Table of Contents

Overview of the dataset

The dataset contains 49945 transactions of 23708 different customers taken from 30th November 2016 to 6th December 2017. The first column uniquely identifies the customer, the second column has the date and time of the transaction and the last column has the amount a customer spent during that transaction.

Introduction about the library and the models

The lifetimes library is useful in predicting when a customer will make a purchase, when a person will come back and visit a website and most importantly predict the lifetime value of the customer. It consists of different types of fitters for this purpose like the BG model, modified BG model and the Pareto/NBD model. In this notebook, we are using the modified BG model to predict the customer behaviour.

The modified BG model has all the assumptions of the BG model except an additional assumption that a customer may drop out after the first purchase. The model assumes that after each transaction, the customer tosses a buy coin (not a fair coin) and depending on the result he chooses to continue his purchases or drop-out. The customer will stay alive after each purchase until the coin toss result is 'T'. Hence the customer's relationship with the firm is geometrically distributed (number of coin tosses). The customers probability of dropping out or dying can take a value between 0 and 1 and this is achieved by using a beta distribution.

The Gamma-Gamma model is used to calculate the economic value of each transaction. It gives us an understanding what a customer will be worth in a future time period.

In this dataset we see that the relationship between the customer and the ecommerce store is non-contractual. In a non-contractual setting, differentiating between those customers who have ended their relationship with the firm versus those who are in the middle of a break between transactions is very difficult.

Loading the data, viewing and checking the data types of columns

Loading the required libraries

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from lifetimes.utils import *
from lifetimes import ModifiedBetaGeoFitter
from lifetimes.plotting import plot_frequency_recency_matrix
from lifetimes.plotting import plot_probability_alive_matrix
from lifetimes.plotting import plot_period_transactions
from lifetimes import GammaGammaFitter
import random
from autograd.scipy.special import gamma
import warnings
warnings.filterwarnings('ignore')
import seaborn as sns
sns.set_palette("husl")
sns.set(rc={'image.cmap': 'coolwarm'})
%matplotlib inline
pd.options.display.float_format = '{:.4f}'.format
```

Loading the dataset

In [2]:
```
df = pd.read_csv('data-science-exercise-data.csv')
```

In [3]:
```
df.head()
```

Out[3]:

	CustomerID	Timestamp	PurchaseValue
0	5611860a94204a4078659f1667372cc4	2017-02-09T18:31:00Z	4.7200
1	07b260307114f9cffc5601eb7ad4b565	2017-06-28T19:26:00Z	5.3300
2	1ede55062d0d68c70fc4d355a5328d11	2017-07-28T17:47:00Z	5.3500
3	42b8e86e8da5e35628bcdc146c358665	2016-12-17T22:59:00Z	6.0000
4	1047f2787b7efc759d7ffcbc40ef7e19	2016-12-05T17:23:00Z	6.2800

In [4]:
```
df.shape
```

Out[4]:
```
(49945, 3)
```

In [5]:
```
df.dtypes
```

Out[5]:
```
CustomerID        object
Timestamp         object
PurchaseValue    float64
dtype: object
```

Converting 'date' column to datetime datatype

In [6]:
```
df['Timestamp'] = pd.to_datetime(df['Timestamp']).dt.date
```

In [7]:
```
df['Timestamp'].min()
```

Out[7]:
```
datetime.date(2016, 11, 30)
```

In [8]:
```
df['Timestamp'].max()
```

Out[8]:
```
datetime.date(2017, 12, 6)
```

Implement the modified BG model from the lifetimes package using the data we provide.

Creating a summary data from the given transaction data with a daily frequency for data aggregation

This will give us the RFM values of each customer.

Here,

Frequency : Number of repeat purchases a customer has made. If the frequency is 0, it means that the customer has made only 1 purchase.

Recency : It is the time between the customer's first purchase and last purchase. It tell the age of the customer when he did his most recent purchase.

T : It is the age of customer in the units of the dataset (here days). It is the time between the customer's first purchase and the end of observation time period.

In [9]:
```
data = summary_data_from_transaction_data(df, 'CustomerID', 'Timestamp', monetary_value_col='PurchaseValue', observation_period_end='2017-12-6', freq = 'D')
data.head(10)
```

Out[9]:

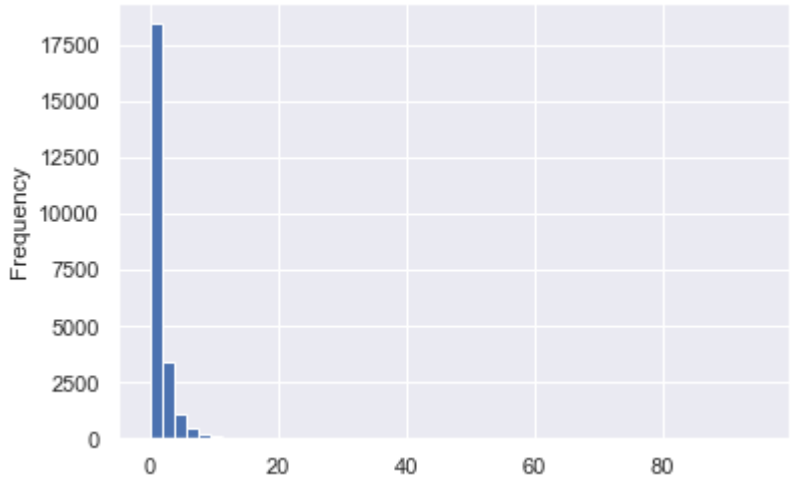
CustomerID	frequency	recency	T	monetary_value
0001117ff1305c1fe840697166e61564	1.0000	11.0000	344.0000	87.2800
00028502859fd7e111d88c20456b59d5	0.0000	0.0000	342.0000	0.0000
000306d200fd6e9b03de48d15348f5c2	0.0000	0.0000	33.0000	0.0000
0003f3458a6e7b495a975c2d9ddda559	2.0000	285.0000	343.0000	88.0650
000784b838b807ad589d4bc69c0c562f	0.0000	0.0000	210.0000	0.0000
0008e6b90a8f191089e8a0757fabf968	1.0000	69.0000	77.0000	145.5400
00090b9fa28029c4839ca0a0306b6b11	1.0000	61.0000	82.0000	349.4400
000ad0f90e9fcb6ff5a0bc480cccbdb3	5.0000	68.0000	200.0000	229.7820
000af852b3020e67ba4385e6477d4e58	0.0000	0.0000	353.0000	0.0000
000c6eab0c01eddd654c20fab4182dae	4.0000	165.0000	183.0000	119.2825

From the above result, we can see that the first customer with **Customer ID '0001117ff1305c1fe840697166e61564'** has made 2 purchases, i.e. he has just made **1 repeat purchase. So his frequency is 1. The time duration between the customer's first purchase and his latest purchase is 11 days.** T represents the customers age, i.e. the time between his first purchase to the end of observation period. Here the **customer we are considering has a age of 344 days.**

Plotting the histogram of frequency of purchases

In [10]:
```
data['frequency'].plot(kind='hist', bins = 50)
```

Out[10]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x146ad5ce860>
```



