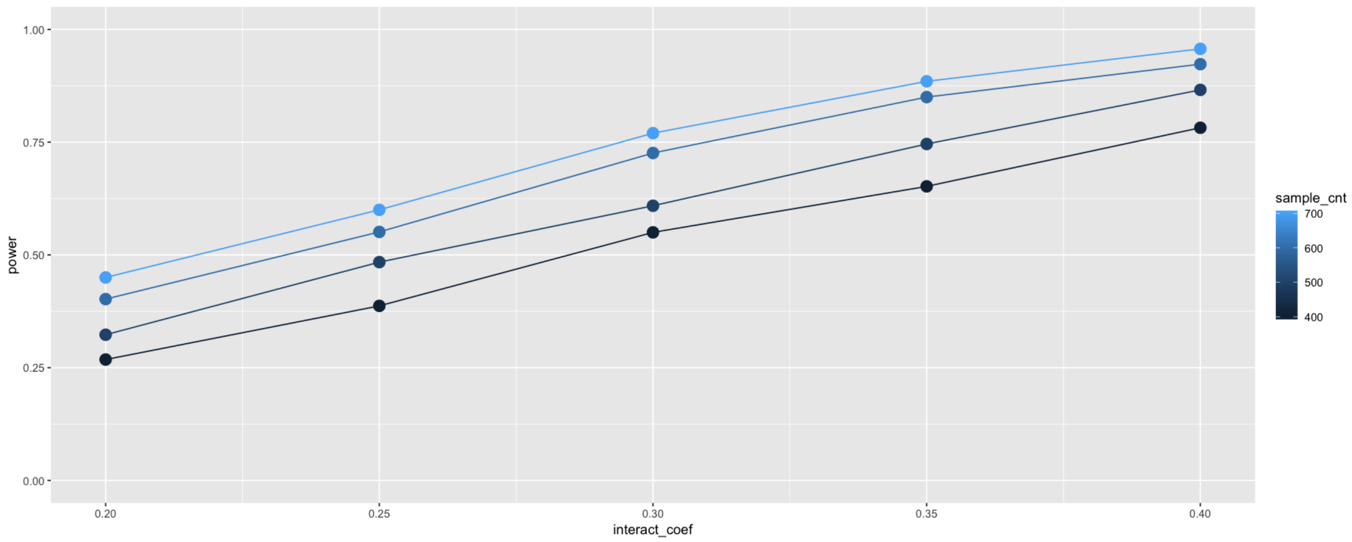
Here's the table from the simulation.

```
sample_cnt  interact_coef   power
400 0.20    0.268
400 0.25    0.387
400 0.30    0.550
400 0.35    0.652
400 0.40    0.782
500 0.20    0.323
500 0.25    0.484
500 0.30    0.609
500 0.35    0.746
500 0.40    0.866
600 0.20    0.402
600 0.25    0.551
600 0.30    0.726
600 0.35    0.850
600 0.40    0.923
700 0.20    0.450
700 0.25    0.600
700 0.30    0.770
700 0.35    0.885
700 0.40    0.957
```

5.1.5 Graph

Here's the graph from the simulation.

```
options(repr.plot.width = 15, repr.plot.height = 6)
ggplot(power_list, aes(interact_coef, power, group = sample_cnt, color =
sample_cnt)) + geom_line(aes(colour = sample_cnt)) + geom_point(size = 4) +
ylim(0,1)
```



5.2 Mixed Model Simulation

5.2.1 Import Library

In mixed model simulation in R, we still use two libraries: lmtest and ggplot2.

```
library(lmtest)
library(ggplot2)
```

5.2.2 Setting Seeds

Set the seed to 1024.

```
set.seed(1024)
```

5.2.3 Simulation

Run simualation of fixed effect model in R.

```
sample_cnt <- c(100, 200, 300, 400, 500)
obser_cnt <- c(5, 6)
repect_cnt <- seq(1, 1000)

power_list <- data.frame(obser=integer(), sample=integer(), power=double())

for (s in sample_cnt){

  for (o in obser_cnt){

    results <- c()

    for (r in repect_cnt){
```

```
      child <- seq(1, s)
      female = rbinom(s, 1, 0.5)
      u_0i = rnorm(s, 0, 0.25)
      u_1i = rnorm(s, 0, 0.60)
      data_set <- data.frame(child = child, female = female, u_0i = u_0i, u_1i =
u_1i)

      data_set_expand <- data_set[rep(seq(nrow(data_set)), o), 1:4]
      age <- c()
      for (obs in seq(0, (o-1)*0.5, 0.5)){

        age = c(age, rep(obs, s))

      }
      data_set_expand = cbind(data_set_expand, age)
      e_ij = rnorm(s*o, 0, 1.2)
      data_set_expand = cbind(data_set_expand, e_ij)
      data_set_expand$interact = data_set_expand$age * data_set_expand$female
      data_set_expand$weight = 5.35 + 3.6*data_set_expand$age +
(-0.5)*data_set_expand$female + (-0.25)*data_set_expand$interact +
data_set_expand$u_0i + data_set_expand$age*data_set_expand$u_1i +
data_set_expand$e_ij

      full_model = lm(data_set_expand$weight ~ data_set_expand$age +
data_set_expand$female + data_set_expand$interact)
      reduced_model = lm(data_set_expand$weight ~ data_set_expand$age +
data_set_expand$female)
      prob = lrtest(full_model, reduced_model)$Pr[2]
      reject = ifelse((prob<=0.05), 1, 0)
      results <- rbind(results, reject)
    }

    power_list = rbind(power_list, data.frame(obser=o, sample=s,
power=mean(results)))

  }
}

power_list
```

5.2.4 Output

THe table result for the mixed effect model in R.

```
obser   sample  power
5   100 0.323
6   100 0.401
5   200 0.505
6   200 0.651
5   300 0.680
```

```
6   300 0.815
5   400 0.780
6   400 0.897
5   500 0.845
6   500 0.951
```

5.2.5 Graph

The graph of the simulation result for fixed effect model in R.

```
options(repr.plot.width = 15, repr.plot.height = 6)
ggplot(power_list, aes(sample, power, group = obser, color = obser)) +
geom_line(aes(colour = obser)) + geom_point(size = 4) + ylim(0,1)
```