Customer_life_time_prediction_using_BGNBD model

September 13, 2019

0.1 BG/NBD model for customer lifetime prediction

0.1.1 Background

In this data exercise, I am going to study a dataset collected from an ecommercce store that sells widgets. The main goal is to study customer purchasing behavior and forecast future purchasing from the transaction history.

[2]:
```
import numpy as np
import pandas as pd
import datetime as dt
import matplotlib as mpl
import matplotlib.pyplot as plt
import seaborn as sns
import pandas_profiling as pp

import warnings
warnings.filterwarnings('ignore')
```

[3]:
```
#plots setting
mpl.style.use('classic')
plt.rcParams['figure.facecolor'] = 'white'
```

[4]:
```
#load the dataset
df = pd.read_csv('data-science-exercise-data.csv')
```

0.1.2 Exploratory data analysis

[4]:
```
pp.ProfileReport(df)
```

```
<IPython.core.display.HTML object>
```

[4]:

From this profile report, I found that the dataset is quite clean with no missing row. The only thing needs to be done before modeling is to convert Timestamp from categorical type to datetime.

[5]:
```
#convert date value to datetime
df['Timestamp'] = pd.to_datetime(df['Timestamp'],format = "%Y-%m-%dT%H:%M:%SZ")
```

```python
df['Date'] = pd.to_datetime(df['Timestamp']).dt.date
#order by date
df = df.sort_values(by = ['Date'])
df = df[['CustomerID','Date','PurchaseValue']]
```

[6]:
```python
df.head()
```

[6]:

	CustomerID	Date	PurchaseValue
37874	71b13293ac280bf4f8d907d9fc19dc99	2016-11-30	293.47
17643	20e7347e8b299041a4387bd43247bcb3	2016-11-30	118.76
17613	01631c072a1105eddbcd7b853f048b08	2016-11-30	118.61
34015	42fda047f58fe65b0ccecc973c614704	2016-11-30	236.96
17600	2f25bb95cb8e61bed6fda5bc6b65861b	2016-11-30	118.53

[7]:
```python
# get the end of data collection
max(df['Date'])
```

[7]:
```
datetime.date(2017, 12, 6)
```

0.1.3 Implementing the modified BG model

BG-NBD model for modeling customer purchasing behavior BG-NBD (Beta Geometric Negative Binomial Distribution) model was proposed by Fader et al in 2005 to describe customer repeat purchases. The intuition for this model is: customers will make purchases at an randomly distributed time interval. After each purchase, they will have certain chance of being inactive. Here, being inactive means never purchasing again. The model has five assumptions: 1. While active, the number of transactions made by a customer follows a Poisson process with transaction rate λ. 2. Heterogeneity in λ follows a gamma distribution. 3. After any transaction, a customer becomes inactive with probability p. 4. Heterogeneity in p follows a beta distribution.
5. The transaction rate λ and the dropout probability p vary independently across customers.

To implement the model, it requires the following components for each customer. 1. Recency: When was the most recent purchase
2. Frequency: the number of repeat purchases 3. Monetary Value: How much money a customer spends on purchases.

These three values together is called RFM Matrix.

For this exercise, I will use the BG/NBD model in lifetime on the dataset directly. The process will be: 1. Split the dataset into training set and validation set. The recommended way for splitting is training period > 3x inter-purchase time and validation period > 1/2* training period. So, here the traning period runs from 2016-11-30 to 2017-07-30. The following period spanning 2017-07-31 to 2017-12-06 is used to evaluate a model's out-of-sample performance. 2. Transform the dataset into RFM matrix. 3. Build the model. 4. Evaluate the performance on validation set.

[15]:
```python
import lifetimes
from lifetimes.utils import summary_data_from_transaction_data
from lifetimes import BetaGeoFitter
from lifetimes.plotting import plot_period_transactions
from lifetimes.utils import calibration_and_holdout_data
from lifetimes.plotting import plot_calibration_purchases_vs_holdout_purchases
from lifetimes.plotting import plot_frequency_recency_matrix
```

```
from lifetimes.plotting import plot_probability_alive_matrix
from lifetimes import GammaGammaFitter

from scipy.stats import beta
from scipy.stats import gamma
from scipy.stats import expon
```