In [11]:
```
print(sum(data['frequency'] == 0)/float(len(data)))
```

```
0.5930909397671672
```

The above result shows that in our dataset, nearly 59.30% of the customers have made a purchase only once, i.e. their frequency is 0

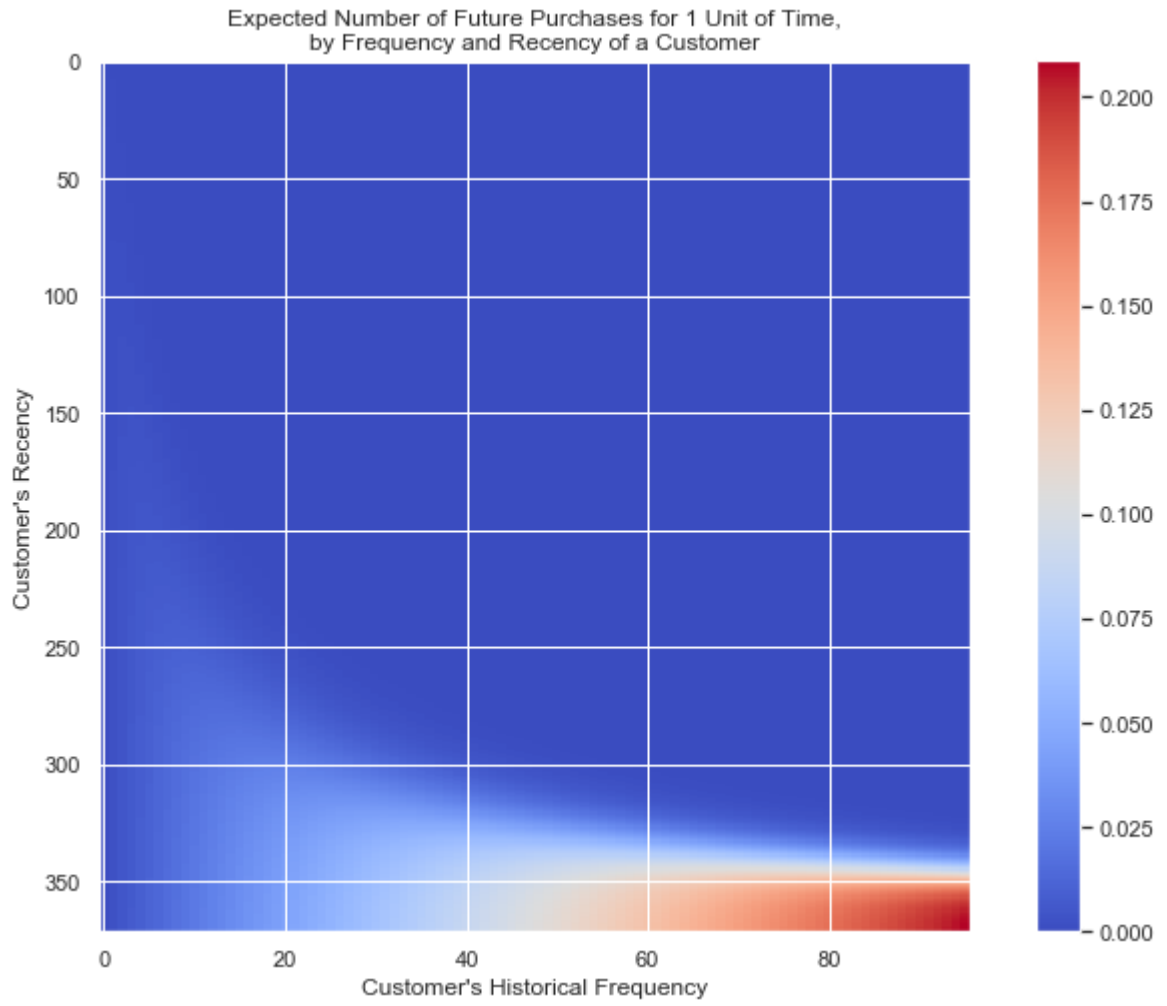
Fitting the data to Modified beta-geometric model

