SI528 – Biostatistics: Project -1 Group—C

Submitted By

Subhojit kayal (205280022)

Amit Verma (205280016)

Aman Patwa (205280020)

Anamika Mishra (205280019)

Sarthak Agrawal (205280017)

Aman kumar (205280021)

Rajat kumar Singh (205280018)

Under the Guidance of

Prof. Kalyan Das

Department of Mathematics

IIT Bombay

Q(1) (i) :

Code

```
#Dataset is generated with population size 2400
import numpy as np
import pandas as pd
import random
import seaborn as sns
import matplotlib.pyplot as plt
i = np.repeat(np.array([1,2,3,4]),600)
j = np.repeat(np.array([1,2,3]),200)
j = np.array(list(j)*4)
x = np.array([20,40,70]*800)
x = np.array(x)
b0 = np.array([-1]*2400)
b1 = np.array([0.2]*2400)
ai = np.random.normal(0,1.5,2400)
bi = np.random.normal(0,2,2400)
eij = np.random.normal(0,1,2400)
b1x = b1*x
bix = bi*x
yij = b0 + b1x + bix + ai + eij
data = pd.DataFrame(np.hstack((yij[:,None],i[:,None],j[:,None],x[:,None])),columns=['Yij','i','j','X'])

# LMM is modelled and fit
import statsmodels.api as sm
import statsmodels.formula.api as smf
import warnings
import math
warnings.filterwarnings("ignore")
n = len(data)
model = smf.mixedlm("Yij ~ X", data, groups=data["i"], re_formula="~X")
```

```
model_fit = model.fit(method=["lbfgs"])

print(model_fit.summary())

fixed_effects = model_fit.bse_fe

random_effects = model_fit.bse_re

residuals = model_fit.resid

random_effect_variance
=((random_effects[0]**2)+(random_effects[1]**2)+(random_effects[2]**2))

residual_variance = [x**2 for x in residuals]

residual_variance = sum(residual_variance)/(n-2)

print('\n\n' ,random_effect_variance,'\n',residual_variance,'\n\n')
```

The table of estimates are :

```
     Mixed Linear Model Regression Results
============================================================
Model:            MixedLM Dependent Variable: Yij
No. Observations: 2400    Method:             REML
No. Groups:       4       Scale:              9386.4329
Min. group size:  600     Log-Likelihood:     -14381.5632
Max. group size:  600     Converged:          No
Mean group size:  600.0
------------------------------------------------------------
               Coef.  Std.Err.   z   P>|z| [0.025 0.975]
------------------------------------------------------------
Intercept       3.899    4.657 0.837 0.403 -5.229 13.026
X               0.075    0.107 0.698 0.485 -0.135  0.285
Group Var       1.532    0.668
Group x X Cov  -0.116    0.024
X Var           0.009    0.001
============================================================
```

```
random_effect_variance = 0.4469984096008471
residual_variance = 9381.021850456276
```

Q1 (ii) here we are going to use the same model as in que 1(i) and just simulating model for 1000 times.

Code for "Running 1000 simulations for the same model with population size 2400"

```
iter = 0

u , v = [] , []

est_0 , est_1 = [] , []

while iter != 1000:
```

```
i = np.repeat(np.array([1,2,3,4]),600)
j = np.repeat(np.array([1,2,3]),200)
j = np.array(list(j)*4)
x = np.array([20,40,70]*800)
x = np.array(x)
b0 = np.array([-1]*2400)
b1 = np.array([0.2]*2400)
ai = np.random.normal(0,1.5,2400)
bi = np.random.normal(0,2,2400)
eij = np.random.normal(0,1,2400)
b1x = b1*x
bix = bi*x
yij = b0 + b1x + bix + ai + eij
data = pd.DataFrame(np.hstack((yij[:,None],i[:,None],j[:,None],x[:,None])),columns=['Yij','i','j','X'])
n = len(data)
model = smf.mixedlm("Yij ~ X", data, groups=data["i"], re_formula="~X")
model_fit = model.fit(method=["lbfgs"])
fixed_effects_estimates = model_fit.fe_params
fixed_effects = model_fit.bse_fe
random_effects = model_fit.bse_re
residuals = model_fit.resid
random_effect_variance
=((random_effects[0]**2)+(random_effects[1]**2)+(random_effects[2]**2))
residual_mean = np.mean(residuals)
residual_variance = [(x-residual_mean)**2 for x in residuals]
residual_variance = sum(residual_variance)/(n-2)
if math.isnan(random_effect_variance):
  continue
else:
  u.append(random_effect_variance)
  v.append(residual_variance)
```

```
    est_0.append(fixed_effects_estimates[0])
    est_1.append(fixed_effects_estimates[1])
    iter+=1
print(u)
print(v)
```

Q1)iii) Result of Random effect variances from this Simulation run

	Random effect variance	Residual variance
1	0.605800	8846.620392
2	0.174168	9482.006454
3	0.741981	8774.057502
4	0.492879	8834.977915
5	0.218044	9265.398901
6	0.452923	9207.538854

Q1)IV) Here, we're going to choose three different sample sizes (sample size=20,60,300) and run 1000 model simulations with each sample size.

