STA2101 Project

December 19, 2018
Jessica Chau
Student ID: 998286554

Introduction:

The purpose of this study is to draw a conclusion on whether the 'Beating the Blues' (BtheB) treatment improves patients' depression scores (evaluated by the Beck Depression Inventory score) compared to the treatment as usual (TAU). The analysis of this data is important in understanding whether patients with depression improved when the BtheB treatment is used. In the data collected, the BDI scores of the same patients were evaluated after 2months, 4months, 6months, and 8months of the treatment and the data includes information on whether the patients were taking anti-depressants and the length of the episode of depression (in months).

The Beck Depression Inventory (BDI) has four standard cut-off scores as follows:

- 0-9 points indicates minimal depression
- 10-18 points indicates mild depression
- 19-29 points indicates moderate depression
- 30-63 points indicates severe depression

This is a between case design, and from the data, it does appear that the candidates were randomly assigned to either BtheB or TAU based on the frequency table calculated (Appendix B-2).

Statistical Methods:

The BDI test comprises of 21 questions and each question is worth three points which results in a maximum score of 63 points. To evaluate whether the Beating the Blues treatment improved compared to the baseline, the scores were categorized into the four standard cut-offs: minimal, mild, moderate and severe (these cut-off scores are the standard groupings for BDI). Then the prior treatment scores were compared against the post-treatment scores. If the patient improved after the treatment (at time t) from the original period, t0, the response is classified as one. Otherwise, the response is zero if the patient remained in the same category or did not improve.

Exploratory analyzes were conducted to better understand the data (Appendix B-4 to Appendix B-7) and noticed a few trends:

- In Appendix B-4, the BtheB treatment has a higher success rate
- In Appendix B-5, the BtheB treatment has a higher success rate for length >6m
- In Appendix B-6, the BtheB treatment has a higher success rate for no drugs

To confirm these trends, the Pearson chi-squared test is computed and the findings show that the results are statistically significant. BtheB treatment has a higher improvement rates in a person's depression after 2 months compared to the treatment as usual.

A logistic regression is plotted to confirm that Beating the Blues treatment improves the patient's BDI scores after 2 months and to understand the variables that best describe the dataset.

The first logistic regression plotted is:

$$y = \beta_0 + \beta_1 * treatment\_type + \beta_2 * length + \beta_3 * drug$$

In the logistic summary output, length (Appendix B – 9) is not significant with an $\alpha = 0.10$.

Earlier in the exploratory analyzes, it was found that the proportion of success was higher for certain 'length' and 'drug use'. To confirm if the model relied on these variables, 'length' and 'drug' was removed from the model and treatment was the only explanatory variable (Appendix B – 8). As a result, the second logistic model is computed as:

$$y = \beta_0 + \beta_1 * treatment\_type$$

Note that treatment type, length and drug are categorical variables and from the summary table, (Appendix B – 8) it appears that treatment is significant (at significance level = 0.10). To test the goodness of fit of the model, and whether the reduced model can be used, the likelihood ratio test is performed. The computed likelihood ratio indicates that the null hypothesis cannot be rejected (the null hypothesis states that the reduced model is true). More models were computed (to determine the interaction terms) and the correlation matrix is calculated. From the correlation matrix, there does not appear to be any strong correlations between the explanatory variables (Appendix B-3). Because of this, no explanatory variables were excluded in the model testing.

More models were fitted (including interaction terms between the variables) and the likelihood ratio test were applied to confirm whether the reduced model had better performance (Appendix B-9 to Appendix B-13). The interaction terms appears to be significant so a Wald test will be performed to test this.

$$\text{y} = \beta_0 + \beta_1 * trmt + \beta_2 * length + \beta_3 * drug + \beta_4 * length * trmt + \beta_5 * drug * trmt$$

The Wald test is computed to test whether a variable is necessary in the logistic regression (Appendix B-14). The null hypothesis tests that the coefficient of the test variable equals to zero. In the end, the model that had the better fit of the data as indicated by the Likelihood Ratio and Wald tests was:

$$\text{y} = \beta_0 + \beta_1 * treatment + \beta_2 * drug + \beta_3 * (length * treatment)$$

The model was computed using the response variable of the 2month BDI results. There were only three missing data values in the 2monh responses, as a result, these missing rows were removed from the analyzes. The data for 4months, 6months and 8months had a larger number of missing values, as a result, imputation will be used.

Mean imputation is computed and compared against the results when the missing data values were removed. At timeframes 4 and 6 months, mean imputation and the non-missing data values

shows the same results. There appears to be an improvement over time when BtheB treatment is used. A chi-squared test is computed to test if BtheB treatment has the same proportion of improvements as TAU (Appendix B-15 to Appendix B-20) but the test shows that the null hypothesis cannot be reject at the 0.05 significance level and there is a chance that the BtheB treatment is similar to TAU at the 4month and 6month period after treatment.

At 8months, imputation and actual results also show the same trend. TAU appears to be performing slightly better than BtheB (Appendix B-16 to Appendix B-17). The chi-squared test is calculated and found that we cannot dismiss that TAU and BtheB may have similar improvement rates (Appendix B-18).

Conclusion:

In conclusion, it was found that 'Beating the Blues' tend to perform better than 'Treatment as Usual' two months after the treatment under two circumstances:

- When patients were not taking medication (drugs)
- When patents' episode of depression (length) were over 6months

In this analyzes, improvement is defined as a patient who moved from a severe category into a less severe depression category. The data was formatted as a binary response. The BDI questionnaire only contains 21 questions, thus answering one question incorrectly would impact the scores by 5% (1/21). The cut-off categories of the BDI scores were established by researchers and thus by converting the fields to binary, it has a stronger test of whether Treatment as Usual and Beating the Blues were effective.

Discussion of the issues:

1. In python, it was important to understand what libraries were available and whether they were credible sources to use (due diligence had to be performed here)
2. The exploratory analysis of the data is important and took a lot of time
3. The model selection process is also time consuming; this is due to testing several variables of inputs into the model
4. There are several imputation approaches, due to constraints on time, the means imputation was computed in the model

Being new to Python, this assignment took more time than it would have in R due to syntax and researching what libraries are available. In addition, there were not as many available statistical libraries available compared to R such as likelihood ratio test for logistic regressions (i.e anova or lrtest functions) and this results in more time spent on testing the models.

Appendix A: Python Program

The python program contains the following items:

1. The code written to clean the data for example:
    a. Converting the '…' in the data to 'NaN'
    b. Removing the nulls
    c. Plotted the frequency of missing values
    d. Modifying the datatype from strings into numerical values in the bdi_2m/bdi_4m etc.
    e. Converted the response variable to be 1 or 0 (1 if there was an improvement after the treatment to the pre-treatment, 0 otherwise)

2. Exploratory analysis
    a. Created frequency tables to show the proportion of success by factors (split several times)
    b. Computed the correlation matric of the variables
    c. Computed the summary statistics (i.e. mean/median/min/max) of the fields

3. Statistical Analyzes
    a. Computed the chi-squared test to confirm whether H0: BtheB = TAU for drug/length/treatment
    b. Computed the Likelihood Ratio tests for model selection
    c. Computed the Wald tests for variable selection in the model

4. Imputation Exercise
    a. Means imputation applied to 4month/6month/8months of data and the results are compared to no imputation (just removing the null rows)
    b. Chi square test performed to test if H0: TAU = BtheB
    c. Results compared between imputation vs. no imputation results

In [1]:
```
# Import the Libraries
import pandas as pd
import numpy as np
import statsmodels.api as sm
import statsmodels.formula.api as smf
```

Did the treatment program worked?

https://en.wikipedia.org/wiki/Beck_Depression_Inventory (https://en.wikipedia.org/wiki/Beck_Depression_Inventory)

When the test is scored, a value of 0 to 3 is assigned for each answer and then the total score is compared to a key to determine the depression's severity. The standard cut-off scores were as follows:[7]