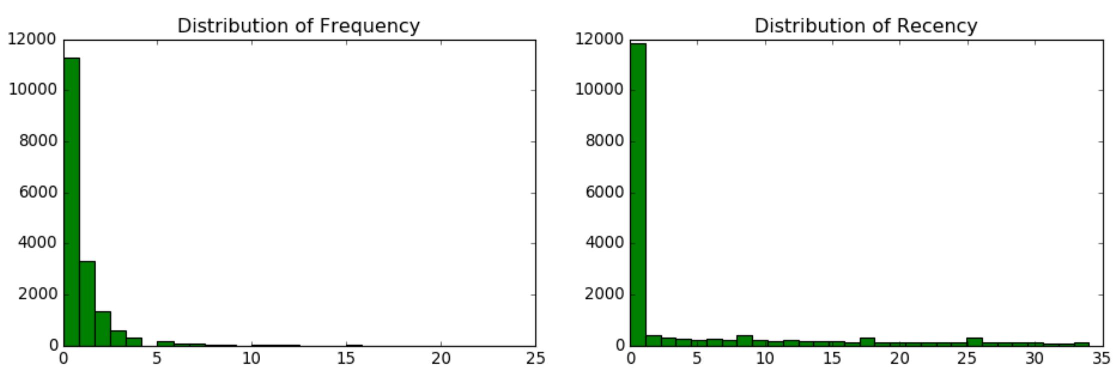
[16]:
```
#Transformation from a normal transaction list to RFM matrix.
rfm = summary_data_from_transaction_data(df,
                                         'CustomerID',
                                         'Date',
                                         monetary_value_col = 'PurchaseValue',
                                         observation_period_end = '2017-07-30',
                                         freq = 'W'
                                         )
```

[17]:
```
rfm.head(5)
```

[17]:
```
                                  frequency  recency     T  monetary_value
CustomerID
0001117ff1305c1fe840697166e61564        1.0      1.0  30.0         87.2800
00028502859fd7e111d88c20456b59d5        0.0      0.0  30.0          0.0000
0003f3458a6e7b495a975c2d9ddda559        0.0      0.0  30.0          0.0000
000784b838b807ad589d4bc69c0c562f        0.0      0.0  11.0          0.0000
000ad0f90e9fcb6ff5a0bc480cccbdb3        4.0     10.0  10.0        287.2275
```

Here, the column **T** means customer age, which is the end of our observation period minus out the period that they made their first purchase. Before modeling, I did some exploratory data analysis to get some basic ideas about the new dataset.

[18]:
```
#preliminary check-up on frequency and recency
plt.figure(figsize=(14,4))
plt.subplot(121)
plt.title('Distribution of Frequency')
temp = plt.hist(rfm['frequency'],30, facecolor='green')
plt.subplot(122)
plt.title('Distribution of Recency')
temp = plt.hist(rfm['recency'],30, facecolor='green')
plt.savefig('test1.png')
```



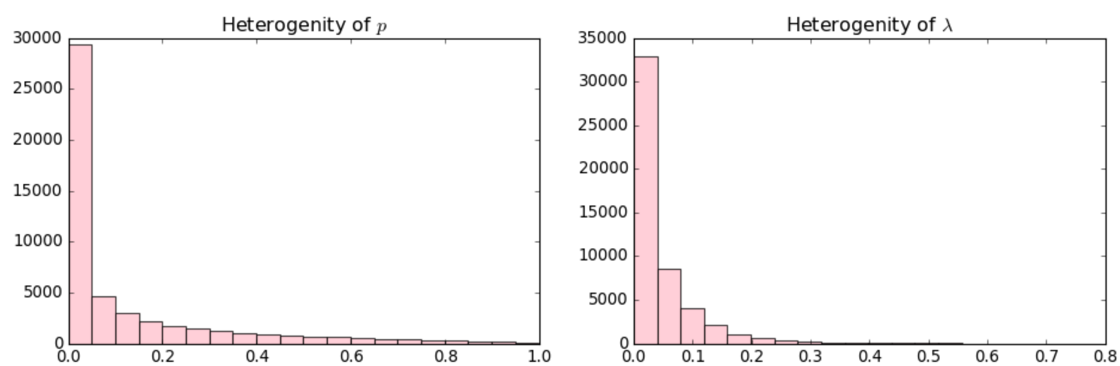
From the above two plots, both frequency and recency are distributed near 0. It means that most customers only made one purchase. Similiar for recency, most customers became inactive after making their last purchase.

[20]:
```
#bg model
bgf = BetaGeoFitter(penalizer_coef=0.0)
bgf.fit(rfm['frequency'], rfm['recency'], rfm['T'])
print(bgf)
```

```
<lifetimes.BetaGeoFitter: fitted with 17247 subjects, a: 0.22, alpha: 12.79, b:
1.54, r: 0.55>
```

In BG/NBD model, two assumptions for incorporating heterogenity of transaction rate λ and drop-out probability p are: λ follows a gamma distribution p follows a beta distribution. So, here I checked the heterogenity of these two parameters.

