Continuous-Time Markov Chains for Modeling User Journies

Drue Jaramillo
Department of Applied Mathematics & Statistics
Johns Hopkins University
Baltimore, MD 21218
`djarami3@jh.edu`

Abstract

Modern digital marketing centers on the analysis and modeling of customer journies in order to determine the effectiveness of both current and potential marketing efforts. While there are a plethora of models of user behavior, we consider the approach of a simple continuous-time Markov chain (CTMC). We also model the total number of conversions as a Poisson process. Using public data, we perform some preliminary analysis of user journies, fit a CTMC and Poisson process using Bayesian methods, and present some insights of the models.

1 Introduction

With the growth of digital marketing and the ability to track users' individual interactions with companies' marketing efforts, there has been much exploration into the study of the sequence of users' interactions with marketing assets, often referred to as the "user journey." By studying these user journies, researchers and companies hope to garner insight into how potential customers truly interact with companies and react to different marketing efforts. As a result, companies achieve greater return on ad spend, and consumers receive less spam-like marketing.

1.1 The data

Website traffic and user interactions tend to be closely-guarded secrets among businesses since they reveal sensitive information that a competitor could easily use to their advantage. Accordingly, companies do not tend to publish this information to the public. However, there is a dataset published at Lab [2017] that we will be using, which consists of 30 days of live traffic data for Criteo, an e-commerce platform. Each row corresponds to a single user impressions (interactions) and contains several potentially relevant features. We will only examine features relating to the impressions time (in seconds), user ID, campaign ID, and whether or not the user converted in the subsequent 30 days (and if so, the time and ID of the conversion). A "conversion" refers to the user taking an action that is the end goal of the company, such as making a purchase or joining an email list. The data has also been anonymized to protect privacy. A high-level summary of the data can be seen in Table 1.

Table 1: Summary of live traffic data

# of users	6,142,256
# of conversions	435,810
# of users that converted	327,860
% of users that converted	5.34%

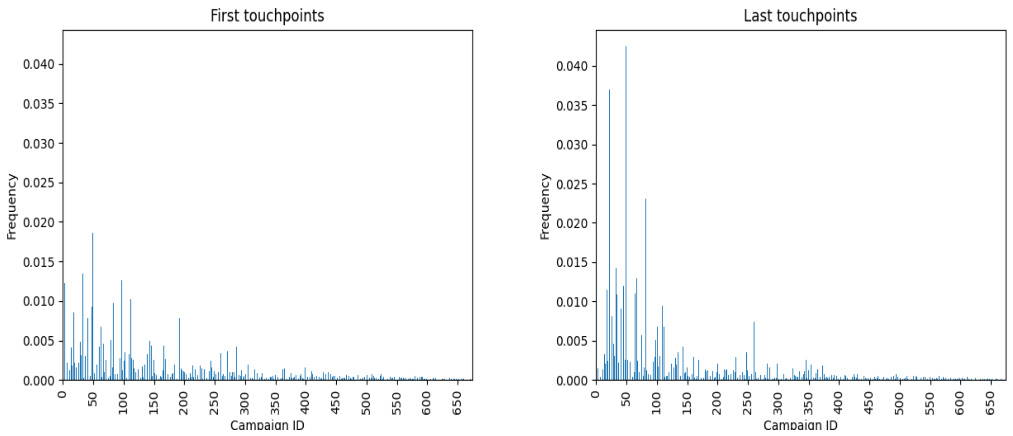


Figure 1: Frequency of first and last touchpoints by campaign

1.2 Related work

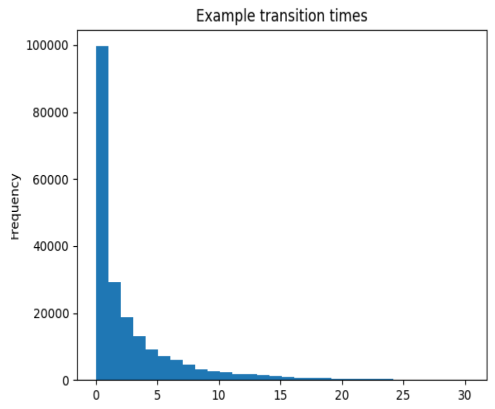
Markov chains have been used in a wide variety of contexts with regards to studying user behavior. Lim et al. [2020] used CTMCs to model user interactions with online heatlh infomediaries in order to provide insights into optimal design and management of these informediaries. Vermeer et al. [2020] utilizes discrete-time Markov chains to analyze how users move across news sites and topics, which enabled the detection of patterns in consumers' news consumption. There have also been many use cases for Markov chains in marketing research, which we address in section 3.3.