- 0–9: indicates minimal depression
- 10–18: indicates mild depression
- 19–29: indicates moderate depression
- 30–63: indicates severe depression

In [2]:
```
# Import the data
dataset = pd.read_csv("http://www.utstat.toronto.edu/~brunner/data/legal/BeatTheBlues.data.txt", delim_whitespace=True)
df = dataset.replace('.', np.nan)
df.head()
```

Out[2]:

	drug	length	treatment	bdi_pre	bdi_2m	bdi_4m	bdi_6m	bdi_8m
1	No	>6m	TAU	29	2	2	NaN	NaN
2	Yes	>6m	BtheB	32	16	24	17	20
3	Yes	<6m	TAU	25	20	NaN	NaN	NaN
4	No	>6m	BtheB	21	17	16	10	9
5	Yes	>6m	BtheB	26	23	NaN	NaN	NaN

In [3]:
```
# Compute the pairwise correlation of the variables
df_corr = df[['drug','length','treatment','bdi_pre']]
df_corr.apply(lambda x : pd.factorize(x)[0]).corr(method='pearson', min_
periods=1)
```

Out[3]:

	drug	length	treatment	bdi_pre
drug	1.000000	0.138629	0.287103	0.013332
length	0.138629	1.000000	0.020821	-0.020498
treatment	0.287103	0.020821	1.000000	0.049253
bdi_pre	0.013332	-0.020498	0.049253	1.000000

In [4]:
```
# Counting the frequency of nulls, possibly use imputation
# There's a lot of nulls in 4,6,8 months
df.isna().sum()
```

Out[4]:
```
drug          0
length        0
treatment     0
bdi_pre       0
bdi_2m        3
bdi_4m       27
bdi_6m       42
bdi_8m       48
dtype: int64
```

In [5]:
```
# The frequency of the columns are fair, with fewer samples with Yes Dru
gs, >6m, and TAU
df_freq1 = df[['drug', 'length', 'treatment', 'bdi_pre']].groupby(['dru
g']).agg(['count'])
df_freq1
df_freq2 = df[['drug', 'length', 'treatment', 'bdi_pre']].groupby(['leng
th']).agg(['count'])
df_freq2
df_freq3 = df[['drug', 'length', 'treatment', 'bdi_pre']].groupby(['trea
tment']).agg(['count'])
df_freq3
df_freq4 = df[['drug', 'length', 'treatment', 'bdi_pre']].groupby(['dru
g', 'length','treatment']).agg(['count'])
df_freq4
```

Out[5]:

			bdi_pre
			count
drug	length	treatment	
No	<6m	BtheB	9
		TAU	15
	>6m	BtheB	13
		TAU	19
Yes	<6m	BtheB	17
		TAU	8
	>6m	BtheB	13
		TAU	6

In [6]:
```
# Classify where 1 = mild; 2=minimal; 3=moderate; 4=severe
conditions = [
    (df['bdi_pre'] < 10),
    (df['bdi_pre'] < 19),
    (df['bdi_pre'] <30)]
choices = [1, 2, 3]
df['pre_response'] = np.select(conditions, choices, default=4)
```

In [7]:
```
# There are few people with mild depression
df_freq5 = df[['bdi_pre','pre_response']].groupby(['pre_response']).agg
(['count'])
df_freq5
```

Out[7]:

	bdi_pre
	count
pre_response	
1	10
2	27
3	33
4	30

In [8]:
```
# Remove nulls from 2months and compute the model
df1 = df[['drug', 'length', 'treatment', 'bdi_pre', 'pre_response','bdi_
2m']]
moddf = df1.dropna()
moddf['bdi_2m'] = pd.to_numeric(moddf['bdi_2m'])
moddf['bdi_2m'].astype(str).astype(int)
moddf.describe()
```

```
/Users/jchau/anaconda3/lib/python3.6/site-packages/ipykernel_launcher.p
y:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-d
ocs/stable/indexing.html#indexing-view-versus-copy
  after removing the cwd from sys.path.
```

Out[8]:

	bdi_pre	pre_response	bdi_2m
count	97.000000	97.000000	97.000000
mean	23.154639	2.824742	16.917526
std	10.786122	0.979055	10.786440
min	2.000000	1.000000	0.000000
25%	15.000000	2.000000	8.000000
50%	22.000000	3.000000	15.000000
75%	30.000000	4.000000	23.000000
max	49.000000	4.000000	48.000000

In [9]:
```
# Compute the conditions of the model
conditions = [
    (moddf['bdi_2m'] < 10),
    (moddf['bdi_2m'] < 19),
    (moddf['bdi_2m'] <30)]
choices = [1, 2, 3]
moddf['2m_response'] = np.select(conditions, choices, default=4)
```

```
/Users/jchau/anaconda3/lib/python3.6/site-packages/ipykernel_launcher.p
y:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-d
ocs/stable/indexing.html#indexing-view-versus-copy
```

In [10]:
```
# Classify as 1 if there's an improvement and 0 otherwise
moddf['2m_y0'] = moddf['pre_response'] - moddf['2m_response']
moddf.head()
# Based on the difference, I will classify the response variable as 1 (i
f the treatement worked) or 0 if no change in treatment or gotten worse
def f(row):
    if row['2m_y0']>0:
        val = 1
    else:
        val = 0
    return val
moddf['2m_y'] = moddf.apply(f, axis=1)
```

```
/Users/jchau/anaconda3/lib/python3.6/site-packages/ipykernel_launcher.p
y:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-d
ocs/stable/indexing.html#indexing-view-versus-copy
  """Entry point for launching an IPython kernel.
/Users/jchau/anaconda3/lib/python3.6/site-packages/ipykernel_launcher.p
y:10: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-d
ocs/stable/indexing.html#indexing-view-versus-copy
  # Remove the CWD from sys.path while we load stuff.
```

In [11]:
```
# Exploratory view of the data
moddf.describe()
```

Out[11]:

	bdi_pre	pre_response	bdi_2m	2m_response	2m_y0	2m_y
count	97.000000	97.000000	97.000000	97.000000	97.000000	97.000000
mean	23.154639	2.824742	16.917526	2.247423	0.577320	0.443299
std	10.786122	0.979055	10.786440	1.108968	0.955631	0.499355
min	2.000000	1.000000	0.000000	1.000000	-1.000000	0.000000
25%	15.000000	2.000000	8.000000	1.000000	0.000000	0.000000
50%	22.000000	3.000000	15.000000	2.000000	0.000000	0.000000
75%	30.000000	4.000000	23.000000	3.000000	1.000000	1.000000
max	49.000000	4.000000	48.000000	4.000000	3.000000	1.000000

In [12]:
```
dat = moddf[['drug', 'length', 'treatment','2m_y']]
dat.head()
```

Out[12]:

	drug	length	treatment	2m_y
1	No	>6m	TAU	1
2	Yes	>6m	BtheB	1
3	Yes	<6m	TAU	0
4	No	>6m	BtheB	1
5	Yes	>6m	BtheB	0

In [13]:
```
dat1 = pd.crosstab(index = dat['treatment'], columns= dat['2m_y'], margi
ns=True)
dat1['PropSucc'] = dat1[1]/dat1['All']
dat1
# It appears that the BtheB success rate is 52% and the TAU is 35%
```

Out[13]:

2m_y	0	1	All	PropSucc
treatment				
BtheB	25	27	52	0.519231
TAU	29	16	45	0.355556
All	54	43	97	0.443299