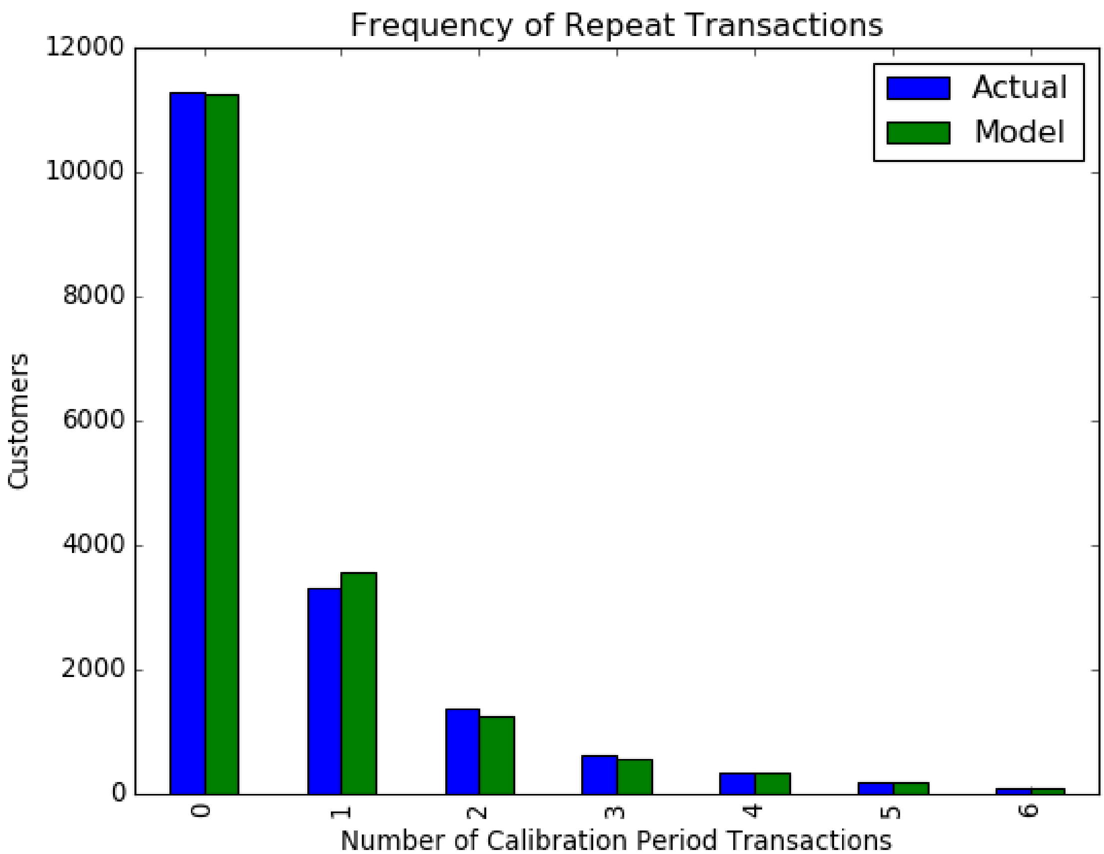
[21]:
```
#Heterogenity check
gbd = beta.rvs(bgf.params_['a'], bgf.params_['b'], size = 50000)
ggd = gamma.rvs(bgf.params_['r'], scale=1./bgf.params_['alpha'], size = 50000)
plt.figure(figsize=(14,4))
plt.subplot(121)
plt.title('Heterogenity of $p$')
temp = plt.hist(gbd, 20, facecolor='pink', alpha=0.75)
plt.subplot(122)
plt.title('Heterogenity of $\lambda$')
temp = plt.hist(ggd, 20, facecolor='pink', alpha=0.75)
plt.savefig('Heterogenity_test.png')
```



The two distributions are distributed around 0. For drop-out probability p, most customers have very low chance of being inactive after each purchase. For λ, it's mostly below 0.2.

For model evaluation, I compared the number of customers that are going to repeat purchase 0,1,2,3,4,5,and 6 times. As it's shown in the plot below, what the model predicted matches pretty well with what the actual numbers were.

[22]:
```
plot_period_transactions(bgf)
plt.savefig('FrequencyofRepeatTransactions.png')
```



For out-of-sample evaluation, the validation set also needs to be transformed into rfm first.