Taking Random samples from the data given and checking for Variances with sample size 20

Code:

```
iter = 0
u1 ,v1= [],[]
while iter != 1000:
  i = np.repeat(np.array([1,2,3,4]),600)
  j = np.repeat(np.array([1,2,3]),200)
  j = np.array(list(j)*4)
  x = np.array([20,40,70]*800)
  x = np.array(x)
  b0 = np.array([-1]*2400)
  b1 = np.array([0.2]*2400)
  ai = np.random.normal(0,1.5,2400)
  bi = np.random.normal(0,2,2400)
  eij = np.random.normal(0,1,2400)
  b1x = b1*x
  bix = bi*x
```

```
  yij = b0 + b1x + bix + ai + eij
  data1 = pd.DataFrame(np.hstack((yij[:,None],i[:,None],j[:,None],x[:,None])),columns=['Yij','i','j','X'])
  data = data1.sample(n = 20)
  n = len(data)
  model = smf.mixedlm("Yij ~ X", data, groups=data["i"], re_formula="~X")
  model_fit = model.fit(method=["lbfgs"])
  fixed_effects = model_fit.bse_fe
  random_effects = model_fit.bse_re
  residuals = model_fit.resid
  random_effect_variance
=((random_effects[0]**2)+(random_effects[1]**2)+(random_effects[2]**2))
  residual_mean = np.mean(residuals)
  residual_variance = [(x-residual_mean)**2 for x in residuals]
  residual_variance = sum(residual_variance)/(n-2)
  if math.isnan(random_effect_variance):
    continue
  else:
    u1.append(random_effect_variance)
    v1.append(residual_variance)
    iter+=1
```

Taking Random samples from the data given and checking for Variances with sample size 60

```
iter = 0
u2 , v2 = [],[]
while iter != 1000:
  i = np.repeat(np.array([1,2,3,4]),600)
  j = np.repeat(np.array([1,2,3]),200)
  j = np.array(list(j)*4)
  x = np.array([20,40,70]*800)
  x = np.array(x)
  b0 = np.array([-1]*2400)
  b1 = np.array([0.2]*2400)
```

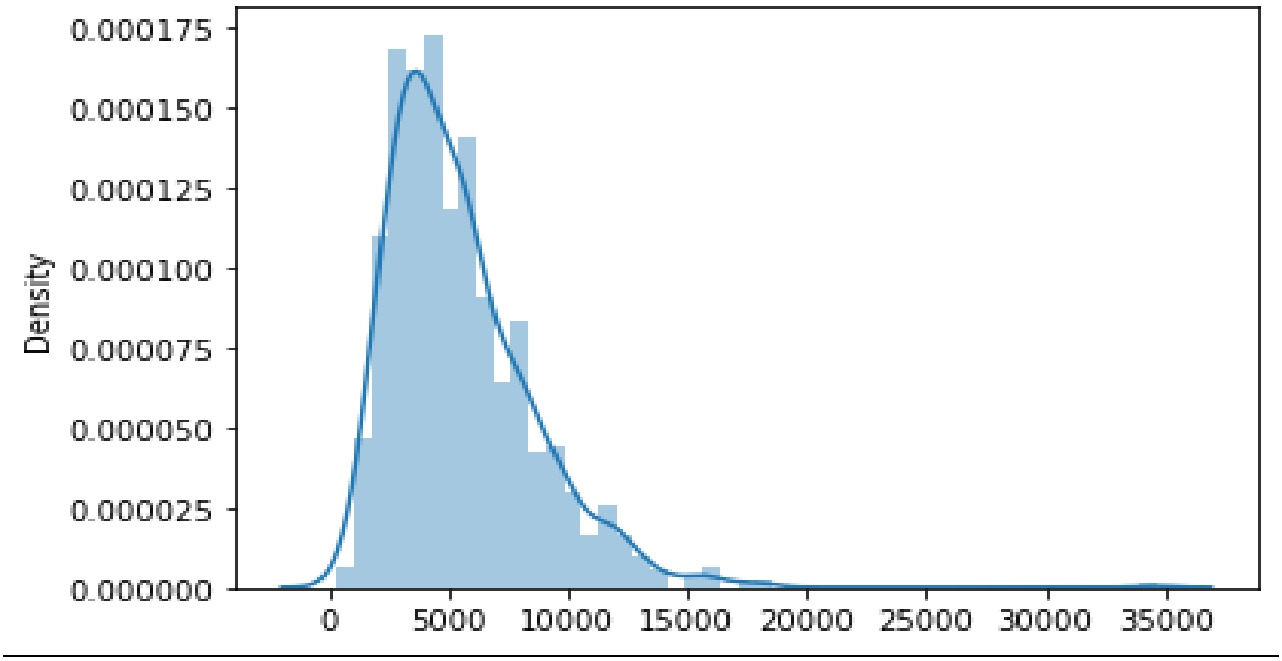
```
  ai = np.random.normal(0,1.5,2400)
  bi = np.random.normal(0,2,2400)
  eij = np.random.normal(0,1,2400)
  b1x = b1*x
  bix = bi*x
  yij = b0 + b1x + bix + ai + eij
  data1 = pd.DataFrame(np.hstack((yij[:,None],i[:,None],j[:,None],x[:,None])),columns=['Yij','i','j','X'])
  data = data1.sample(n = 60)
  n = len(data)
  model = smf.mixedlm("Yij ~ X", data, groups=data["i"], re_formula="~X")
  model_fit = model.fit(method=["lbfgs"])
  fixed_effects = model_fit.bse_fe
  random_effects = model_fit.bse_re
  residuals = model_fit.resid
  random_effect_variance
=((random_effects[0]**2)+(random_effects[1]**2)+(random_effects[2]**2))
  residual_mean = np.mean(residuals)
  residual_variance = [(x-residual_mean)**2 for x in residuals]
  residual_variance = sum(residual_variance)/(n-2)
  if math.isnan(random_effect_variance):
    continue
  else:
    u2.append(random_effect_variance)
    v2.append(residual_variance)
    iter+=1
```