In [18]:
```
(27/25)/(16/29) # odds of the BtheB being effective

# Chi squared test

# H0: BtheB is the same as TAU
from scipy.stats import chi2_contingency
from scipy.stats import chi2

table = [[25,27],
        [29,16]]
stat, p, dof, expected = chi2_contingency(table)
# interpret test-statistic
prob = 0.95
critical = chi2.ppf(prob, dof)
print('probability=%.3f, critical=%.3f, stat=%.3f' % (prob, critical, st
at))
if abs(stat) >= critical:
    print('Dependent (reject H0)')
else:
    print('Independent (fail to reject H0)')
```

```
probability=0.950, critical=3.841, stat=1.997
Independent (fail to reject H0)
```

In [19]:
```
# Examining if length has an impact on the proportion of successes
dat_length = moddf[['length', 'treatment','2m_y']]
piv1 = pd.pivot_table(dat_length, index = ['length', 'treatment'], colum
ns = ['2m_y'], aggfunc = len)
piv1['propsucc'] = piv1[1]/(piv1[1] + piv1[0])
piv1
# it appears that BtheB seems to do poorer when length is a factor in th
e analysis
```

Out[19]:

	2m_y	0	1	propsucc
length	treatment			
<6m	BtheB	15	11	0.423077
	TAU	11	9	0.450000
>6m	BtheB	10	16	0.615385
	TAU	18	7	0.280000

In [20]:
```
# Calculating the odds ratio
(16/10)/(7/18)
# For length >6months, the BtheB treatment is 4x the odds of performing
 better than TAU

table = [[15,11],
        [11,9]]
stat, p, dof, expected = chi2_contingency(table)
# interpret test-statistic
prob = 0.95
critical = chi2.ppf(prob, dof)
print('probability=%.3f, critical=%.3f, stat=%.3f' % (prob, critical, st
at))
if abs(stat) >= critical:
    print('Dependent (reject H0)')
else:
    print('Independent (fail to reject H0 for <6m)')
```

```
probability=0.950, critical=3.841, stat=0.014
Independent (fail to reject H0 for <6m)
```

In [21]:
```
table1 = [[10,16],
        [18,7]]
stat, p, dof, expected = chi2_contingency(table1)
prob = 0.95
critical = chi2.ppf(prob, dof)
print('probability=%.3f, critical=%.3f, stat=%.3f' % (prob, critical, st
at))
if abs(stat) >= critical:
    print('Dependent (reject H0)')
else:
    print('Independent (fail to reject H0 for >6m)')
```

```
probability=0.950, critical=3.841, stat=4.515
Dependent (reject H0)
```

In [22]:
```
# Examining if drug has an impact on the proportion of successes
dat_drug = moddf[['drug', 'treatment','2m_y']]
piv1 = pd.pivot_table(dat_drug, index = ['drug', 'treatment'], columns =
 ['2m_y'], aggfunc = len)
piv1['propsucc'] = piv1[1]/(piv1[1] + piv1[0])
piv1
# with drug as a characteristic, it appears that BtheB works well when p
atients are not using drugs
```

Out[22]:

	2m_y	0	1	propsucc
drug	treatment			
No	BtheB	11	11	0.500000
	TAU	24	9	0.272727
Yes	BtheB	14	16	0.533333
	TAU	5	7	0.583333

In [23]:
```
# Calculating the odds ratio
(11/1)/(9/24)
# For no drugs, the BtheB treatment has 29x the odds of performing bette
r than TAU
```

Out[23]: 29.333333333333332

In [24]:
```
# Examining if drug has an impact on the proportion of successes
piv = pd.pivot_table(dat, index = ['drug', 'length', 'treatment'], colum
ns = ['2m_y'], aggfunc = len)
piv['propsucc'] = piv[1]/(piv[1] + piv[0])
piv
```

Out[24]:

		2m_y	0	1	propsucc
drug	length	treatment			
No	<6m	BtheB	5	4	0.444444
		TAU	8	6	0.428571
	>6m	BtheB	6	7	0.538462
		TAU	16	3	0.157895
Yes	<6m	BtheB	10	7	0.411765
		TAU	3	3	0.500000
	>6m	BtheB	4	9	0.692308
		TAU	2	4	0.666667

Plotting a logistic regression

In [25]:
```
dat_log = dat.replace(['No', 'Yes'], [0,1])
dat_log = dat.replace(['<6m', '>6m'], [0,1])
dat_log = dat.replace(['TAU', 'BtheB'], [0,1])
dat_log.head()

dat_log['y'] = dat_log['2m_y']
```

In [26]:
```
explanatory_var = dat_log[['treatment', 'drug', 'length']]
correlations = explanatory_var.corr()
correlations
```

Out[26]:

	treatment
treatment	1.0

In [27]:
```
logitmod1 = smf.logit(formula = 'y ~ treatment', data = dat_log).fit()
print(logitmod1.summary())
```

```
Optimization terminated successfully.
         Current function value: 0.673113
         Iterations 4
                           Logit Regression Results

==============================================================================
=======
Dep. Variable:                      y   No. Observations:
     97
Model:                          Logit   Df Residuals:
     95
Method:                           MLE   Df Model:
      1
Date:                Sun, 09 Dec 2018   Pseudo R-squ.:
0.01979
Time:                        17:51:02   Log-Likelihood:
-65.292
converged:                       True   LL-Null:
-66.610
                                        LLR p-value:
0.1044
==============================================================================
=======
                 coef    std err          z      P>|z|      [0.025
0.975]
------------------------------------------------------------------------------
-------
Intercept     -0.5947      0.311     -1.910      0.056      -1.205
  0.016
treatment      0.6717      0.417      1.610      0.107      -0.146
  1.489
==============================================================================
=======
```

In [28]:
```
logitmod2 = smf.logit(formula = 'y ~ treatment+length+drug', data = dat_
log).fit()
print(logitmod2.summary2())
```

```
Optimization terminated successfully.
         Current function value: 0.662337
         Iterations 4
                         Results: Logit
=================================================================
Model:              Logit            Pseudo R-squared: 0.035
Dependent Variable: y                AIC:              136.4934
Date:               2018-12-09 17:51 BIC:              146.7922
No. Observations:   97               Log-Likelihood:   -64.247
Df Model:           3                LL-Null:          -66.610
Df Residuals:       93               LLR p-value:      0.19291
Converged:          1.0000           Scale:            1.0000
No. Iterations:     4.0000
-----------------------------------------------------------------
                 Coef.  Std.Err.    z    P>|z|   [0.025  0.975]
-----------------------------------------------------------------
Intercept       -0.8728   0.4264 -2.0468 0.0407 -1.7086 -0.0370
length[T.>6m]    0.1804   0.4243  0.4250 0.6708 -0.6513  1.0120
drug[T.Yes]      0.6282   0.4425  1.4197 0.1557 -0.2391  1.4954
treatment        0.4984   0.4394  1.1343 0.2567 -0.3628  1.3596
=================================================================
```

In [29]:
```
# Apply the Pearson Chi Squared Test
piv1 = pd.pivot_table(dat_length, index = ['length', 'treatment'], colum
ns = ['2m_y'], aggfunc = len)
piv1['propsucc'] = piv1[1]/(piv1[1] + piv1[0])
piv1
```

Out[29]:

	2m_y	0	1	propsucc
length	**treatment**			
<6m	**BtheB**	15	11	0.423077
	TAU	11	9	0.450000
>6m	**BtheB**	10	16	0.615385
	TAU	18	7	0.280000

1. let;s try imputation
2. Bonferroni correction

In [30]:
```
# Computing the Likelihood Ratio Test
from scipy.stats.distributions import chi2
def likelihood_ratio(llmin, llmax):
    return(2*(llmax-llmin))
```

In [31]:
```
from scipy.optimize import minimize
from math import exp, log
```

In [32]:
```
# Based on the difference, I will classify the response variable as 1 (i
f the treatement worked) or 0 if no change in treatment or gotten worse
dat_log2 = dat_log

def f(row):
    if row['drug']=='Yes':
        val = 1
    else:
        val = 0
    return val
dat_log['drug'] = dat_log.apply(f, axis=1)

def f(row):
    if row['length']=='>6m':
        val = 1
    else:
        val = 0
    return val
dat_log['length'] = dat_log.apply(f, axis=1)

dat_log['b0'] = '0'
def f(row):
    if row['b0']=='0':
        val = 1
    else:
        val = 1
    return val
dat_log['b0'] = dat_log.apply(f, axis=1)
```