[23]:
```
#convert the hold-out dataset into rfm matrix

from lifetimes.utils import calibration_and_holdout_data

holdout_rfm = calibration_and_holdout_data(df,
                                          customer_id_col = 'CustomerID',
                                          datetime_col = 'Date',
                                          calibration_period_end = '2017-07-30',
                                          observation_period_end = '2017-12-06',
                                          freq = 'W'
                                          )
```

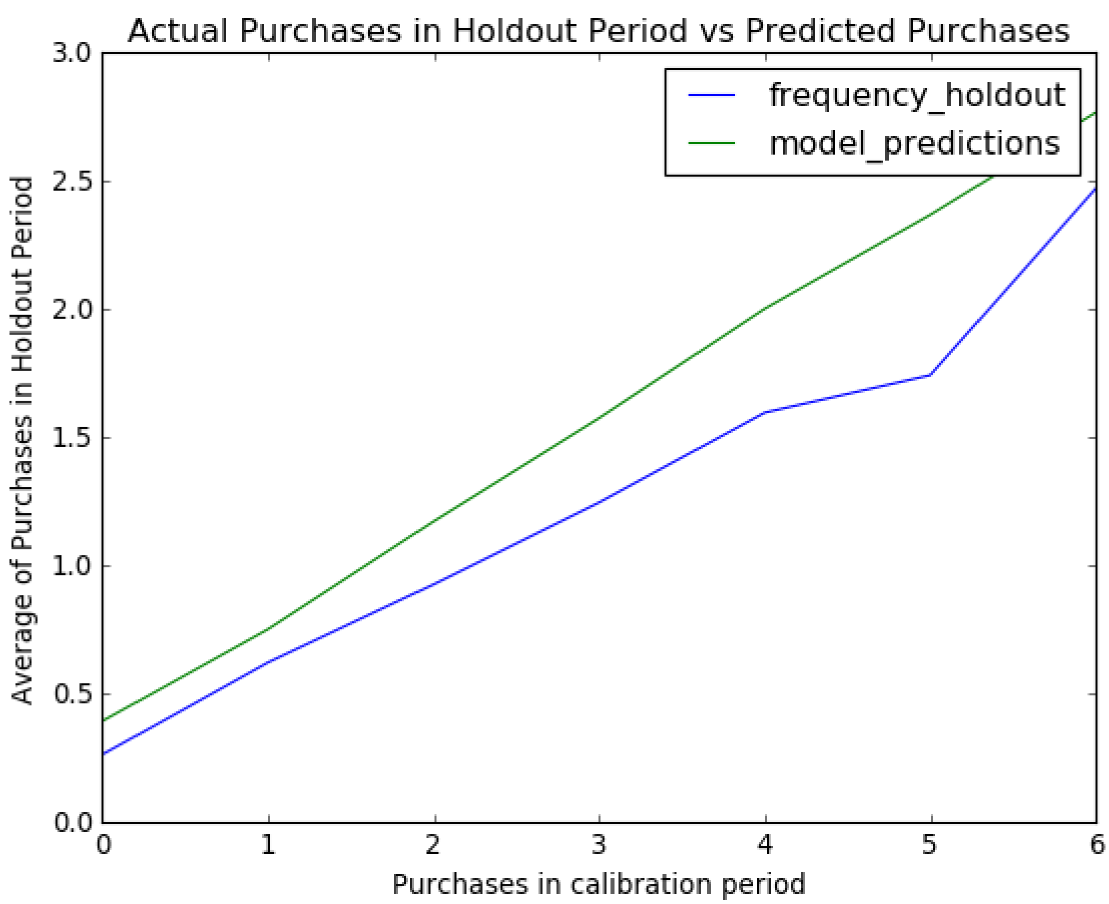
[24]:
```
holdout_rfm.head(5)
```

[24]:
```
                                  frequency_cal  recency_cal  T_cal  \
CustomerID
0001117ff1305c1fe840697166e61564            1.0          1.0   30.0
```

```
00028502859fd7e111d88c20456b59d5           0.0           0.0   30.0
0003f3458a6e7b495a975c2d9ddda559           0.0           0.0   30.0
000784b838b807ad589d4bc69c0c562f           0.0           0.0   11.0
000ad0f90e9fcb6ff5a0bc480cccbdb3           4.0          10.0   10.0

                                  frequency_holdout  duration_holdout
CustomerID
0001117ff1305c1fe840697166e61564                0.0                19
00028502859fd7e111d88c20456b59d5                0.0                19
0003f3458a6e7b495a975c2d9ddda559                1.0                19
000784b838b807ad589d4bc69c0c562f                0.0                19
000ad0f90e9fcb6ff5a0bc480cccbdb3                0.0                19
```

[29]:
```
#perform fitting on the _cal columns and test on the _holdout columns
bgf.fit(holdout_rfm['frequency_cal'], holdout_rfm['recency_cal'],␣
 ↪holdout_rfm['T_cal'])
plot_calibration_purchases_vs_holdout_purchases(bgf,holdout_rfm)
plt.savefig('ActualPurchaseVSPredictedPurchases.png')
```



The above plot groups all customers in the calibration period by their number of repeat purchases (x-axis) and then averages over their repeat purchases in the holdout period (y-axis). The green and blue line presents the model prediction and actual result of the y-axis respectively. As we can see, in general, the model overforecasts the number of purchases in the holdout period. Due to the lack of data for those customers purchasing many times, the model produces more errors and failed to catch those osciliations in the data.