In [12]:
```
mbg = ModifiedBetaGeoFitter(penalizer_coef=0.0)
mbg.fit(data['frequency'], data['recency'], data['T'])
print(mbg)
```

```
<lifetimes.ModifiedBetaGeoFitter: fitted with 23708 subjects, a: 0.54, alpha:
85.27, b: 1.45, r: 0.79>
```

In [13]:
```
fig = plt.figure(figsize=(12,8))
plot_frequency_recency_matrix(mbg)
```

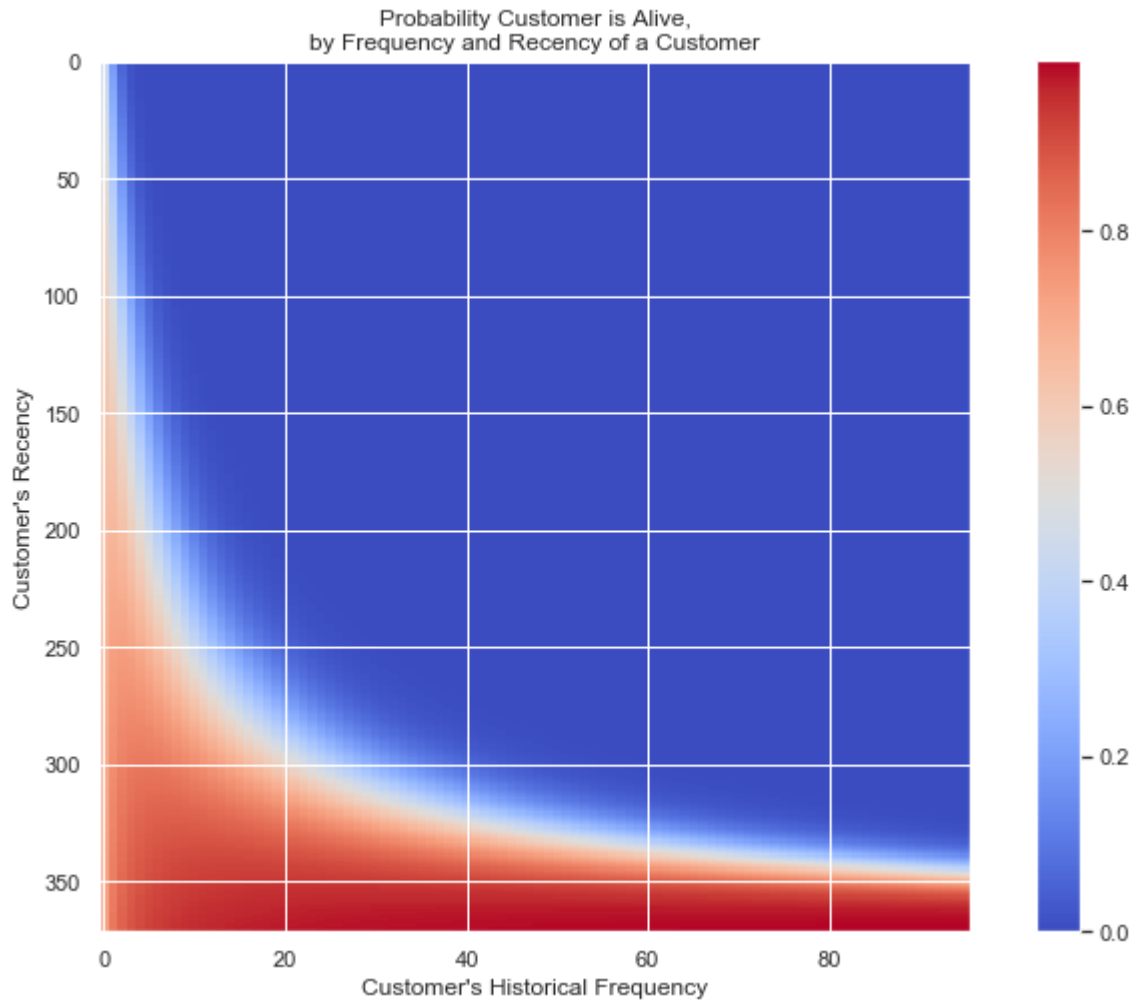
Out[13]: <matplotlib.axes._subplots.AxesSubplot at 0x146ad6230f0>