In [33]:
```
parameters = logitmod1.params
parameters = np.asmatrix(parameters.values)
parameters = parameters.transpose()
y = logitmod1.fittedvalues
y = np.asmatrix(y.values)
x = dat_log[['b0','treatment']]
x = np.asmatrix(x.values)

def log_L(parameters):
    term1 = y*x*parameters
    term2 = np.log(1+np.exp(x*parameters))
    result = sum(term1 - term2)
    return result.item(0)

log_L(parameters)
```

Out[33]: 1515.8191057613108

In [34]:
```
parameters2 = logitmod2.params
parameters2 = np.asmatrix(parameters2.values)
parameters2 = parameters2.transpose()
y2 = logitmod2.fittedvalues
y2 = np.asmatrix(y2.values)
x2 = dat_log[['b0','treatment','length','drug']]
x2 = np.asmatrix(x2.values)

def log_L2(parameters2):
    term1 = y2*x2*parameters2
    term2 = np.log(1+np.exp(x2*parameters2))
    result = sum(term1 - term2)
    return result.item(0)
log_L2(parameters2)
```

Out[34]: 1730.491119784164

In [35]:
```
# Finding the MLE for fun
#lik_model = minimize(log_L, np.array([1,1]), method='Nelder-Mead')
#lik_model
```

In [36]:
```
from scipy.stats.distributions import chi2

def likelihood_ratio(llmin, llmax):
    return(2*(llmax-llmin))

L1 = log_L(parameters)
L2 = log_L2(parameters2)
LR = likelihood_ratio(L1,L2)
#LR
p = chi2.sf(LR, 2) # L2 has 2 DoF more than L1
'{0:.10f}'.format(p)
# We do not reject H0, which means we favour the reduced model
```

Out[36]: '0.0000000000'

In [37]:
```
logitmod3 = smf.logit(formula = 'y ~ treatment+drug+treatment*drug', dat
a = dat_log).fit()
print(logitmod3.summary2())
```

```
Optimization terminated successfully.
         Current function value: 0.654265
         Iterations 5
                         Results: Logit
=================================================================
Model:              Logit            Pseudo R-squared: 0.047
Dependent Variable: y                AIC:              134.9274
Date:               2018-12-09 17:51 BIC:              145.2262
No. Observations:   97               Log-Likelihood:   -63.464
Df Model:           3                LL-Null:          -66.610
Df Residuals:       93               LLR p-value:      0.098191
Converged:          1.0000           Scale:            1.0000
No. Iterations:     5.0000
-----------------------------------------------------------------
                 Coef.  Std.Err.    z    P>|z|   [0.025  0.975]
-----------------------------------------------------------------
Intercept       -0.9808   0.3909 -2.5094 0.0121 -1.7469 -0.2147
treatment        0.9808   0.5784  1.6956 0.0900 -0.1529  2.1146
drug             1.3173   0.7040  1.8711 0.0613 -0.0625  2.6971
treatment:drug  -1.1838   0.9008 -1.3142 0.1888 -2.9492  0.5817
=================================================================
```

In [38]:
```
parameters3 = logitmod3.params
parameters3 = np.asmatrix(parameters3.values)
parameters3 = parameters3.transpose()
y3 = logitmod3.fittedvalues
y3 = np.asmatrix(y3.values)
dat_log['treatmentxdrug'] = dat_log['treatment'] * dat_log['drug']
x3 = dat_log[['b0','treatment','drug','treatmentxdrug']]
x3 = np.asmatrix(x3.values)


def log_L3(parameters3):
    term1 = y3*x3*parameters3
    term2 = np.log(1+np.exp(x3*parameters3))
    result = sum(term1 - term2)
    return result.item(0)

log_L3(parameters3)

L1 = log_L(parameters)
L2 = log_L3(parameters3)
LR = likelihood_ratio(L1,L2)
p = chi2.sf(LR, 2) # L2 has 2 DoF more than L1
'{0:.10f}'.format(p)
# We reject H0
```

Out[38]: '0.0000000000'

In [39]:
```
logitmod4 = smf.logit(formula = 'y ~ treatment+length+treatment*length',
 data = dat_log).fit()
print(logitmod4.summary2())
```

```
Optimization terminated successfully.
         Current function value: 0.655905
         Iterations 5
                         Results: Logit
=================================================================
Model:              Logit            Pseudo R-squared: 0.045
Dependent Variable: y                AIC:              135.2455
Date:               2018-12-09 17:51 BIC:              145.5444
No. Observations:   97               Log-Likelihood:   -63.623
Df Model:           3                LL-Null:          -66.610
Df Residuals:       93               LLR p-value:      0.11284
Converged:          1.0000           Scale:            1.0000
No. Iterations:     5.0000
-----------------------------------------------------------------
                  Coef.  Std.Err.    z    P>|z|   [0.025 0.975]
-----------------------------------------------------------------
Intercept        -0.2007   0.4495 -0.4465 0.6553 -1.0816 0.6803
treatment        -0.1095   0.5997 -0.1826 0.8551 -1.2848 1.0658
length           -0.7438   0.6328 -1.1754 0.2398 -1.9841 0.4965
treatment:length  1.5239   0.8488  1.7954 0.0726 -0.1397 3.1876
=================================================================
```

In [40]:
```
parameters4 = logitmod4.params
parameters4 = np.asmatrix(parameters4.values)
parameters4 = parameters4.transpose()
y4 = logitmod4.fittedvalues
y4 = np.asmatrix(y4.values)
dat_log['lengthxtreatment'] = dat_log['treatment'] * dat_log['length']
x4 = dat_log[['b0','treatment','length','lengthxtreatment']]
x4 = np.asmatrix(x4.values)

def log_L4(parameters4):
    term4a = y4*x4*parameters4
    term4b = np.log(1+np.exp(x4*parameters4))
    result = sum(term4a - term4b)
    return result.item(0)

log_L4(parameters4)

L1 = log_L(parameters)
L2 = log_L4(parameters4)
LR = likelihood_ratio(L1,L2)
LR
p = chi2.sf(LR, 1)
'{0:.10f}'.format(p)
```

Out[40]: '0.0000000000'

In [41]:
```
logitmod5 = smf.logit(formula = 'y ~ treatment+length+drug+lengthxtreatm
ent+treatment*drug', data = dat_log).fit()
print(logitmod5.summary2())
```

```
Optimization terminated successfully.
         Current function value: 0.637191
         Iterations 5
                         Results: Logit
=================================================================
Model:              Logit            Pseudo R-squared: 0.072
Dependent Variable: y                AIC:              135.6151
Date:               2018-12-09 17:51 BIC:              151.0634
No. Observations:   97               Log-Likelihood:   -61.808
Df Model:           5                LL-Null:          -66.610
Df Residuals:       91               LLR p-value:      0.087221
Converged:          1.0000           Scale:            1.0000
No. Iterations:     5.0000
-----------------------------------------------------------------
                  Coef.  Std.Err.    z    P>|z|   [0.025 0.975]
-----------------------------------------------------------------
Intercept        -0.5940   0.5140 -1.1556 0.2479 -1.6015 0.4135
treatment         0.1040   0.7568  0.1374 0.8907 -1.3792 1.5873
length           -0.7214   0.6596 -1.0938 0.2741 -2.0143 0.5714
drug              1.3021   0.7149  1.8215 0.0685 -0.0990 2.7033
lengthxtreatment  1.5471   0.8761  1.7659 0.0774 -0.1700 3.2643
treatment:drug   -1.0293   0.9227 -1.1155 0.2646 -2.8376 0.7791
=================================================================
```