Taking Random samples from the data given and checking for Variances with sample size 300

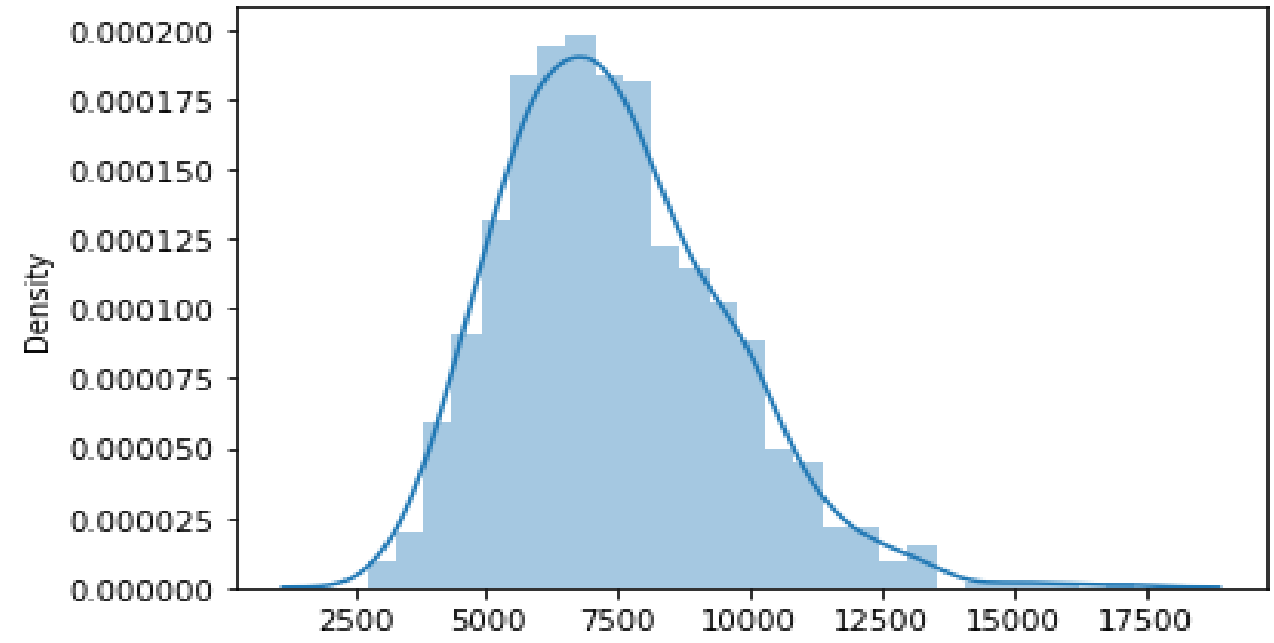
```
iter = 0
u3 ,v3 = [] , []
while iter != 1000:
  i = np.repeat(np.array([1,2,3,4]),600)
  j = np.repeat(np.array([1,2,3]),200)
```

```
  j = np.array(list(j)*4)
  x = np.array([20,40,70]*800)
  x = np.array(x)
  b0 = np.array([-1]*2400)
  b1 = np.array([0.2]*2400)
  ai = np.random.normal(0,1.5,2400)
  bi = np.random.normal(0,2,2400)
  eij = np.random.normal(0,1,2400)
  b1x = b1*x
  bix = bi*x
  yij = b0 + b1x + bix + ai + eij
  data1 = pd.DataFrame(np.hstack((yij[:,None],i[:,None],j[:,None],x[:,None])),columns=['Yij','i','j','X'])
  data = data1.sample(n = 300)
  n = len(data)
  model = smf.mixedlm("Yij ~ X", data, groups=data["i"], re_formula="~X")
  model_fit = model.fit(method=["lbfgs"])
  fixed_effects = model_fit.bse_fe
  random_effects = model_fit.bse_re
  residuals = model_fit.resid
  random_effect_variance
=((random_effects[0]**2)+(random_effects[1]**2)+(random_effects[2]**2))
  residual_mean = np.mean(residuals)
  residual_variance = [(x-residual_mean)**2 for x in residuals]
  residual_variance = sum(residual_variance)/(n-2)
  if math.isnan(random_effect_variance):
    continue
  else:
    u3.append(random_effect_variance)
    v3.append(residual_variance)
    iter+=1
```