0.1.4 Find the top 100 customers predicted to make the most purchases over the next 12 months

[18]:
```
#Here, I created an rfm matrix again on the whole dataset and use bgf to fit␣
↪the whole dataset
data = summary_data_from_transaction_data(df,
                                          'CustomerID',
                                          'Date',
                                          monetary_value_col = 'PurchaseValue',
                                          observation_period_end = '2017-12-06',
                                          freq = 'W'
                                          )
bgf.fit(data['frequency'], data['recency'], data['T'])
```

[18]:
```
<lifetimes.BetaGeoFitter: fitted with 23708 subjects, a: 0.38, alpha: 13.26, b:
2.75, r: 0.50>
```

[19]:
```
#forecasting period is 52 weeks (1 year)
t = 52
data['predicted_purchases'] = bgf.
↪conditional_expected_number_of_purchases_up_to_time(t, data['frequency'],␣
↪data['recency'], data['T'])
best_projected_cust = data.sort_values(by = ['predicted_purchases'],ascending =␣
↪False).head(100)
best_projected_cust[['predicted_purchases']]
```

[19]:
```
                                  predicted_purchases
CustomerID
a5fadc51b1ae844ad9a70ad3cfa46a64            30.143812
48a503edbaded96a3be27deee11967a1            22.323538
8d2ce54737dd404d20cadf1405d46dc8            19.265271
9f447f9415a380ac2eeee7df49c6ee7e            18.708444
5f01420f0edda6555df5ce1cc62b986c            17.065927
75fda9ea22086bf3814ff8c3f53de8ca            16.560979
a62a17bb46864da2c6da691d838971b3            16.441666
2ad9a83ee23110d8c2f4c01600b94f20            16.333001
a1c8d419a97af1f7152e21c0dddfcbce            16.216532
5ac5ed64cd99ed2a8403b7a927e644ef            14.716529
4f672b7c5d7a4214b08ec6906163c980            14.156900
a719d6643a7832535de9aded2f467825            14.039253
3b11478939967e896ae2619615650f97            14.001025
bb5d927e9f1aefa4db1881579e055a3e            13.011643
```

742d5a52d4df7cb14246d7f390de5d8a	12.900145
ca5cc4f526fcff10e85ce2685999e2c1	12.890864
a6d16066cc225139ddf01aa9fd8723aa	12.889464
16134915d822fe17588ae585935e1e81	12.832175
9cb19c3fc7311aae01cf16571b528001	12.736499
db0bd3b5ec2a35041de7436c699b215a	12.736499
f6a4f156c817ecb376d678f2aafdc570	12.655867
08cf1040da9bd693c2373478bc984dac	12.573083
cdf642859ae8d9a1e489c3a655cba827	12.476242
30aa99d3357244cf38ca04eadef1473a	12.164449
9690217261010ad90c43c1dd1d058e9c	11.871359
663b3df249f614305792ec7bf1dead30	11.756182
46454d378a1332750c086fb1101c07ce	11.746440
dca76db00cc59dfbcdcc97c8bbc7f9f1	11.732527
2b75f007b50d6b21e1501f47dc8d632b	11.663707
15e20f36220dd72101f937433465f328	11.600781
...	...
feeb6791f303fd34059c06d368f218e3	9.304552
4b669fc1c9dbc9a55e3faa7ad1b7ddc5	9.277344
cff00f3b67e59b8c9a89972c6cbdabf5	9.262861
51b3b64735d01118ff09427f8082277a	9.210227
5cde4c0e001f042b081bf18070c53f4e	9.207122
e3a48ef6277d4ee1f0dffe0b50cc9a99	9.205937
4278f21f39173ebb616faa118c22384a	9.134119
b75f1e1eee5b929de4c38cb7969662c8	9.060911
e68148a76bdfd39b8d99d878e5a8621e	9.004043
39d526e769cb1baa1dd29d05a3a4bdb2	8.986319
14526eaf59eef233af1e07b1a021f887	8.977002
7e70507512f04b9a1fc0594534443c8e	8.958066
9459310958c7741a71359ef3a151b9ed	8.957557
d0ad6b624dfc37784a755144be8c76a6	8.936879
a92534133444b5028d12a129b1b128f7	8.873109
52cc702cd8c70995cf6ba1c762e341d4	8.863842
e675875db3648ab4dbbd52768296425c	8.849578
a16bc4a9aab57a69d416c64ac77fc796	8.792156
0b9f48aef3295165238b3b14fffb981a	8.782164
23d45be9a8057f64e1103185e53e4f2a	8.768034
8770527c316c8732cec34377f12e9299	8.768034
950e9dd87391b872ae7b1afe1c778bfd	8.751185
7a04d32e6941f996df21c52d1ccbc7e8	8.751185
725d0a526feea495ab917ad7f8262765	8.630066
052a3c248db5edab01d32871d5acdbfa	8.630066
29505921638c03d201f08fd602dee9ab	8.612025
42f0d47cbd705d7da473b5d3e5cabc6c	8.571526
801266226172319918cca5963b492beb	8.542090
c0932084a28f1f941072feac6a4d4570	8.534502
790db2925ed67d6763fc0525aa7fa52e	8.520258

```
[100 rows x 1 columns]
```