In [139]:
```
parameters5 = logitmod5.params
parameters5 = np.asmatrix(parameters5.values)
parameters5 = parameters5.transpose()
y5 = logitmod5.fittedvalues
y5 = np.asmatrix(y5.values)
x5 = dat_log[['b0','treatment','length','drug','lengthxtreatment','treat
mentxdrug']]
x5 = np.asmatrix(x5.values)

def log_L5(parameters5):
    term5a = y5*x5*parameters5
    term5b = np.log(1+np.exp(x5*parameters5))
    result = sum(term5a - term5b)
    return result.item(0)

log_L5(parameters5)

# L1 = log_L(parameters)
# L2 = log_L5(parameters5)
# LR = likelihood_ratio(L1,L2)
# LR
# p = chi2.sf(LR, 1)
# '{0:.10f}'.format(p)
```

Out[139]: 4796.033282280123

In [141]:
```
hypothesis_1 = '(length = 0)'
hypothesis_2 = '(drug = 0)'
hypothesis_3 = '(lengthxtreatment = 0)'
hypothesis_4 = '(treatmentxdrug = 0)'

results = smf.logit(formula = 'y ~ treatment+length+drug+lengthxtreatmen
t+treatmentxdrug', data = dat_log).fit()
```

```
Optimization terminated successfully.
         Current function value: 0.637191
         Iterations 5
```

In [143]:
```
wald_1 = results.wald_test(hypothesis_1)
wald_2 = results.wald_test(hypothesis_2)
wald_3 = results.wald_test(hypothesis_3)
wald_4 = results.wald_test(hypothesis_4)
```

In [145]:
```
wald_1
```

Out[145]:
```
<class 'statsmodels.stats.contrast.ContrastResults'>
<Wald test (chi2): statistic=[[1.19630261]], p-value=0.274061922485458
6, df_denom=1>
```

In [146]:
```
wald_2 # reject
```

Out[146]:
```
<class 'statsmodels.stats.contrast.ContrastResults'>
<Wald test (chi2): statistic=[[3.31770831]], p-value=0.0685373045788513
2, df_denom=1>
```

In [147]:
```
wald_3 # reject
```

Out[147]:
```
<class 'statsmodels.stats.contrast.ContrastResults'>
<Wald test (chi2): statistic=[[3.11846091]], p-value=0.0774098667994206
5, df_denom=1>
```

In [148]:
```
wald_4
```

Out[148]:
```
<class 'statsmodels.stats.contrast.ContrastResults'>
<Wald test (chi2): statistic=[[1.24441715]], p-value=0.264621457251622
2, df_denom=1>
```

In [45]:
```
logitmod6 = smf.logit(formula = 'y ~ treatment+drug+lengthxtreatment', d
ata = dat_log).fit()
print(logitmod6.summary2())
```

```
Optimization terminated successfully.
         Current function value: 0.650204
         Iterations 5
                         Results: Logit
=================================================================
Model:               Logit            Pseudo R-squared: 0.053
Dependent Variable:  y                AIC:              134.1395
Date:                2018-12-09 17:51 BIC:              144.4383
No. Observations:    97               Log-Likelihood:   -63.070
Df Model:            3                LL-Null:          -66.610
Df Residuals:        93               LLR p-value:      0.069362
Converged:           1.0000           Scale:            1.0000
No. Iterations:      5.0000
-----------------------------------------------------------------
                  Coef.  Std.Err.    z    P>|z|   [0.025  0.975]
-----------------------------------------------------------------
Intercept        -0.7960   0.3426 -2.3235 0.0202 -1.4675 -0.1245
treatment         0.0112   0.5403  0.0207 0.9835 -1.0479  1.0702
drug              0.7096   0.4510  1.5734 0.1156 -0.1743  1.5935
lengthxtreatment  0.9147   0.5829  1.5692 0.1166 -0.2278  2.0572
=================================================================
```

Means Imputation Exercise

In [50]:
```
# imputation of data for 4m to 8m
import numpy as np

# Imput bdi_4m & bdi_6m & bdi_8m

from sklearn.preprocessing import Imputer
df.head()
```

Out[50]:

	drug	length	treatment	bdi_pre	bdi_2m	bdi_4m	bdi_6m	bdi_8m	pre_response
1	No	>6m	TAU	29	2	2	NaN	NaN	3
2	Yes	>6m	BtheB	32	16	24	17	20	4
3	Yes	<6m	TAU	25	20	NaN	NaN	NaN	3
4	No	>6m	BtheB	21	17	16	10	9	3
5	Yes	>6m	BtheB	26	23	NaN	NaN	NaN	3

In [89]:
```
# Imputation method 1: Means substitution for 4 months, 6months, 8 month
s
y_4m = df[['bdi_4m']]
y_4m.values
imp = Imputer(missing_values='NaN', strategy='mean', axis=0)
y_4m_imputed = imp.fit_transform(y_4m)
dataframe_4y=pd.DataFrame(y_4m_imputed, columns=['y_4m_imputed'])

y_6m = df[['bdi_6m']]
y_6m.values
imp = Imputer(missing_values='NaN', strategy='mean', axis=0)
y_6m_imputed = imp.fit_transform(y_6m)
dataframe_6y=pd.DataFrame(y_6m_imputed, columns=['y_6m_imputed'])

y_8m = df[['bdi_8m']]
y_8m.values
imp = Imputer(missing_values='NaN', strategy='mean', axis=0)
y_8m_imputed = imp.fit_transform(y_8m)
dataframe_8y=pd.DataFrame(y_8m_imputed, columns=['y_8m_imputed'])
```

In [91]:
```
Imputation method 1: Means
dataframe_4y.index -= 1
df_imp = df.join(dataframe_4y)
df_imp

dataframe_6y.index += 1
df_imp = df_imp.join(dataframe_6y)

dataframe_8y.index += 1

df_imp = df_imp.join(dataframe_8y)
df_imp
```

Out[91]:

	bdi_pre	pre_response	y_4m_imputed	y_6m_imputed	y_8m_imputed
count	100.000000	100.00000	100.000000	100.000000	100.000000
mean	23.330000	2.83000	14.808219	12.758621	11.134615
std	10.840492	0.97499	10.080141	8.463004	6.678821
min	2.000000	1.00000	0.000000	0.000000	0.000000
25%	15.000000	2.00000	8.000000	8.750000	10.000000
50%	22.000000	3.00000	14.808219	12.758621	11.134615
75%	30.250000	4.00000	16.000000	12.758621	11.134615
max	49.000000	4.00000	53.000000	47.000000	40.000000

In [92]:
```
# Remove nulls from 4 months and compute the model
df_imp1 = df_imp[['drug', 'length', 'treatment', 'bdi_pre', 'pre_respons
e','y_4m_imputed', 'y_6m_imputed', 'y_8m_imputed']]
df_imp1 = df_imp1.dropna()
df_imp1.describe()


conditions = [
    (df_imp1['y_4m_imputed'] < 10),
    (df_imp1['y_4m_imputed'] < 19),
    (df_imp1['y_4m_imputed'] <30)]
choices = [1, 2, 3]
df_imp1['4m_response'] = np.select(conditions, choices, default=4)

conditions = [
    (df_imp1['y_6m_imputed'] < 10),
    (df_imp1['y_6m_imputed'] < 19),
    (df_imp1['y_6m_imputed'] <30)]
choices = [1, 2, 3]
df_imp1['6m_response'] = np.select(conditions, choices, default=4)

conditions = [
    (df_imp1['y_8m_imputed'] < 10),
    (df_imp1['y_8m_imputed'] < 19),
    (df_imp1['y_8m_imputed'] <30)]
choices = [1, 2, 3]
df_imp1['8m_response'] = np.select(conditions, choices, default=4)
```