1.3 Preliminary analysis

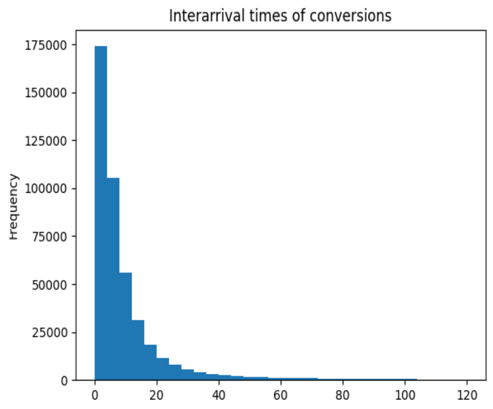
As we will address in section 3.3, one use case of user journey analysis is the attribution of conversions to different marketing channels and/or campaigns. Besides analytical models such as Markov chains, Schultz and Dellnitz [2017] shows that there are also simple heuristics such as: 1) attributing a conversion to the *first* touchpoint in the user's journey, and 2) attributing a conversion to the *last* touchpoint in the user's journey. Accordingly, Figure 1 shows the frequency that each campaign is the first or last touchpoint for a user's journey. Note that each graph shows only a subset of the campaigns in order to make the bars more visible. Immediately, we see that for both the first and last touchpoints, a subset of the campaigns seem to be the most frequent for each, but the subset is not the same for both.

Furthermore, if we are modeling a user's journey as a CTMC, we would expect the transition times between any particular pair of states (in our case, campaigns) to be exponentially distributed. Figure 2a displays a histogram of the transition times between the pair of campaigns that had the most transitions over the 30-day period we examined. Indeed, the data appear exponentially distributed, which gives us reason to believe that the CTMC model may be appropriate. Similarly, if we are modeling the total number of conversions as a Poisson process, we would expect the interarrival times of the conversions to also be exponentially distributed. Figure 2b displays a histogram of the interarrival times. Again, the data appear exponentially distributed, which lends credence to the Poisson process model.

There are two more pieces of information that may be of interest in the analysis: the number of touchpoints among converted users and the total amount of time a user takes to convert. The distribution of this information helps visualize the overall length – in terms of both steps and time – of the typical user that converts, as well as spot potential outliers. Table 2 displays summary statistics for the total number of touchpoints among users that converted, and Table 3 displays summary statistics for the total time (in days) it took those users to convert.



(a) Histogram of example transition times



(b) Histogram of conversion interarrival times

Figure 2: Evidence of exponential distribution

Table 2: Number of touchpoints

Mean	3.952
SD	5.086
Min	1
25%	1
50%	2
75%	5
Max	234

Table 3: Total time to conversion (days)

Mean	11.094
SD	11.828
Min	0.000012
25%	0.753
50%	7.083
75%	18.335
Max	60.114

2 The model

See Dobrow [2016]. We assume that the transition times between states i and j are exponentially distributed according to a transition rate q_{ij}:

$$T_{ij,1}, \dots, T_{ij,n_{ij}} \overset{\text{i.i.d.}}{\sim} \exp(q_{ij}) \tag{1}$$

where $T_{ij,1}, \dots, T_{ij,n_{ij}}$ are the observed transition times from state i to state j. Furthermore, we assume that the user journey ends when the user finally converts or has no touchpoints for 30 days. Hence, the "conversion" and "null" (no conversion) states are absorbing, which means $q_{ij} = 0$ whenever i corresponds to these two states. Lastly, we assume that the interarrival times $X_1, \dots, X_n$ between conversions are exponentially distributed according to some parameter λ:

$$X_1, \dots, X_n \overset{\text{i.i.d.}}{\sim} \exp(\lambda) \tag{2}$$

2.1 Prior distributions

We will use standard conjugate priors for the q_{ij} and λ:

$$q_{ij} \sim \text{gamma}(\alpha_q, \beta_q) \tag{3}$$
$$\lambda \sim \text{gamma}(\alpha_\lambda, \beta_\lambda) \tag{4}$$

Note that all the transition rates q_{ij} share a common prior. In order to make these priors weakly-informative, we will set $\alpha_q = 1$ and $\beta_q = 259,200$ (which is 3 days), as well as $\alpha_\lambda = 1$ and $\beta_\lambda = 300$ (which is 5 minutes). In essence, these priors give an expected transition time of 3 days for each pair of campaigns and an expected interarrival time of 5 minutes between conversions. Figure 3 plots these priors.