The list above is the top 100 customers predicted to make the most purchases in the next year.

0.1.5 List the top 100 customers predicted to spend the most over the next 12 months

The above BG/NBD model is predicting the number of transactions and it doesn't take into account the economic value of each transaction. To estimate the transaction value, the Gamma-gamma model can be used here.

Here, I will only study customers with repeat purchases. Also, this model assumes that there is no relationship between the monetary value and the purchase frequency. So I first checked the correlation between these two features. From the correlation matrix below, we can see that there is almost no correlation between the monetary value and the number of purchases.

[20]:
```
# only look at customers with repeat purchases
ret_cust_data = data[data['frequency'] > 0]
#check correlations
ret_cust_data[['monetary_value','frequency']].corr()
```

[20]:

	monetary_value	frequency
monetary_value	1.000000	0.053254
frequency	0.053254	1.000000

[21]:
```
ret_cust_data.head(2)
```

[21]:
```
                                  frequency  recency     T  monetary_value  \
CustomerID
0001117ff1305c1fe840697166e61564        1.0      1.0  49.0           87.28
0003f3458a6e7b495a975c2d9ddda559        1.0     41.0  49.0           99.50

                                  predicted_purchases
CustomerID
0001117ff1305c1fe840697166e61564             0.510208
0003f3458a6e7b495a975c2d9ddda559             0.984358
```

[22]:
```
ggf = GammaGammaFitter(penalizer_coef = 0)
ggf.fit(ret_cust_data['frequency'], ret_cust_data['monetary_value'])
p,q,v = ggf._unload_params('p', 'q', 'v')
print (ggf)
```

```
<lifetimes.GammaGammaFitter: fitted with 9353 subjects, p: 4.36, q: 3.46, v:
139.52>
```

[23]:
```
# refit the BG model to the new dataset
bgf.fit(ret_cust_data['frequency'], ret_cust_data['recency'],␣
↪ret_cust_data['T'])
```

[23]:
```
<lifetimes.BetaGeoFitter: fitted with 9353 subjects, a: 0.35, alpha: 70.91, b:
0.86, r: 6.78>
```

[24]:
```python
#compute the total transaction for each customer
ggf_clv = ggf.customer_lifetime_value(
    bgf, #the model to use to predict the number of future transactions
    ret_cust_data['frequency'],
    ret_cust_data['recency'],
    ret_cust_data['T'],
    ret_cust_data['monetary_value'],
    time=52, # weeks
    freq = 'W',
    discount_rate = 0 #by default, this rate is 0.01
)
```

[25]:
```python
top100_transaction = ggf_clv.sort_values(ascending = False).head(100)
top100_transaction
```