In [95]:
```
df_imp1['4m_y'] = df_imp1['pre_response'] - df_imp1['4m_response']
df_imp1['6m_y'] = df_imp1['pre_response'] - df_imp1['6m_response']
df_imp1['8m_y'] = df_imp1['pre_response'] - df_imp1['8m_response']
df_imp1.head()
```

Out[95]:

	drug	length	treatment	bdi_pre	pre_response	y_4m_imputed	y_6m_imputed	y_8m_i
1	No	>6m	TAU	29	3	2.000000	12.758621	11.1346
2	Yes	>6m	BtheB	32	4	24.000000	17.000000	20.0000
3	Yes	<6m	TAU	25	3	14.808219	12.758621	11.1346
4	No	>6m	BtheB	21	3	16.000000	10.000000	9.00000
5	Yes	>6m	BtheB	26	3	14.808219	12.758621	11.1346

In [96]:
```
# Based on the difference, I will classify the response variable as 1 (i
f the treatement worked) or 0 if no change in treatment or gotten worse
def f(row):
    if row['4m_y']>0:
        val = 1
    else:
        val = 0
    return val
df_imp1['4m_y'] = df_imp1.apply(f, axis=1)

def f(row):
    if row['6m_y']>0:
        val = 1
    else:
        val = 0
    return val
df_imp1['6m_y'] = df_imp1.apply(f, axis=1)

def f(row):
    if row['8m_y']>0:
        val = 1
    else:
        val = 0
    return val
df_imp1['8m_y'] = df_imp1.apply(f, axis=1)
```

In [108]:
```
dat_4m = df_imp1[['drug', 'length', 'treatment','4m_y']]
dat_4m = pd.crosstab(index = dat_4m['treatment'], columns= dat_4m['4m_y'
], margins=True)
dat_4m['PropSucc'] = dat_4m[1]/dat_4m['All']
dat_4m
```

Out[108]:

4m_y	0	1	All	PropSucc
treatment				
BtheB	18	34	52	0.653846
TAU	21	27	48	0.562500
All	39	61	100	0.610000

In [114]:
```
dat_4m_drug = df_imp1[['drug', 'treatment','4m_y']]
piv1 = pd.pivot_table(dat_4m_drug, index = ['drug', 'treatment'], column
s = ['4m_y'], aggfunc = len)
piv1['propsucc'] = piv1[1]/(piv1[1] + piv1[0])
piv1
```

Out[114]:

	4m_y	0	1	propsucc
drug	**treatment**			
No	**BtheB**	7	15	0.681818
	TAU	17	17	0.500000
Yes	**BtheB**	11	19	0.633333
	TAU	4	10	0.714286

In [115]:
```
dat_4m_drug = df_imp1[['length', 'treatment','4m_y']]
piv2 = pd.pivot_table(dat_4m_drug, index = ['length', 'treatment'], colu
mns = ['4m_y'], aggfunc = len)
piv2['propsucc'] = piv2[1]/(piv2[1] + piv2[0])
piv2
```

Out[115]:

	4m_y	0	1	propsucc
length	**treatment**			
<6m	**BtheB**	10	16	0.615385
	TAU	8	15	0.652174
>6m	**BtheB**	8	18	0.692308
	TAU	13	12	0.480000

In [128]:
```
# No imputation for 4months
df_4m_no_imp = df[['drug', 'length', 'treatment', 'bdi_pre', 'pre_respon
se','bdi_4m']]
df_4m_no_imp = df_4m_no_imp.dropna()
df_4m_no_imp.head()
```

Out[128]:

	drug	length	treatment	bdi_pre	pre_response	bdi_4m
1	No	>6m	TAU	29	3	2
2	Yes	>6m	BtheB	32	4	24
4	No	>6m	BtheB	21	3	16
6	Yes	<6m	BtheB	7	1	0
7	Yes	<6m	TAU	17	2	7

In [129]:
```
df_4m_no_imp['bdi_4m'] = pd.to_numeric(df_4m_no_imp['bdi_4m'])
df_4m_no_imp['bdi_4m'].astype(str).astype(int)

conditions = [
    (df_4m_no_imp['bdi_4m'] < 10),
    (df_4m_no_imp['bdi_4m'] < 19),
    (df_4m_no_imp['bdi_4m'] <30)]
choices = [1, 2, 3]
df_4m_no_imp['4m_response'] = np.select(conditions, choices, default=4)

df_4m_no_imp['4m_y'] = df_4m_no_imp['pre_response'] - df_4m_no_imp['4m_r
esponse']

def f(row):
    if row['4m_y']>0:
        val = 1
    else:
        val = 0
    return val
df_4m_no_imp['4m_y'] = df_4m_no_imp.apply(f, axis=1)
```

In [130]:
```
# Even without imputation, BtheB program is still effective at 4 months
df_4m_no_imp1 = df_4m_no_imp[['drug', 'length', 'treatment','4m_y']]
df_4m_no_imp1 = pd.crosstab(index = df_4m_no_imp1['treatment'], columns=
 df_4m_no_imp1['4m_y'], margins=True)
df_4m_no_imp1['PropSucc'] = df_4m_no_imp1[1]/df_4m_no_imp1['All']
df_4m_no_imp1
```

Out[130]:

4m_y	0	1	All	PropSucc
treatment				
BtheB	12	25	37	0.675676
TAU	18	18	36	0.500000
All	30	43	73	0.589041

In [153]:
```
table = [[12,25],
         [18,18]]

# H0: BtheB is the same as TAU at 4 months
stat, p, dof, expected = chi2_contingency(table)
# interpret test-statistic
prob = 0.95
critical = chi2.ppf(prob, dof)
print('probability=%.3f, critical=%.3f, stat=%.3f' % (prob, critical, st
at))
if abs(stat) >= critical:
    print('Dependent (reject H0)')
else:
    print('Independent (fail to reject H0)')
```

```
probability=0.950, critical=3.841, stat=1.657
Independent (fail to reject H0)
```

In [132]:
```
# Imputation for 6 months:
dat_6m = df_imp1[['drug', 'length', 'treatment','6m_y']]
dat_6m = pd.crosstab(index = dat_6m['treatment'], columns= dat_6m['6m_y'
], margins=True)
dat_6m['PropSucc'] = dat_6m[1]/dat_6m['All']
dat_6m
```

Out[132]:

6m_y	0	1	All	PropSucc
treatment				
BtheB	17	35	52	0.673077
TAU	20	28	48	0.583333
All	37	63	100	0.630000

In [133]:
```
# Imputation for 8 months:
dat_8m = df_imp1[['drug', 'length', 'treatment','8m_y']]
dat_8m = pd.crosstab(index = dat_8m['treatment'], columns= dat_8m['8m_y'
], margins=True)
dat_8m['PropSucc'] = dat_8m[1]/dat_8m['All']
dat_8m
```

Out[133]:

8m_y	0	1	All	PropSucc
treatment				
BtheB	18	34	52	0.653846
TAU	14	34	48	0.708333
All	32	68	100	0.680000

In [134]:
```
# No imputation for 6months
df_6m_no_imp = df[['drug', 'length', 'treatment', 'bdi_pre', 'pre_respon
se','bdi_6m']]
df_6m_no_imp = df_6m_no_imp.dropna()
df_6m_no_imp.head()

df_6m_no_imp['bdi_6m'] = pd.to_numeric(df_6m_no_imp['bdi_6m'])
df_6m_no_imp['bdi_6m'].astype(str).astype(int)

conditions = [
    (df_6m_no_imp['bdi_6m'] < 10),
    (df_6m_no_imp['bdi_6m'] < 19),
    (df_6m_no_imp['bdi_6m'] <30)]
choices = [1, 2, 3]
df_6m_no_imp['6m_response'] = np.select(conditions, choices, default=4)

df_6m_no_imp['6m_y'] = df_6m_no_imp['pre_response'] - df_6m_no_imp['6m_r
esponse']

def f(row):
    if row['6m_y']>0:
        val = 1
    else:
        val = 0
    return val
df_6m_no_imp['6m_y'] = df_6m_no_imp.apply(f, axis=1)
```