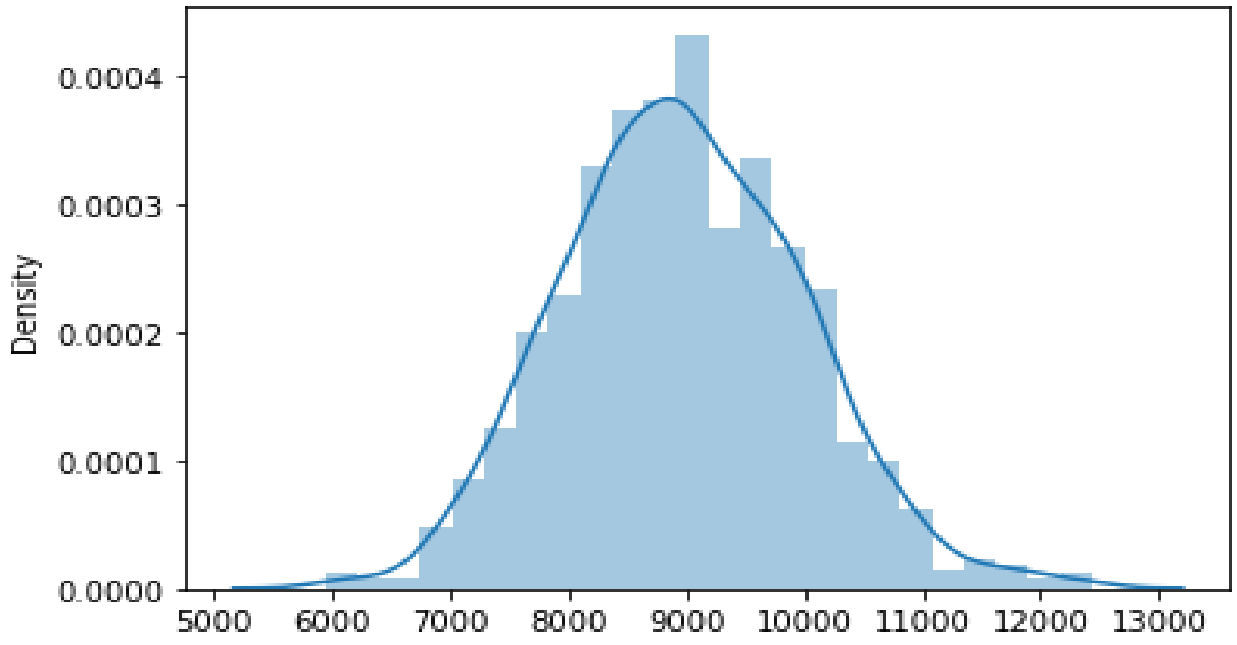
Plot of distribution of Residual variance for the three samples:



For 1st sample of sample size=20



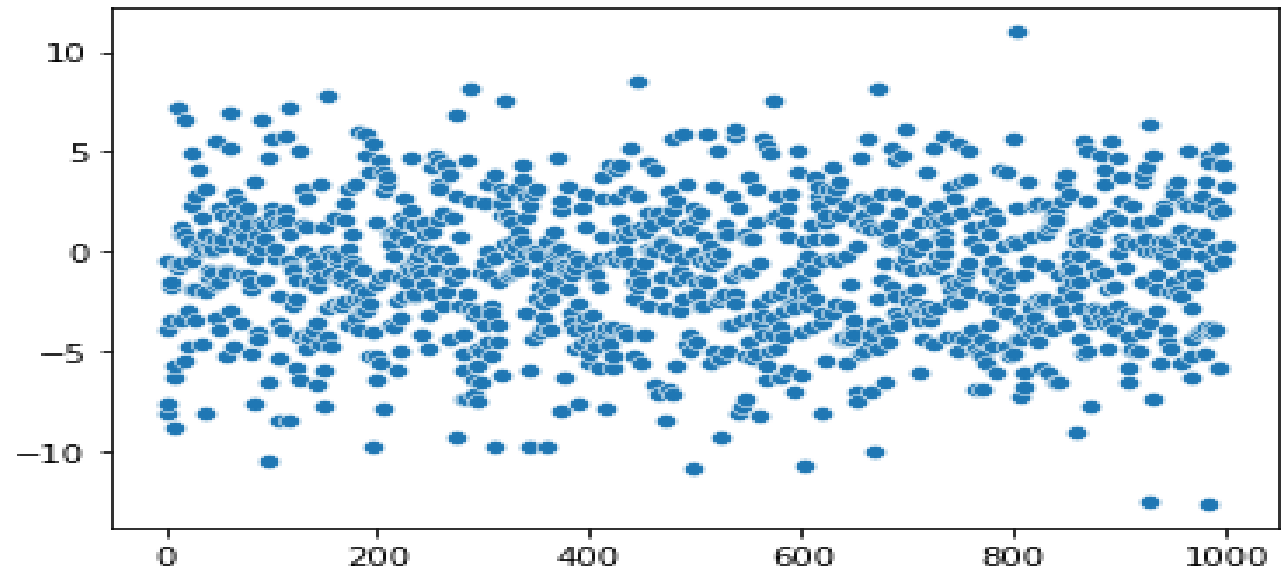
For 2nd sample of sample size=60



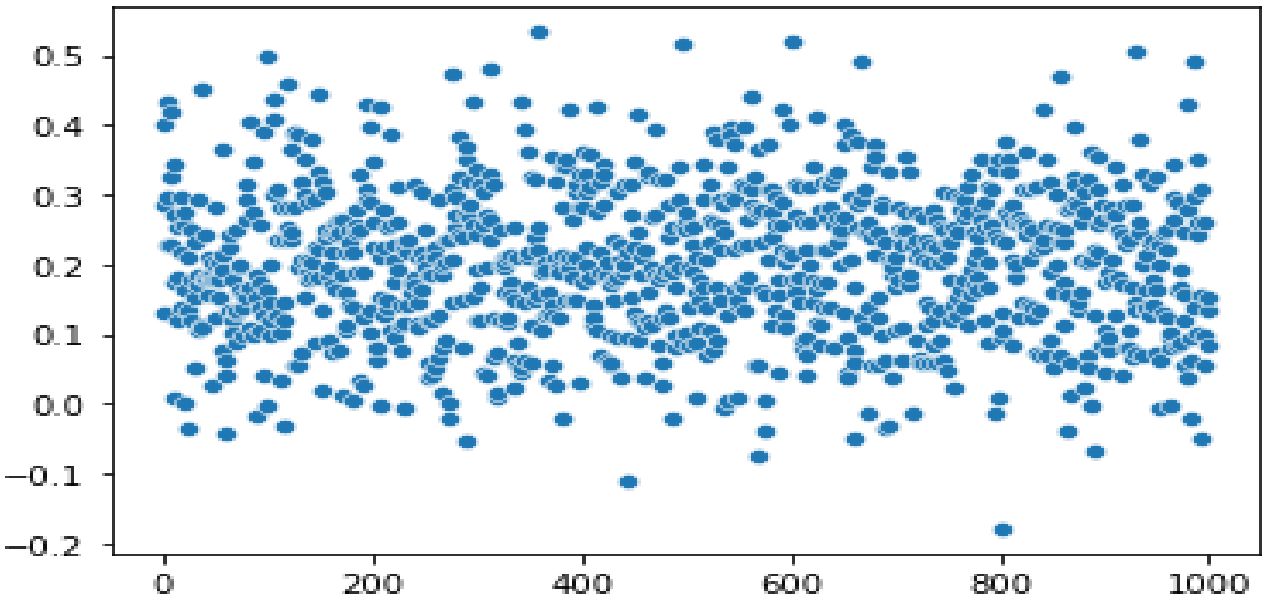
For 3rd sample of sample size=300

Q1)V) <u>Plot for estimates of Fixed effects:</u>

For 1st Estimate:



For 2nd Estimate:



(Q2) A public health study was conducted to estimate the association between maternal smoking and respiratory health of children in two cities Goteborg and Lund.

Let y_{ij} be the wheezing indicator on the ith child at the jth age t_{ij} , where t_{ij} ideally takes on all values 9; 10; 11; 12. For each child i, let

x_{0ij} = 1 if smoking = none at t_{ij}

x_{0ij} = 0 otherwise

x_{1ij} = 1 if smoking = moderate at t_{ij}

x_{1ij} = 0 otherwise

ci = 0 if city = Göteborg

ci = 1 if city = Lund

Table1 Logistic regression result (independent responses)

whz	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
x0	-.7347176	.5406551	-1.36	0.174	-1.794382	.3249469
x1	-.8623741	.5199692	-1.66	0.097	-1.881495	.1567469
cind	.2117842	.4010502	0.53	0.597	-.5742597	.9978281
time	-.1993475	.1803634	-1.11	0.269	-.5528533	.1541583
_cons	1.679783	1.952625	0.86	0.390	-2.147292	5.506858

(f)

Table 2. GEE results with unstructured correlation matrix.

whz	Coef.	Semi-robust Std. Err.	z	P>|z|	[95% Conf.	Interval]
x0	-.8193055	.4853743	-1.69	0.091	-1.770622	.1320106
x1	-.8416823	.5060132	-1.66	0.096	-1.83345	.1500853
cind	.2001139	.411357	0.49	0.627	-.606131	1.006359
time	-.2144158	.1804719	-1.19	0.235	-.5681342	.1393027
_cons	1.903247	1.862532	1.02	0.307	-1.747248	5.553742

Table 3. Estimated within-subject correlation matrix

	c1	c2	c3	c4
r1	1.0000			
r2	-0.0932	1.0000		
r3	0.0543	0.2669	1.0000	
r4	0.0231	-0.0708	0.0768	1.0000

(g)

Table 4. Logistic regression of wheezing on past and present smoking.

whz	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
x0	-1.273384	.6339722	-2.01	0.045	-2.515947	-.0308217
x1	-1.235653	.6262809	-1.97	0.048	-2.463141	-.0081652
x0m1	-.0052262	.9997849	-0.01	0.996	-1.964769	1.954316
x1m1	-.2210449	.810579	-0.27	0.785	-1.809751	1.367661
cind	.5702549	.524679	1.09	0.277	-.458097	1.598607
time	-.1071572	.3212241	-0.33	0.739	-.7367449	.5224306
_cons	.9027534	3.613741	0.25	0.803	-6.180049	7.985556

Table 5. Transitional logistic regression model with past wheezing as a covariate.

whz	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
x0	-1.08873	.6442791	-1.69	0.091	-2.351494	.1740336
x1	-1.304317	.5991779	-2.18	0.029	-2.478684	-.1299495
whzm1	.3495777	.5113633	0.68	0.494	-.6526759	1.351831
cind	.5113185	.5191123	0.98	0.325	-.5061228	1.52876
time	.0141006	.330052	0.04	0.966	-.6327895	.6609906
_cons	-.6796604	3.800424	-0.18	0.858	-8.128354	6.769033