The above graph tells us that our best customer will be the one who has made more than 80-85 purchases and has a recency of more than 350 days. Customers who have purchased a lot but their recency is low, will probably have no future purchases, while the customers who have around 50 to 60 purchases and a recency of 325 to 350 days have a medium chance of buying in the future

In [14]:
```
fig = plt.figure(figsize=(12,8))
plot_probability_alive_matrix(mbg)
```

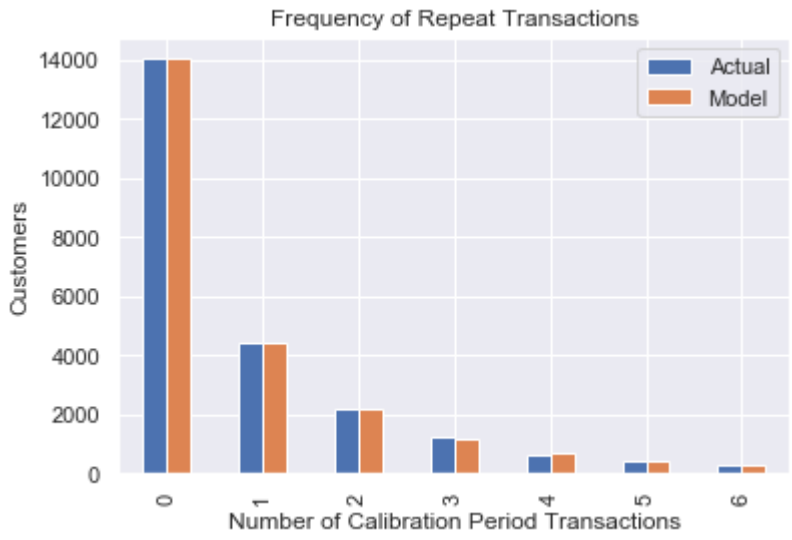
Out[14]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x146adb7b390>
```



The above graph tells us that the customers who have a recency of more than 350 days have a higher chance of being alive. As the recency goes on decreasing, the higher is the chance that the customer is dead or has dropped out

In [15]:
```
plot_period_transactions(mbg)
```

Out[15]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x146adcbb080>
```



The above graph shows that our model fits well, predicting closely to the actual observed data

List the 10 customers predicted to make the most purchases over the next 12 months.

As we have a granularity of days, we take 365 days as the time period. An important thing to note here is that if we take 12 months instead of 365 days, the results will differ as the in-built factor of converting months to days is of 30, which will make the number of days equal to 360 instead of 365

In [16]:
```
t = 365
predicted_purchases = mbg.conditional_expected_number_of_purchases_up_to_time(
t, data['frequency'], data['recency'], data['T'])
predicted_purchases = pd.DataFrame(predicted_purchases, columns = ['PredictedV
alues'])
```

Displaying the top 100 customers who are predicted to have the most purchases

In [17]:
```
pd.set_option('display.max_rows', None)
predicted_purchases.nlargest(10, 'PredictedValues')
```

Out[17]:

CustomerID	PredictedValues
a5fadc51b1ae844ad9a70ad3cfa46a64	64.4729
48a503edbaded96a3be27deee11967a1	30.9204
9f447f9415a380ac2eeee7df49c6ee7e	25.9509
5f01420f0edda6555df5ce1cc62b986c	25.0565
8d2ce54737dd404d20cadf1405d46dc8	22.9183
a62a17bb46864da2c6da691d838971b3	20.7534
3b11478939967e896ae2619615650f97	20.7343
2ad9a83ee23110d8c2f4c01600b94f20	19.9430
75fda9ea22086bf3814ff8c3f53de8ca	19.6025
30aa99d3357244cf38ca04eadef1473a	18.8912

List the 10 customers predicted to spend the most over the next 12 months.