In [135]:
```
# Even without imputation, BtheB program is still effective at 6 months
df_6m_no_imp1 = df_6m_no_imp[['drug', 'length', 'treatment','6m_y']]
df_6m_no_imp1 = pd.crosstab(index = df_6m_no_imp1['treatment'], columns=
 df_6m_no_imp1['6m_y'], margins=True)
df_6m_no_imp1['PropSucc'] = df_6m_no_imp1[1]/df_6m_no_imp1['All']
df_6m_no_imp1
```

Out[135]:

6m_y	0	1	All	PropSucc
treatment				
BtheB	9	20	29	0.689655
TAU	15	14	29	0.482759
All	24	34	58	0.586207

In [157]:
```
table = [[9,20],
        [15,14]]

# H0: BtheB is the same as TAU at 6 months
stat, p, dof, expected = chi2_contingency(table)
# interpret test-statistic
prob = 0.95
critical = chi2.ppf(prob, dof)
print('probability=%.3f, critical=%.3f, stat=%.3f' % (prob, critical, st
at))
if abs(stat) >= critical:
    print('Dependent (reject H0)')
else:
    print('Independent (fail to reject H0)')
```

```
probability=0.950, critical=3.841, stat=1.777
Independent (fail to reject H0)
```

In [136]:
```
# No imputation for 8months
df_8m_no_imp = df[['drug', 'length', 'treatment', 'bdi_pre', 'pre_respon
se','bdi_8m']]
df_8m_no_imp = df_8m_no_imp.dropna()
df_8m_no_imp.head()

df_8m_no_imp['bdi_8m'] = pd.to_numeric(df_8m_no_imp['bdi_8m'])
df_8m_no_imp['bdi_8m'].astype(str).astype(int)

conditions = [
    (df_8m_no_imp['bdi_8m'] < 10),
    (df_8m_no_imp['bdi_8m'] < 19),
    (df_8m_no_imp['bdi_8m'] <30)]
choices = [1, 2, 3]
df_8m_no_imp['8m_response'] = np.select(conditions, choices, default=4)

df_8m_no_imp['8m_y'] = df_8m_no_imp['pre_response'] - df_8m_no_imp['8m_r
esponse']

def f(row):
    if row['8m_y']>0:
        val = 1
    else:
        val = 0
    return val
df_8m_no_imp['8m_y'] = df_8m_no_imp.apply(f, axis=1)
```

In [137]:
```
# without imputation, BtheB program is not effective at 8 months
df_8m_no_imp1 = df_8m_no_imp[['drug', 'length', 'treatment','8m_y']]
df_8m_no_imp1 = pd.crosstab(index = df_8m_no_imp1['treatment'], columns=
 df_8m_no_imp1['8m_y'], margins=True)
df_8m_no_imp1['PropSucc'] = df_8m_no_imp1[1]/df_8m_no_imp1['All']
df_8m_no_imp1
```

Out[137]:

8m_y	0	1	All	PropSucc
treatment				
BtheB	8	19	27	0.703704
TAU	6	19	25	0.760000
All	14	38	52	0.730769

In [149]:
```
table = [[19,27],
        [19,25]]

# H0: BtheB is the same as TAU at 8 months
stat, p, dof, expected = chi2_contingency(table)
# interpret test-statistic
prob = 0.95
critical = chi2.ppf(prob, dof)
print('probability=%.3f, critical=%.3f, stat=%.3f' % (prob, critical, st
at))
if abs(stat) >= critical:
    print('Dependent (reject H0)')
else:
    print('Independent (fail to reject H0)')
```

```
probability=0.950, critical=3.841, stat=0.001
Independent (fail to reject H0)
```

Appendix B: Results file

Figure B-1: Percentage of nulls in the data by column

drug	0
length	0
treatment	0
bdi_pre	0
bdi_2m	3
bdi_4m	27
bdi_6m	42
bdi_8m	48

Figure B-2: Frequency of categories

			bdi_pre
			count
drug	length	treatment	
No	<6m	BtheB	9
		TAU	15
	>6m	BtheB	13
		TAU	19
Yes	<6m	BtheB	17
		TAU	8
	>6m	BtheB	13
		TAU	6

Figure B-3: Correlation between explanatory variables

	treatment	drug	length
treatment	1.000000	0.312265	-0.055484
drug	0.312265	1.000000	-0.128440
length	-0.055484	-0.128440	1.000000

Figure B-4: Proportion of Successes by Treatment Type

2m_y	0	1	All	PropSucc
treatment				
BtheB	25	27	52	0.519231
TAU	29	16	45	0.355556
All	54	43	97	0.443299

Figure B-5: Proportion of Successes by Treatment Type and Length

	2m_y	0	1	propsucc
length	treatment			
<6m	BtheB	15	11	0.423077
	TAU	11	9	0.450000
>6m	BtheB	10	16	0.615385
	TAU	18	7	0.280000

Figure B-6: Proportion of Successes by Treatment Type and Drug

	2m_y	0	1	propsucc
drug	treatment			
No	BtheB	11	11	0.500000
	TAU	24	9	0.272727
Yes	BtheB	14	16	0.533333
	TAU	5	7	0.583333

Figure B-7: Proportion of Successes by Treatment Type, Drug and Length

		2m_y	0	1	propsucc
drug	length	treatment			
No	<6m	BtheB	5	4	0.444444
		TAU	8	6	0.428571
	>6m	BtheB	6	7	0.538462
		TAU	16	3	0.157895
Yes	<6m	BtheB	10	7	0.411765
		TAU	3	3	0.500000
	>6m	BtheB	4	9	0.692308
		TAU	2	4	0.666667

Figure B-8: Summary Statistic of Logistic Model with treatment type variable

```
Optimization terminated successfully.
         Current function value: 0.673113
         Iterations 4
                           Logit Regression Results
==============================================================================
Dep. Variable:                      y   No. Observations:                   97
Model:                          Logit   Df Residuals:                       95
Method:                           MLE   Df Model:                            1
Date:                Sat, 08 Dec 2018   Pseudo R-squ.:                 0.01979
Time:                        18:19:25   Log-Likelihood:                -65.292
converged:                       True   LL-Null:                       -66.610
                                        LLR p-value:                    0.1044
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept     -0.5947      0.311     -1.910      0.056      -1.205       0.016
treatment      0.6717      0.417      1.610      0.107      -0.146       1.489
==============================================================================
```

Figure B-9: Summary Statistic of Logistic Model with treatment type, drug and length variables

```
Optimization terminated successfully.
         Current function value: 0.662337
         Iterations 4
                         Results: Logit
================================================================
Model:              Logit            Pseudo R-squared: 0.035
Dependent Variable: y                AIC:              136.4934
Date:               2018-12-08 18:19 BIC:              146.7922
No. Observations:   97               Log-Likelihood:   -64.247
Df Model:           3                LL-Null:          -66.610
Df Residuals:       93               LLR p-value:      0.19291
Converged:          1.0000           Scale:            1.0000
No. Iterations:     4.0000
----------------------------------------------------------------
                 Coef.  Std.Err.    z    P>|z|   [0.025  0.975]
----------------------------------------------------------------
Intercept       -0.8728   0.4264 -2.0468 0.0407 -1.7086 -0.0370
length[T.>6m]    0.1804   0.4243  0.4250 0.6708 -0.6513  1.0120
drug[T.Yes]      0.6282   0.4425  1.4197 0.1557 -0.2391  1.4954
treatment        0.4984   0.4394  1.1343 0.2567 -0.3628  1.3596
================================================================
```