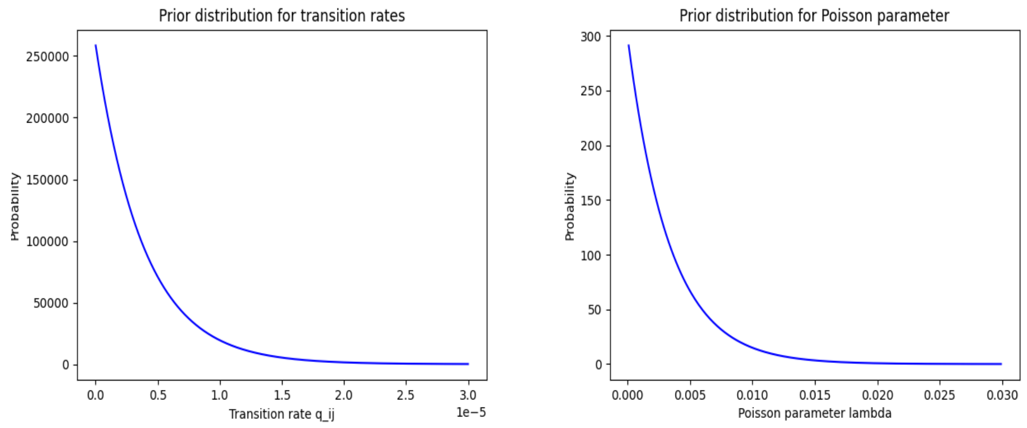


Figure 3: Prior distributions for transition rates and Poisson process parameter

2.2 Posterior distributions

Using Bayes' theorem, we find that the posterior distributions of the q_{ij} and λ are

$$q_{ij} \sim \text{gamma}\left(\alpha_q + n_{ij},\ \beta_q + \sum_{k=1}^{n_{ij}} t_{ijk}\right) \tag{5}$$

$$\lambda \sim \text{gamma}\left(\alpha_\lambda + n,\ \beta_\lambda + \sum_{i=1}^{n} x_i\right) \tag{6}$$

where n_{ij} is the number of transitions observed from state i to j, t_{ijk} is the kth observed transition time from state i to j, n is the total number of conversions, and x_i is the ith observed interarrival time between conversions.

3 Analysis

Figure 4 plots the posterior distribution for: 1) the transition rate q_{ij} of the example pair of campaigns given earlier, and 2) the Poisson process parameter λ. It is immediately obvious that due to the large amount of data, the posterior distributions for all parameters has somewhat "collapsed" into almost a point mass. Thus, by taking the expected value of these distributions, we know that our estimates are highly probable given the data. Now, we know that the expectation of a gamma(α, β) random variable is $\frac{\alpha}{\beta}$. This means that our estimators are

$$\hat{q}_{ij} = E[q_{ij} | t_{ijk},\ 1 \le k \le n_{ij}] = \frac{\alpha_q + n_{ij}}{\beta_q + \sum_{k=1}^{n_{ij}} t_{ijk}}$$

$$\hat{\lambda} = E[\lambda | x_i,\ 1 \le i \le n] = \frac{\alpha_\lambda + n}{\beta_\lambda + \sum_{i=1}^{n} x_i}$$

Now that we have estimates for the transition rates, we can estimate the holding time parameters $q_i = \sum_j q_{ij}$ for each state i. We will use the estimator $\hat{q}_i = \sum_j \hat{q}_{ij}$ for simplicity. Since the holding times of state i are exponentially distributed with parameter q_i, it follows that the expected holding time of state i is $\frac{1}{q_i}$. Plugging in our estimate, we get an expected holding time for each transient state, as shown in Figure 5a. We see that most states have relatively short expected holding times, whereas a small subset of states have expected holding times that are far longer than the others. In other words, a small number of campaigns seem to make users take a longer amount of time to come across another campaign.