(h)

Table 6. Results from the random intercept model.

whz	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
cind	.2168998	.4234064	0.51	0.608	-.6129616	1.046761
x0	-.7577636	.5637224	-1.34	0.179	-1.862639	.347112
x1	-.8792885	.5362379	-1.64	0.101	-1.930296	.1717185
time	-.2041793	.183191	-1.11	0.265	-.563227	.1548685
_cons	1.719086	1.983047	0.87	0.386	-2.167614	5.605786
/lnsig2u	-2.168139	3.734647			-9.487913	5.151635
sigma_u	.3382163	.6315593			.0087041	13.14206
rho	.0336021	.0368633			.000023	.9813079

(Q3)

Generated the response variable

Code:

```
import numpy as np

import math

import pandas as pd

x1 = np.random.normal(0 ,1 ,150)

x2 = np.random.normal(0 ,1 ,150)

x3 = np.random.normal(0 ,1 ,150)

b0 = np.array([0.67]*150)

b1 = np.array([-0.32]*150)

b1x1 = b1*x1

y1 = []

for i in range(150):

 y1.append(math.exp(b0[i]+b1x1[i])/( 1 + math.exp(b0[i]+b1x1[i])))

y = np.array(y1)

x = pd.DataFrame((np.hstack((x1[:,None],x2[:,None],x3[:,None]))),columns=['x1','x2','x3'])
```

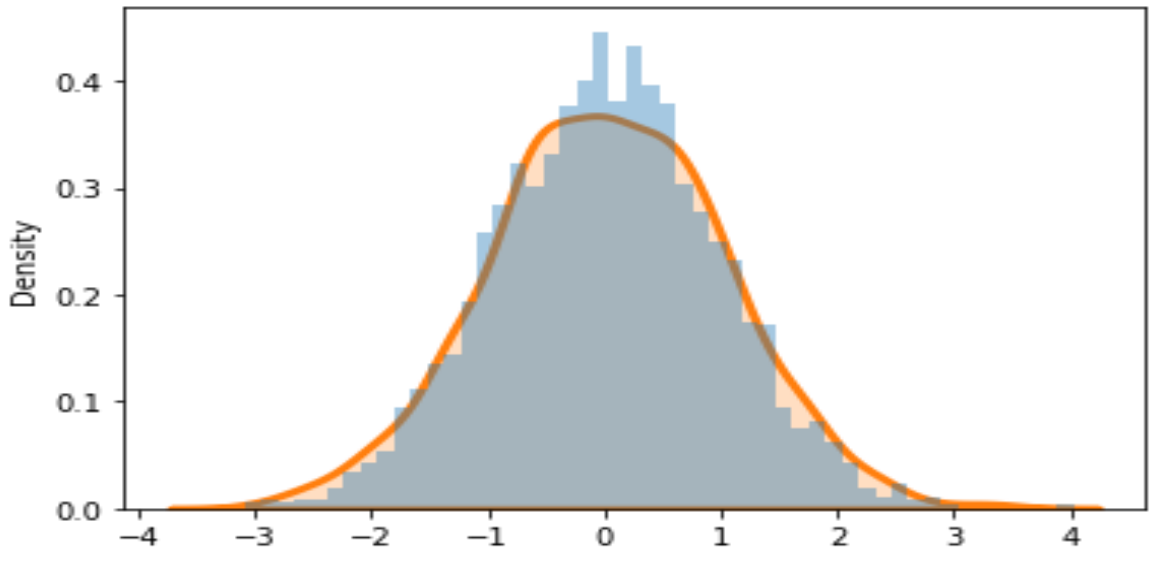
(a) To investigate the distributions, we're going to simulate from this model 2500 times. Each time, fit the model:

$$\log(p(x)/\{1-p(x)\}) = \beta_0+\beta_1x_1+\beta_2x_2+\beta_3x_3$$

Code:

```
import statsmodels.api as sm
import warnings
warnings.filterwarnings("ignore")
u , v = [] , []
for i in range(2500):
 x1 = np.random.normal(0 ,1 ,150)
 x2 = np.random.normal(0 ,1 ,150)
 x3 = np.random.normal(0 ,1 ,150)
 x_1 = pd.DataFrame((np.hstack((x1[:,None],x2[:,None],x3[:,None]))),columns=['x1','x2','x3'])
 x_2 = pd.DataFrame((np.hstack(x1[:,None])),columns=['x1'])
 exog_1 , exog_2 , end_og = sm.add_constant(x_1),sm.add_constant(x_2), y
 model_1 = sm.GLM(end_og, exog_1,family=sm.families.Binomial(link=sm.families.links.logit))
 model_2 = sm.GLM(end_og, exog_2,family=sm.families.Binomial(link=sm.families.links.logit))
 model_1_fit , model_2_fit = model_1.fit() , model_2.fit()
 v.append(2*(model_1_fit.llf - model_2_fit.llf))
 u.append((((150**0.5)*(model_1_fit.params[2]))/(model_1_fit.bse[2])))
print(v)
print(u)
```