To calculate this we use the Gamma-Gamma model. We filter out the customer who have no repeat purchases i.e. frequency is 0

In [18]:
```
frequency_filtered_customers = data[data['frequency'] > 0]

ggf = GammaGammaFitter(penalizer_coef = 0)
ggf.fit(frequency_filtered_customers['frequency'],
        frequency_filtered_customers['monetary_value'])
```

Out[18]:
```
<lifetimes.GammaGammaFitter: fitted with 9647 subjects, p: 4.57, q: 3.65, v:
135.88>
```

In [19]:
```
predicted_spend = ggf.customer_lifetime_value(
    mbg,
    data['frequency'],
    data['recency'],
    data['T'],
    data['monetary_value'],
    time = 365,
    freq = 'D'
    )
```

In [20]:
```
predicted_spend = pd.DataFrame(predicted_spend)
```

In [21]:
```
pd.set_option('display.max_rows', None)
predicted_spend.nlargest(10, 'clv')
```

Out[21]:

CustomerID	clv
a5fadc51b1ae844ad9a70ad3cfa46a64	119961.2341
ca2202a96c2de6ca6b8a37a4a73fa730	68488.2378
dca76db00cc59dfbcdcc97c8bbc7f9f1	51609.8739
5ac5ed64cd99ed2a8403b7a927e644ef	44989.8911
60c19a709e3ced2d16d7100eb1069df5	41079.6040
5f01420f0edda6555df5ce1cc62b986c	32223.4828
742d5a52d4df7cb14246d7f390de5d8a	31082.3446
a719d6643a7832535de9aded2f467825	30322.9973
ed2b4332b3ca253cfbb0ffb54d3f5ae0	29955.1282
eba458987dc67827871c1d4d92e646e1	29438.2539

Analysis

Assumptions for Modified Beta Gemetric Model:

Assumption 1 - While active, the number of transactions made by a customer follows a poisson process with transaction rate λ. This is equivalent to assuming that the time between transactions is distributed exponential with transaction rate λ, i.e.,

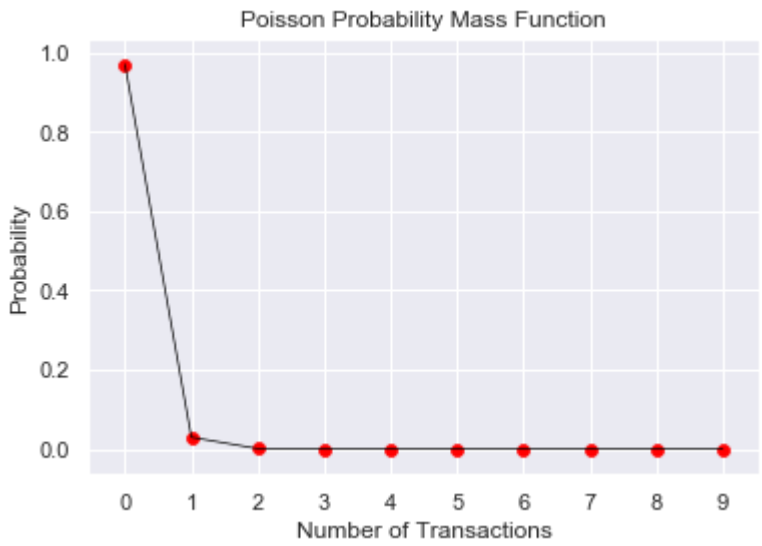
$$f(tj|tj-1;\lambda) = \lambda e - \lambda(tj - tj - 1), tj > tj - 1 \geq 0$$

Explanation - At every sub-period of a specific time interval, each customer tosses his buy coin and depending on the result, he purchases or not. The number of transactions (heads) we observe in the period depends on each customer's probability distribution around λ and not on his recent buying behavior.

In [22]:
```
from scipy.stats import poisson
poisson_lambda = 0.03
p_arr = []

distribution = poisson(poisson_lambda)
for transactions in range(0,10):
     p_arr.append(distribution.pmf(transactions))

plt.ylabel('Probability')
plt.xlabel('Number of Transactions')
plt.xticks(range(0, 10))
plt.title('Poisson Probability Mass Function')
plt.plot(p_arr, color='black', linewidth=0.7, zorder=1)
plt.scatter(range(0, 10), p_arr, color='red', linewidth=0.7)
plt.show()
```



Assumption 2 - The transaction rate λ and the dropout probability p vary independently across customers.

Explanation - This assumption captures an individual customer's chances of not purchasing again and the customer's rate of transaction. It states that it is different for each customer and does not depend on any other individual.