Lastly, since we have all transition rates and holding time parameters, we can calculate the generator Q of our CTMC model. From there, we can derive the fundamental matrix F, whose row sums give

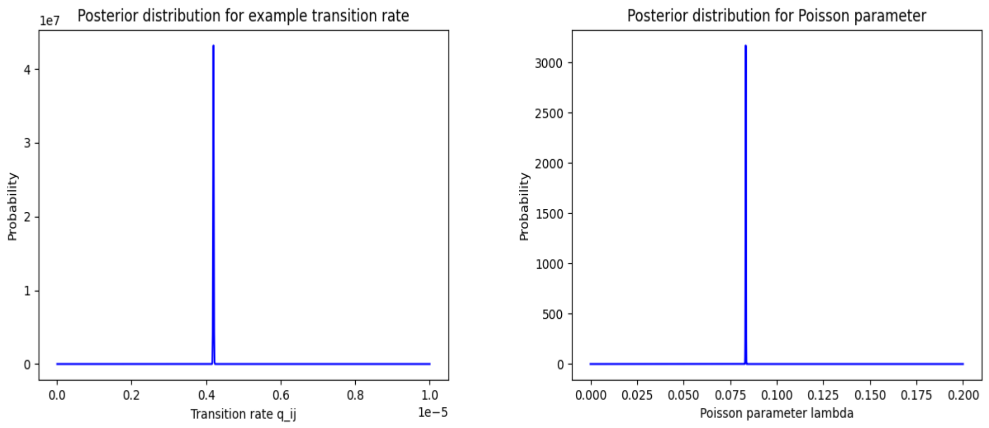
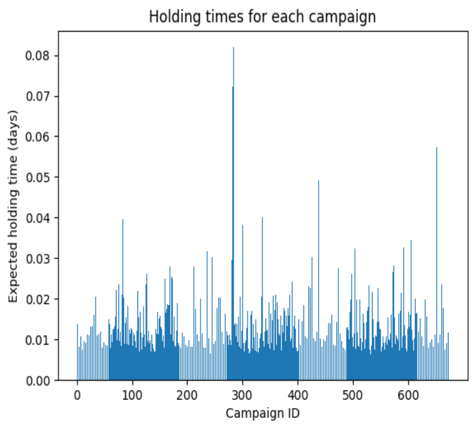
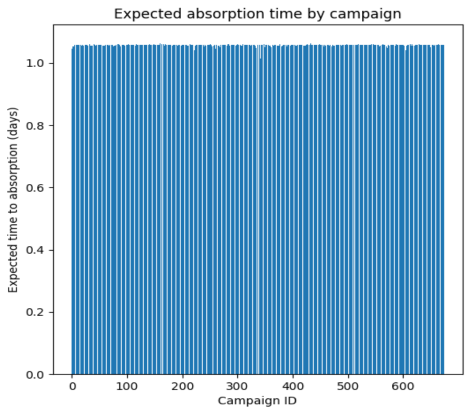


Figure 4: Posterior distributions for example and Poisson process parameter



(a) Expected holding times for transient states



(b) Expected absorption times for transient states

Figure 5: Expected holding and absorption times

us the expected time until absorption for each state i. Figure 5b shows the results of this calculation (in days). Unexpectedly, it appears that the expected time until absorption is roughly the same no matter what state we start in. In essence, user journies have roughly the same expected length (in time) no matter which campaign they first encounter.

3.1 Parting paradox

Despite the fact that the expected interarrival time between conversions is $E[X_i] = \frac{1}{\lambda}$, it is simultaneously the case that at any particular time t, the expected time between the last conversion and the next conversion is roughly $\frac{2}{\lambda}$. We will prove this fact for any Poisson process with parameter λ:

Consider a fixed t. The current total number of arrivals is N_t. Accordingly, the previous arrival time is S_{N_t}, and the next arrival time is S_{N_t+1}. The expected time between the previous and next arrivals is

$$E[S_{N_t+1} - S_{N_t}] = E[S_{N_t+1}] - E[S_{N_t}]$$

If $N_t = k$, then the $(k+1)$th arrival occurs after time t and is thereby independent of N_t (due to independent increments). Hence,

$$E[S_{N_t+1}|N_t = k] = E[S_{k+1}|N_t = k] = E[S_{k+1}] = \frac{k+1}{\lambda}$$

which implies that $E[S_{N_t+1}|N_t] = \frac{N_t+1}{\lambda}$. By the law of total expectation,

$$E[S_{N_t+1}] = E[E[S_{N_t+1}|N_t]] = E\left[\frac{N_1+1}{\lambda}\right] = \frac{\lambda t+1}{\lambda} = t + \frac{1}{\lambda}$$