(b) Histogram for the wald statistics and histogram for SNV

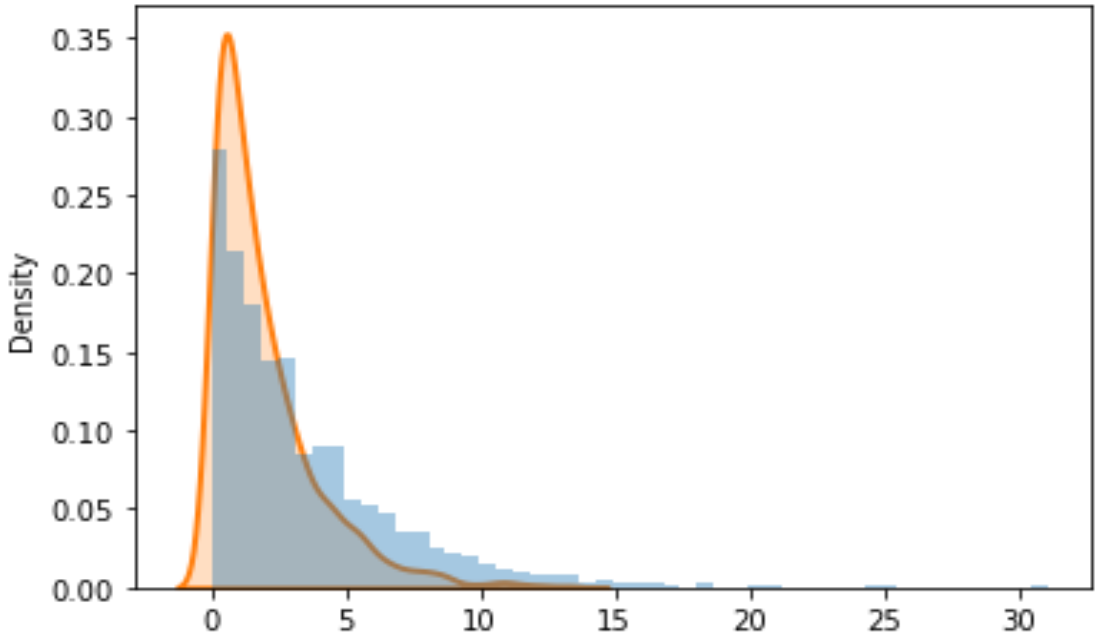


(c)

```
Estimated probability (observing the test statistic larger than 1): 0.304

Estimated probability (the true distribution of the test statistic assuming
a large sample): 0.15865525393145707
```

(d)

Histogram for the Likelihood Ratio statistic and histogram for Chi square distribution with 2 DF



(e)

```
Estimated probability (observing a test statistic larger than 5): 0.0056

Estimated probability (the true distribution of the test statistic assuming
a large sample): 0.08208499862389884
```

(f) Now, we're going to run the entire simulation but with smaller sample size = 10

Code:

```
x1 = np.random.normal(0 ,1 ,10)

x2 = np.random.normal(0 ,1 ,10)

x3 = np.random.normal(0 ,1 ,10)

b0 = np.array([0.67]*10)

b1 = np.array([-0.32]*10)

b1x1 = b1*x1

y1 = []

for i in range(10):
```

```
  y1.append(math.exp(b0[i]+b1x1[i])/( 1 + math.exp(b0[i]+b1x1[i])))
y = np.array(y1)
x = pd.DataFrame((np.hstack((x1[:,None],x2[:,None],x3[:,None]))),columns=['x1','x2','x3'])
u , v = [] , []
for i in range(2500):
  x1 = np.random.normal(0 ,1 ,10)
  x2 = np.random.normal(0 ,1 ,10)
  x3 = np.random.normal(0 ,1 ,10)
  x_1 = pd.DataFrame((np.hstack((x1[:,None],x2[:,None],x3[:,None]))),columns=['x1','x2','x3'])
  x_2 = pd.DataFrame((np.hstack(x1[:,None])),columns=['x1'])
  exog_1 , exog_2 , end_og = sm.add_constant(x_1),sm.add_constant(x_2), y
  model_1 = sm.GLM(end_og, exog_1,family=sm.families.Binomial(link=sm.families.links.logit))
  model_2 = sm.GLM(end_og, exog_2,family=sm.families.Binomial(link=sm.families.links.logit))
  model_1_fit , model_2_fit = model_1.fit() , model_2.fit()
  v.append(2*(model_1_fit.llf - model_2_fit.llf))
  u.append((((10**0.5)*(model_1_fit.params[2]))/(model_1_fit.bse[2])))
print(v)
print(u)
```