Assumption 3 - Heterogeneity in p follows a beta distribution, with parameters a and b.

$$f(p|a,b) = pa - 1(1-p)b - 1B(a,b), 0 \leq p \leq 1.$$

Heterogeneity in transaction rates among customers follows a Gamma distribution

$$f(\lambda|r,\alpha) = \alpha r \lambda r - 1e - \lambda\alpha\Gamma(r), \lambda > 0.$$

Explanation - To capture each customer's individual behaviour, we assume that each individual's probability of dropping out at the end of a purchase period can take on any one of an infinite number of possible values between 0 and 1, this is achieved by assuming that variability in these probabilities across customers is captured by a continuous probability distribution or a beta distribution.

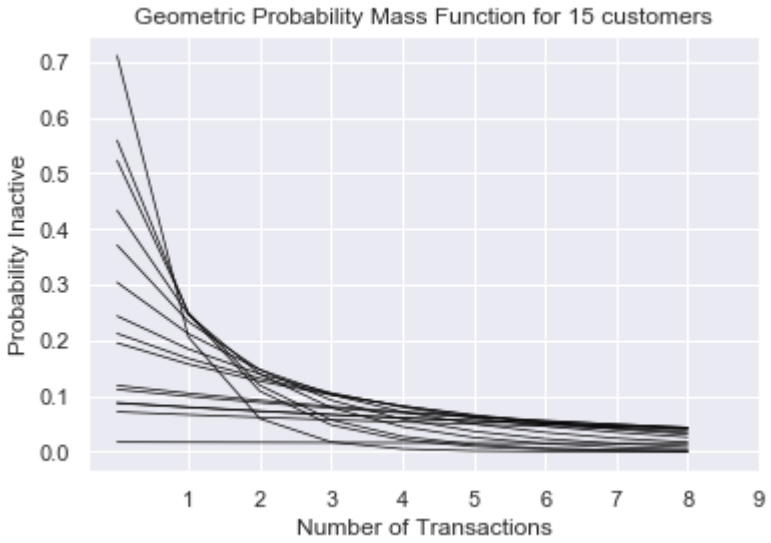
As Gamma distribution can take different shapes, it can capture the transactions for a customer, which can occur at any point in time, in a much better sense. The r controls the shape and skewness of distribution, while larger α leads to wider spread of the distribution.

In [23]:
```
beta_a = 0.54
beta_b = 1.45

for customer in range(0, 15):
    p_arr = []
    beta  = np.random.beta(a=beta_a, b=beta_b)
    for transaction in range(1,10):
        proba_inactive = beta*(1-beta)**(transaction-1)
        p_arr.append(proba_inactive)
    p_arr = np.array(p_arr)
    plt.plot(p_arr, color='black', linewidth=0.7)

plt.ylabel('Probability Inactive')
plt.xlabel('Number of Transactions')
plt.xticks(range(1, 10))
plt.title('Geometric Probability Mass Function for 15 customers')
plt.show
```

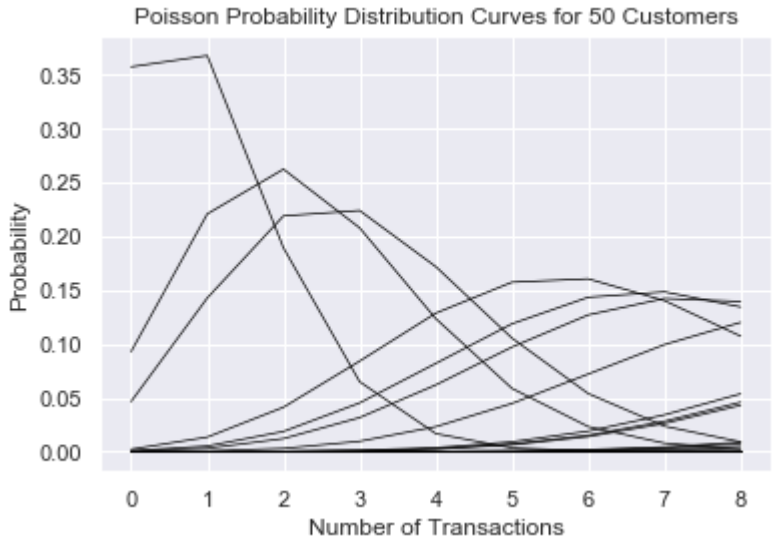
Out[23]:
```
<function matplotlib.pyplot.show(*args, **kw)>
```



In [24]:
```
gamma_shape = 0.79
gamma_scale = 85.27

for customer in range(0, 50):
    distribution = poisson(np.random.gamma(shape=gamma_shape, scale=gamma_scal
e))
    p_arr = []
    for transactions in range(0,9):
        p_arr.append(distribution.pmf(transactions))
    plt.plot(p_arr, color='black', linewidth=0.7)

plt.ylabel('Probability')
plt.xlabel('Number of Transactions')
plt.xticks(range(0,9))
plt.title('Poisson Probability Distribution Curves for 50 Customers')
plt.show()
```



Assumption 4 - After any transaction, a customer becomes inactive with probability p. Therefore the point at which the customer drops out is distributed across transactions according to a (shifted) geometric distribution.