[25]:
```
CustomerID
a5fadc51b1ae844ad9a70ad3cfa46a64    67227.628707
ca2202a96c2de6ca6b8a37a4a73fa730    36410.681025
dca76db00cc59dfbcdcc97c8bbc7f9f1    34032.837888
5ac5ed64cd99ed2a8403b7a927e644ef    30054.024599
60c19a709e3ced2d16d7100eb1069df5    28698.573392
089ecc49200cfe79584d0bec2a3cf8c0    25933.416352
98f8e41f45721cbe49a3147f6cf62432    24371.364230
42059a7ede026d409ff0f255635d7a08    24219.448276
cd4cb9ec252a085ed4d2d3af7c18280a    22117.158395
eba458987dc67827871c1d4d92e646e1    21921.001143
5f01420f0edda6555df5ce1cc62b986c    20458.151843
742d5a52d4df7cb14246d7f390de5d8a    20077.985133
66162981fc95e268e45bbfc738059687    19783.111445
ed2b4332b3ca253cfbb0ffb54d3f5ae0    19738.238711
24f05bfab01fef56ec049a828ebe20ab    19427.169965
f09ff1c6c4ac8ea95d8621a94bb325fd    17819.951371
a719d6643a7832535de9aded2f467825    17601.810759
48a503edbaded96a3be27deee11967a1    17111.442954
2f486887c2edb2571d32c8cd15301711    17081.441662
a92534133444b5028d12a129b1b128f7    16952.394980
2b75f007b50d6b21e1501f47dc8d632b    16857.766261
a62a17bb46864da2c6da691d838971b3    16838.574705
741bbe09f8795badb5292473bff42ead    16754.609048
ede2a476c3894cf65d1619987d148422    16737.996958
f2114d783824ed6c7c2658be58579e3e    16364.781659
fe403ffcf47b4efdf39874d181ae6da4    16258.899461
a63d0d4fe5bc662678bbbe6fbde1d900    16211.340527
9cb19c3fc7311aae01cf16571b528001    15975.342054
80b4fe892813996f469f44c28a0d1c10    15966.095177
30aa99d3357244cf38ca04eadef1473a    15844.230732
                                        ...
b783899d4ba9b328769e55d592fa8deb    11322.764168
```

```
b3aafd913eaf62e3156912378fd4b8a6    11246.774445
d6cef078357c829aed16490e7fee5aa2    11245.956269
51b3b64735d01118ff09427f8082277a    11244.875354
af77cebc5448b68b026556858d60c8cc    11079.909757
35bd91d013d04ffe65a66c2864be2c63    11031.406581
34bff14a781e6eabd03f59ac8f610ed7    11012.867392
ffe8da9e101a7d6c33b8c3be8eb705bc    10993.837181
5030e07233cb3cfb3c83c42bbb0e1e4f    10991.335255
607b2c3d2eb689b96ebf551fe735e203    10981.354829
16d851d4eb417ae73e48392063df8fb6    10967.551944
6f9ebbe87978c734ba71c0032e1f3e45    10873.707159
a5582b5136e2e597113e690c8a85d7b5    10865.806632
e0d591d2bbc656a278b5675d9bcae6c3    10817.237756
0368729f6f064e2f961cc22bfe5d60a1    10800.224804
0de9fe4a38ad31889dd8c2d0ded96a29    10786.825854
6061e84cb60705e4a2a378538353ba4d    10785.664930
cda380d6b9e87ecf02a85e994622131c    10760.343650
b6bc057a20765bb312cb740b32264fe4    10756.148014
bfd2558c6ca88693b5721952422b5b40    10745.604175
25e5fe3494dcb7d0de25fcd6f6b499d9    10687.111992
c718db0b1a517ec697119b0de9e2680e    10665.547154
a15d0cac4e30d80cc140913d01671552    10628.486839
cbb0220ede711ef61b2bbb667867bd85    10534.913540
8847105aa7e0a197252c8a942aff8779    10489.903263
ddbfaa20a84909a5a41f9e8f90b4d332    10469.881302
2143bad4fa8805114a00a73853ac6ace    10447.511904
d09642082305535d3c6192819a8c5a9b    10340.134382
52cc702cd8c70995cf6ba1c762e341d4    10325.136089
d6415e9e368a32ba39f756716c94e754    10285.595908
Name: clv, Length: 100, dtype: float64
```

0.1.6 Simulation

Write a simulation that shows how many customers are alive after 10 days, 1 year, 10 years and 100 years and how many purchases they have made in that time using the modified BG model. Use a simple random sample of 100 customers and show the result for 1 run of your simulation. We've outlined a possible approach below: * a. Generate a random sample of 100 customers. * b. Simulate how each customer makes purchases over time. * c. Count how many purchases the customers have made in 10 days. * d. Count how many customers are alive after 10 days. * e. Repeat b-d for 1 year, 10 year, 100 years.

The Simulation class below is written based on the assumption that all 100 customers in the sample are active when observation starts.