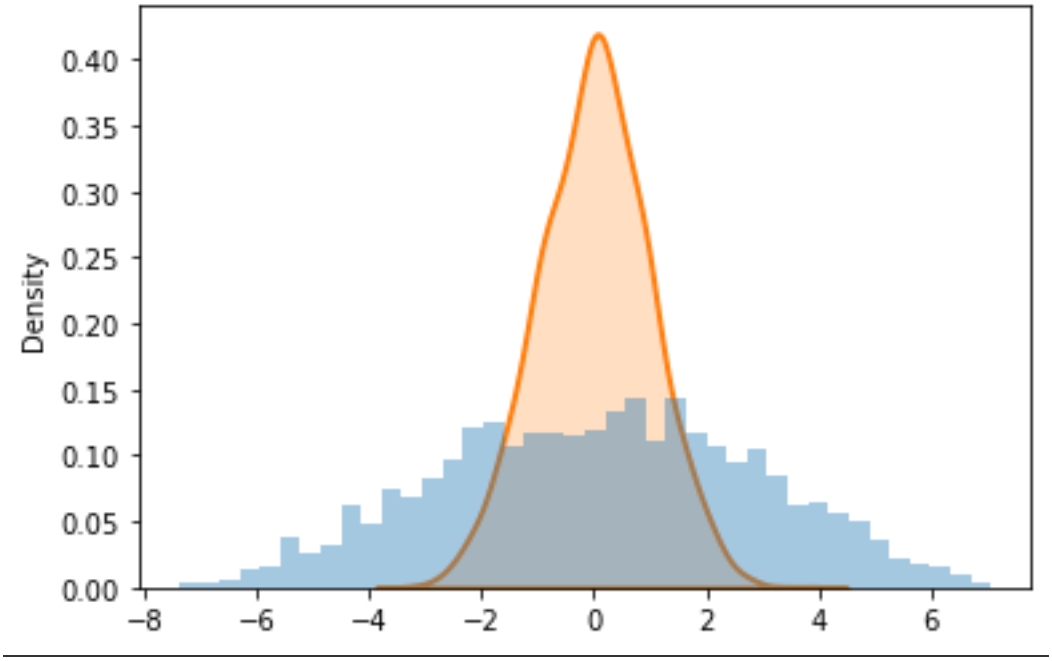
Repeat (a)-(e)

Code:

```
u1 = np.array(u)*5
v1 = np.array(v)*70
plt.hist(u1 , density=True , bins = 40, alpha=0.4)
sns.distplot(np.random.normal( 0 , 1 ,2500) , hist = False,kde_kws = {'shade': True, 'linewidth': 2})
plt.show()
count = 0
for i in u:
  if i>1:
    count +=1
prob = count/len(u)
print(prob)
```

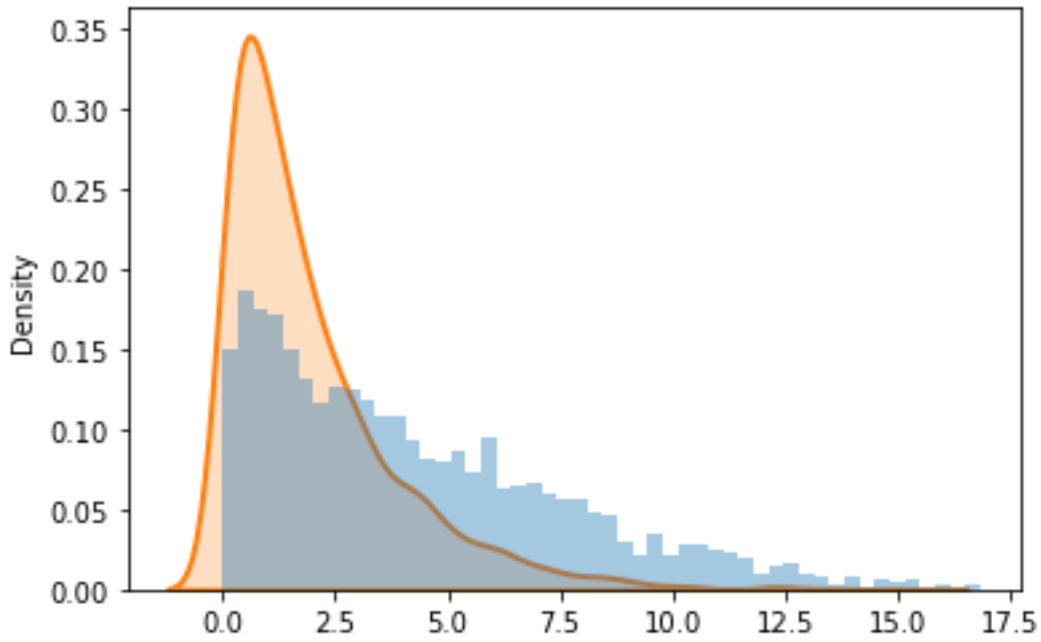
```
print(1-norm.cdf(1))

plt.hist(v1 , density =True, bins = 50 , alpha=0.4)

sns.distplot(np.random.chisquare( 2 ,2500) , hist = False,kde_kws = {'shade': True, 'linewidth': 2})

plt.show()

count = 0

for i in u:

  if i>5:

    count +=1

prob = count/len(u)

print(prob)

print(1-chi2.cdf(5,2))
```

Plot a histogram of the empirical values for the Wald test statistic




```
Estimated probability (observing the test statistic larger than 1): 0.0348

Estimated probability (the true distribution of the test statistic assuming
a large sample): 0.15865525393145707
```

Plot a histogram of the empirical values for the likelihood ratio test statistic




```
Estimated probability (observing a test statistic larger than 5): 0.0

Estimated probability (the true distribution of the test statistic assuming
a large sample): 0.08208499862389884
```

Explanation

A sample size of 10 does not seem large enough to use either a standard normal or chi-squa red distribution. It is noticeable that the histogram of the empirical results and the curve of t he true distribution don't match well. The estimated and true probabilities are also pretty far off.