P(inactive immediately after j^{th} transaction) $= p(1-p)j-1, j = 1, 2, 3, \ldots.$

Explanation - At the end of each period, a customer decides whether or not to drop-out. It is like saying that the customer tosses a coin to decide this, where T is the result when the customer will stop purchasing. The coin is not a fair coin and the customer continues his purchases till he gets a T as the result of coin toss.

Assumption 5 - A customer can drop out immediately after the first purchase

Explanation - An improvement over BG model, this assumption helps to make the model flexible. This assumption helps to identify the customers who just buy once and never return.

Assumptions for Gamma - Gamma Model:

Assumption 1 - For each individual customer, the profit per transaction is independent of the number of transactions

Assumption 2 - Expected profit per transaction does not vary over time

Assumption 3 - The monetary value of a customer's given transaction varies randomly around their average transaction value.

The application of the MBG/NBD model in an ecommerce store environment shows that the model captures a customer's purchase and dropping out process in an accurate way. These assumptions appear to be valid given that they are able to forecast the future purchase patterns of each customer. Also by using modified beta geometric model, we are able to capture the additional chance of dropout at time zero which particularly matches the ecommerce purchase behavior.

Simulation:

a. Generate a random sample of 100 customers.

Creating a sample data of 100 unique customers

In [25]:
```
random_list_customers = random.sample(list(df['CustomerID'].unique()), 100)
```

In [26]:
```
sample_100_data = df[df['CustomerID'].isin(random_list_customers)]
sample_100_data['CustomerID'].nunique()
```

Out[26]: 100

In [27]:
```
sample_100_data.head()
```

Out[27]:

	CustomerID	Timestamp	PurchaseValue
590	4e13563000896200814ce5345f6ff872	2017-09-15	34.1200
923	270a28f8a0e2cf6682178cad1595951e	2017-07-15	37.9800
1051	270a28f8a0e2cf6682178cad1595951e	2017-09-13	39.0500
1257	0d6b2ec49235fc36e18c2b27d5b19a26	2017-06-27	40.6600
1382	270a28f8a0e2cf6682178cad1595951e	2017-07-26	41.9400

In [28]:
```
sample_100_data.shape
```

Out[28]:
```
(194, 3)
```

b. Simulate how each customer makes purchases over time.

Creating a summary data from the given transaction data

In [29]:
```
data_simulated = summary_data_from_transaction_data(sample_100_data, 'Customer
ID', 'Timestamp',
                                                    monetary_value_col='PurchaseValu
e', observation_period_end = '2017-12-6')
```

In [30]:
```
data_simulated.head()
```

Out[30]:

CustomerID	frequency	recency	T	monetary_value
09d44a89072bf545685f385f6a9a6c48	0.0000	0.0000	48.0000	0.0000
0d6b2ec49235fc36e18c2b27d5b19a26	11.0000	336.0000	351.0000	300.2391
0ed99c98366ca00e18be722dfb3c38c6	0.0000	0.0000	319.0000	0.0000
103229e2eb1b03aac9a7be5d09452e8e	6.0000	185.0000	334.0000	176.9050
140ac4af2dc831ff8fe169b88f081bcb	0.0000	0.0000	349.0000	0.0000

In [31]:
```
mbgs = ModifiedBetaGeoFitter(penalizer_coef=0.0)

mbgs.fit(data_simulated['frequency'], data_simulated['recency'], data_simulate
d['T'])
```

Out[31]:
```
<lifetimes.ModifiedBetaGeoFitter: fitted with 100 subjects, a: 0.00, alpha: 1
14.40, b: 0.63, r: 0.47>
```

c. Count how many purchases the customers have made.

Creating a function to estimate the the purchases a customer would have made in a future time 't':

In [32]:
```
def future_purchases(t):

    predicted_purchases = mbgs.conditional_expected_number_of_purchases_up_to_
time(t, data_simulated['frequency'],

data_simulated['recency'], data_simulated['T'])

    predicted_purchases = pd.DataFrame(predicted_purchases, columns = ['Predic
tedValues'])

    return predicted_purchases
```

d. Count how many customers are alive after 10 days.