[32]:
```
class Simulation:
    def __init__(self,T,r,alpha,a,b,observation_period_end,size):
        self.T = T
        self.r = r
```

```
        self.alpha = alpha
        self.a = a
        self.b = b
        self.observation_period_end = pd.to_datetime(observation_period_end)
        self.size = size

    def simulate(self):
        self.num_active = self.size #assume all customers in the sample are␣
↪active when observation starts
        columns = ['CustomerID', 'Date'] #dataframe for generated 100 customers
        self.df = pd.DataFrame(columns=columns)

        self.T = self.T/7 #convert time intervals from days to weeks
        first_purchase = [self.observation_period_end - pd.Timedelta(self.T -␣
↪1, unit='W')] * self.size
        #This T is the customer purchase age, assuming that they all make their␣
↪first purchases on the first day
        T = self.T * np.ones(self.size)

        #the drop-out probability p follows the beta distrition
        drop_out_probability = beta.rvs(self.a, self.b, size=self.size)
        #transaction rate follows gamma distribution
        transaction_rate = gamma.rvs(self.r, scale=1. / self.alpha, size=self.
↪size)

        #loop for each customers in the sample
        for i in range(self.size):
            start_purchase = first_purchase[i]
            p = drop_out_probability[i]
            l = transaction_rate[i]
            age = T[i]

            purchases = [[i, start_purchase - pd.Timedelta(1, unit='W')]]
            next_purchase_in = expon.rvs(scale=1./l) #when is the next purchase
            active = True

            while next_purchase_in < age and active:
                purchases.append([i, start_purchase + pd.
↪Timedelta(next_purchase_in, unit='W')])
                next_purchase_in += expon.rvs(scale=1./l)
                active = np.random.random() > p

            self.df = self.df.append(pd.DataFrame(purchases, columns=columns))
            if not active:
                self.num_active -= 1

        self.df = self.df.reset_index(drop=True)
```

```
        return self
```

To run the simulation, here I used the parameteres from the above BG model fitter in the question 1.

[33]:
```
simulation1 = Simulation(T=10, r=0.50, alpha=13.26, a=0.38, b=2.75,␣
 ↪observation_period_end='2017-12-16', size=100)
simulation1.simulate()
print("After 10 days:")
print("Total number of customers alive:", simulation1.num_active)
print("Total number of purchases:", simulation1.df.shape[0])
```

```
After 10 days:
Total number of customers alive: 100
Total number of purchases: 105
```

[29]:
```
simulation2 = Simulation(T=365, r=0.50, alpha=13.26, a=0.38, b=2.75,␣
 ↪observation_period_end='2018-12-06', size=100)
simulation2.simulate()
print("After 1 year:")
print("Total number of customers alive:", simulation2.num_active)
print("Total number of purchases:", simulation2.df.shape[0])
```

```
After 1 year:
Total number of customers alive: 88
Total number of purchases: 237
```

[30]:
```
simulation3 = Simulation(T=3650, r=0.50, alpha=13.26, a=0.38, b=2.75,␣
 ↪observation_period_end='2027-12-06', size=100)
simulation3.simulate()
print("After 10 years:")
print("Total number of customers alive:", simulation3.num_active)
print("Total number of purchases:", simulation3.df.shape[0])
```

```
After 10 years:
Total number of customers alive: 63
Total number of purchases: 1361
```

[31]:
```
simulation4 = Simulation(T=36500, r=0.50, alpha=13.26, a=0.38, b=2.75,␣
 ↪observation_period_end='2117-12-06', size=100)
simulation4.simulate()
print("After 100 years:")
print("Total number of customers alive:", simulation4.num_active)
print("Total number of purchases:", simulation4.df.shape[0])
```

```
After 100 years:
Total number of customers alive: 28
Total number of purchases: 3408
```

This model is based on the assumption that the customers' behavior won't change after 1 year, 10 year and 100 year. Also, from the results, after 10 days, most of customers are still active, however, as time increases, the number of customers keeps decreasing. This is because the model doesn't consider those new customers. Also, for the last prediction, 100-year is longer than people's physical lifetime.

0.1.7 Conclusions

This model works well in some ways for this dataset: 1. The interpretability is good. Each of these variables (frequency, recency and customer age) is very easy to understand. So, it's easy to help managers to understand the model and have a more clear idea of their own level of customer value. 2. The model is simple and inexpensive. It can be used as benchmark study for later model improvement.

However, this model doesn't consider some covariates, such as marketing activity, which affect the customer behaviors. From the simulation test, I noticed that this model ignores the real customer life cycle. The estimated number of purchases is always decreasing with time because the model doesn't consider new customers. Also, one important purpose of this model is to identify those potential active customers. However, there are cases where some customers might click items' webpages but don't make any purchase. These group of potential customers are important for business as well. What's more, in this model, customer behavior information are all collapsed into RFM matrix. However, I think this matrix might be oversimplify the problem. Detailed purchase information, such as purchase frequency, and purchase amount can also reflect customer purchase behavior. There is little use of personalized information of each individual customer. Personalized information, such as gender, age, occupation, can also describe purchase behaviors.