Figure B-10: Summary Statistic of Logistic Model with treatment type and length variables

```
Optimization terminated successfully.
         Current function value: 0.655905
         Iterations 5
                         Results: Logit
================================================================
Model:              Logit            Pseudo R-squared: 0.045
Dependent Variable: y                AIC:              135.2455
Date:               2018-12-08 22:55 BIC:              145.5444
No. Observations:   97               Log-Likelihood:   -63.623
Df Model:           3                LL-Null:          -66.610
Df Residuals:       93               LLR p-value:      0.11284
Converged:          1.0000           Scale:            1.0000
No. Iterations:     5.0000
----------------------------------------------------------------
                  Coef.  Std.Err.    z    P>|z|   [0.025 0.975]
----------------------------------------------------------------
Intercept        -0.2007   0.4495 -0.4465 0.6553 -1.0816 0.6803
treatment        -0.1095   0.5997 -0.1826 0.8551 -1.2848 1.0658
length           -0.7438   0.6328 -1.1754 0.2398 -1.9841 0.4965
treatment:length  1.5239   0.8488  1.7954 0.0726 -0.1397 3.1876
================================================================
```

Figure B-11: Summary Statistic of Logistic Model with treatment type and drug variables

```
Optimization terminated successfully.
         Current function value: 0.654265
         Iterations 5
                         Results: Logit
================================================================
Model:              Logit            Pseudo R-squared: 0.047
Dependent Variable: y                AIC:              134.9274
Date:               2018-12-08 22:54 BIC:              145.2262
No. Observations:   97               Log-Likelihood:   -63.464
Df Model:           3                LL-Null:          -66.610
Df Residuals:       93               LLR p-value:      0.098191
Converged:          1.0000           Scale:            1.0000
No. Iterations:     5.0000
----------------------------------------------------------------
                 Coef.  Std.Err.    z    P>|z|   [0.025  0.975]
----------------------------------------------------------------
Intercept       -0.9808   0.3909 -2.5094 0.0121 -1.7469 -0.2147
treatment        0.9808   0.5784  1.6956 0.0900 -0.1529  2.1146
drug             1.3173   0.7040  1.8711 0.0613 -0.0625  2.6971
treatment:drug  -1.1838   0.9008 -1.3142 0.1888 -2.9492  0.5817
================================================================
```

Figure B-12: Summary Statistic of Logistic Model with treatment type, drug, length and interaction variables

```
Optimization terminated successfully.
         Current function value: 0.637191
         Iterations 5
                         Results: Logit
================================================================
Model:              Logit            Pseudo R-squared: 0.072
Dependent Variable: y                AIC:              135.6151
Date:               2018-12-09 00:21 BIC:              151.0634
No. Observations:   97               Log-Likelihood:   -61.808
Df Model:           5                LL-Null:          -66.610
Df Residuals:       91               LLR p-value:      0.087221
Converged:          1.0000           Scale:            1.0000
No. Iterations:     5.0000
----------------------------------------------------------------
                 Coef.  Std.Err.    z    P>|z|   [0.025 0.975]
----------------------------------------------------------------
Intercept       -0.5940   0.5140 -1.1556 0.2479 -1.6015 0.4135
treatment        0.1040   0.7568  0.1374 0.8907 -1.3792 1.5873
length          -0.7214   0.6596 -1.0938 0.2741 -2.0143 0.5714
treatment:length 1.5471   0.8761  1.7659 0.0774 -0.1700 3.2643
drug             1.3021   0.7149  1.8215 0.0685 -0.0990 2.7033
treatment:drug  -1.0293   0.9227 -1.1155 0.2646 -2.8376 0.7791
================================================================
```

Figure B-13: Summary Statistic of Logistic Model with treatment type, drug, length and interaction variables

```
Optimization terminated successfully.
         Current function value: 0.650204
         Iterations 5
                         Results: Logit
================================================================
Model:                Logit            Pseudo R-squared: 0.053
Dependent Variable:   y                AIC:              134.1395
Date:                 2018-12-09 17:51 BIC:              144.4383
No. Observations:     97               Log-Likelihood:   -63.070
Df Model:             3                LL-Null:          -66.610
Df Residuals:         93               LLR p-value:      0.069362
Converged:            1.0000           Scale:            1.0000
No. Iterations:       5.0000
----------------------------------------------------------------
                  Coef.  Std.Err.    z    P>|z|   [0.025  0.975]
----------------------------------------------------------------
Intercept        -0.7960   0.3426 -2.3235 0.0202 -1.4675 -0.1245
treatment         0.0112   0.5403  0.0207 0.9835 -1.0479  1.0702
drug              0.7096   0.4510  1.5734 0.1156 -0.1743  1.5935
lengthxtreatment  0.9147   0.5829  1.5692 0.1166 -0.2278  2.0572
================================================================
```

Figure B-14: Wald test results

```
hypothesis_1 = '(length = 0)'
hypothesis_2 = '(drug = 0)'
hypothesis_3 = '(lengthxtreatment = 0)'
hypothesis_4 = '(treatmentxdrug = 0)'

wald_1 = results.wald_test(hypothesis_1)
wald_2 = results.wald_test(hypothesis_2)
wald_3 = results.wald_test(hypothesis_3)
wald_4 = results.wald_test(hypothesis_4)
```

```
wald_1
```

```
<class 'statsmodels.stats.contrast.ContrastResults'>
<Wald test (chi2): statistic=[[1.19630261]], p-value=0.2740619224854586, df_denom=1>
```

```
wald_2 # reject
```

```
<class 'statsmodels.stats.contrast.ContrastResults'>
<Wald test (chi2): statistic=[[3.31770831]], p-value=0.06853730457885132, df_denom=1>
```

```
wald_3 # reject
```

```
<class 'statsmodels.stats.contrast.ContrastResults'>
<Wald test (chi2): statistic=[[3.11846091]], p-value=0.07740986679942065, df_denom=1>
```

```
wald_4
```

```
<class 'statsmodels.stats.contrast.ContrastResults'>
<Wald test (chi2): statistic=[[1.24441715]], p-value=0.2646214572516222, df_denom=1>
```

Figure B-15: 4month No Imputation Results

4m_y	0	1	All	PropSucc
treatment				
BtheB	12	25	37	0.675676
TAU	18	18	36	0.500000
All	30	43	73	0.589041

Figure B-16: 4month Mean Imputation Results

4m_y	0	1	All	PropSucc
treatment				
BtheB	18	34	52	0.653846
TAU	21	27	48	0.562500
All	39	61	100	0.610000

Figure B-17: 4month Chi-squared results (H0: BtheB = TAU)

```
probability=0.950, critical=3.841, stat=1.657
Independent (fail to reject H0)
```

Figure B-18: 6month No Imputation Results

6m_y	0	1	All	PropSucc
treatment				
BtheB	9	20	29	0.689655
TAU	15	14	29	0.482759
All	24	34	58	0.586207

Figure B-19: 6month Mean Imputation Results

6m_y	0	1	All	PropSucc
treatment				
BtheB	17	35	52	0.673077
TAU	20	28	48	0.583333
All	37	63	100	0.630000

Figure B-20: 6month Chi-squared results (H0: BtheB = TAU)

```
probability=0.950, critical=3.841, stat=1.777
Independent (fail to reject H0)
```

Figure B-21: 8month No Imputation Results

8m_y	0	1	All	PropSucc
treatment				
BtheB	8	19	27	0.703704
TAU	6	19	25	0.760000
All	14	38	52	0.730769

Figure B-22: 8month Mean Imputation Results

8m_y	0	1	All	PropSucc
treatment				
BtheB	18	34	52	0.653846
TAU	14	34	48	0.708333
All	32	68	100	0.680000

Figure B-23: 8month Chi-squared results (H0: BtheB = TAU)

```
probability=0.950, critical=3.841, stat=0.001
Independent (fail to reject H0)
```