From *Batislam, E. M., Denizel, M., & Filiztekin, A. (2007)*, we get the formula for predicting the probability that the customer is alive at time 't' as:

$$P(Active|x, t_x, t, r, \alpha, a, b) = \frac{1}{1 + \frac{\Gamma(a+1)*\Gamma(b+x)}{\Gamma(a)*\Gamma(b+x+1)}(\frac{\alpha+t}{\alpha+t_x})^{r+x}}$$

We get the values for r, alpha, a, b from the following:

In [33]:
```
mbgs.params_
```

Out[33]:
```
r         0.4661
alpha   114.3980
a         0.0000
b         0.6303
dtype: float64
```

Creating a function to estimate the probability of the customers alive at time 't'

In [34]:
```
def future_probability_alive(t):

    future_probability = 1 / (1 + ((gamma(mbgs.params_['a'] + 1) * gamma(mbgs.
params_['b'] + data_simulated['frequency'])) /
                (gamma(mbgs.params_['a']) * gamma(mbgs.params_['b'] + data_simula
ted['frequency'] + 1))) *
                (((mbgs.params_['alpha'] + t) / (mbgs.params_['alpha'] + data_sim
ulated['recency'])) ** (mbgs.params_['r'] + data_simulated['frequency'])))

    future_probability = pd.DataFrame(future_probability, columns = ['Probabil
ityAlive'])

    return future_probability
```

For 10 Davs

The purchases the customers would make in 10 days are:

In [35]:
```
future_purchases(10).sum()
```

Out[35]:
```
PredictedValues   4.0741
dtype: float64
```

The number of customers alive after 10 days with alive probability greater than 0.5 are:

In [36]:
```
future_10 = future_probability_alive(10)
```

In [37]:
```
future_10.loc[future_10['ProbabilityAlive'] >= 0.5].count()
```

Out[37]:
```
ProbabilityAlive    100
dtype: int64
```

e. Repeat b-d for 1 year, 10 years, 100 years.

For 1 year

The purchases the customers would make in 365 days or 1 year are:

In [38]:
```
future_purchases(365).sum()
```

Out[38]:
```
PredictedValues   148.7041
dtype: float64
```

The number of customers alive after 365 days or 1 year with alive probability greater than 0.5 are:

In [39]:
```
future_365 = future_probability_alive(365)
```

In [40]:
```
future_365.loc[future_365['ProbabilityAlive'] >= 0.5].count()
```

Out[40]:
```
ProbabilityAlive    100
dtype: int64
```

For 10 years

The purchases the customers would make in 3652 days or 10 years are:

In [41]:
```
future_purchases(3652).sum()
```

Out[41]:
```
PredictedValues   1487.8558
dtype: float64
```

The number of customers alive after 3652 days or 10 years with alive probability greater than 0.5 are:

In [42]:
```
future_3652 = future_probability_alive(3652)
```

In [43]:
```
future_3652.loc[future_3652['ProbabilityAlive'] >= 0.5].count()
```

Out[43]:
```
ProbabilityAlive    99
dtype: int64
```

The number of customers alive after 3652 days or 10 years with alive probability greater than 0.1 are:

In [44]:
```
future_3652.loc[future_3652['ProbabilityAlive'] >= 0.1].count()
```

Out[44]:
```
ProbabilityAlive    99
dtype: int64
```

For 100 years

The purchases the customers would make in 36524 days or 100 years are:

In [45]:
```
future_purchases(36524).sum()
```

Out[45]:
```
PredictedValues   14880.1853
dtype: float64
```

The number of customers alive after 36524 days or 100 years with alive probability greater than 0.5 are:

In [46]:
```
future_36524 = future_probability_alive(36524)
```

In [47]:
```
future_36524.loc[future_36524['ProbabilityAlive'] >= 0.5].count()
```

Out[47]:
```
ProbabilityAlive    92
dtype: int64
```

The number of customers alive after 36524 days or 100 years with alive probability greater than 0.05 are:

In [48]:
```
future_36524.loc[future_36524['ProbabilityAlive'] >= 0.05].count()
```

Out[48]:
```
ProbabilityAlive    94
dtype: int64
```

Some strengths of the model -

1) The model performs pretty well to estimate a customer's characteristics like future purchases and drop out probability

2) The model does well to predict what a group of customers, having the same characteristics will do collectively with respect to the number of purchases.

3) Due to the assumption that the customer can drop out after first purchase, the modified beta geometric model tends to fit better than beta geometric model

Some deficiencies of the model -

1) The model struggles to understand that the customer's behaviour may change overtime. For example, if a customer builds a trust in a particular product or starts disliking a particular product, it may affect his purchasing patterns.

2) The model struggles to take into account a customer's purchase pattern during some promotional events. The promotional events may give a rise to a customer's rate of purchases disturbing the pattern of recency and frequency of purchases. Also the model struggles to capture the seasonality of sales. For example, increased sales during Christmas.

3) The customers who have a high probability of being active in their last purchase remain with an active status if no other purchase is observed. A major misclassification of active status might take place for customers with a very low number of repeat purchases in the observation period which might not be sufficient to specify their purchase pattern.

How can we address the deficiencies of the model?

If we observe the seasonality problem in the model, we can address this issue by bringing in seasonality covariates, which would help to reduce the prediction gaps in the seasonal purchases. If we split the data by taking out the parts of data where seasonality occurs and fit it separately it might help to predict the seasonality in a much better way. Bringing in cyclicality related to business, customers and macroeconomy will also help to address this issue.