Recall that given $N_t = k$, the arrival times $S_1, \dots, S_k$ have the same distribution has the order statistics of k uniform random variables on $(0, t)$, which means that S_k has the same distribution as the maximum of k uniform random variables on $(0, t)$. Thus,

$$E[S_{N_t}|N_t] = \frac{tN_t}{N_t+1} = t - \frac{t}{N_t+1}$$

By the law of total expectation,

$$E[S_{N_t}] = E[E[S_{N_t}|N_t]] = E\left[t - \frac{t}{N_t+1}\right] = t - t \cdot E\left[\frac{1}{N_t+1}\right]$$

Furthermore,

$$\begin{aligned}
E\left[\frac{1}{N_t+1}\right] &= \sum_{k=0}^{\infty} \frac{1}{k+1} \cdot \frac{e^{-\lambda t}(\lambda t)^k}{k!} \\
&= \frac{e^{-\lambda t}}{\lambda t} \sum_{k=0}^{\infty} \frac{(\lambda t)^{k+1}}{(k+1)!} \\
&= \frac{e^{-\lambda t}}{\lambda t} \sum_{k=1}^{\infty} \frac{(\lambda t)^k}{k!} \\
&= \frac{e^{-\lambda t}}{\lambda t}(e^{\lambda t} - 1) \\
&= \frac{1 - e^{-\lambda t}}{\lambda t}
\end{aligned}$$

Thus, substituting this into the previous equation,

$$E[S_{N_t}] = t - \frac{1}{\lambda} + \frac{e^{-\lambda t}}{\lambda}$$

It follows that

$$E[S_{N_t+1} - S_{N_t}] = \left(t + \frac{1}{\lambda}\right) - \left(t - \frac{1}{\lambda} + \frac{e^{-\lambda t}}{\lambda}\right) = \frac{2 - e^{-\lambda t}}{\lambda}$$

Therefore, for large t,

$$E[S_{N_t+1} - S_{N_t}] \approx \frac{2}{\lambda}$$

Applying this to our model, the expected interarrival time between conversions is $\frac{1}{\lambda} = 11.9989$ seconds, whereas at any given time t, the expected time between the previous conversion and the next conversion is roughly $\frac{2}{\lambda} = 23.9978$ seconds.

3.2 Assumptions

Throughout this process, we have made several simplifying assumptions. Here, we will address each of them:

1. The transition rates and Poisson process parameter have gamma priors.
   This assumption was made for two reasons. First, our model's likelihood function was an exponential distribution, and the gamma distribution is the only conjugate distribution for the exponential distribution. Hence, if we used a gamma prior, we would get a gamma posterior, which made the process much simpler. Second, it made intuitive sense that the parameters would be more likely to be close to 0, and the gamma distribution fit this intuition.

2. When a journey ended, the hard-coded transition time from the final touchpoint to the "null" state was 3 days.
   This was just a random, semi-inuitive choice. There is no right answer, so the goal was to choose something that was "long enough" to be sure the journey ended but "short enough" so as to not prolong the journey's length unnecessarily. The drawback of this model is that the expected holding times and absorption times are necessarily sensitive to this choice.
3. Journies have no touchpoints after the first conversion.
   Intuitively, user behavior will be different with regards to a particular company *after* they have made some commitment to that company (purchase, email signup, etc.). Accordingly, the focus was on analyzing how users get to the first conversion since this is the fundamental driver of business.
4. Deleting journies with only one touchpoint.
   Markov chains are sequences. It becomes impossible to analyze a Markov chain with only one observation since there are no transitions. In addition, user journies with one touchpoint don't tell us anything except that the user interacted with the company once and then disappeared.
5. User journies have all non-conversion touchpoints within 30 days.
   Since our data was limited to 30 days, we must assume that the journies we analyze (which are necessarily shorter than 30 days) are complete. Otherwise, we cannot create a model at all. However, for many businesses, this is a mostly reasonable assumption.

3.3 Applications

There are many applications for Markov chain models of user journies that are beyond the scope of this paper. Kakalejcik et al. [2021] used Markov chains to attribute revenue to different marketing channels and found improvement over heuristics such as first-touch and last-touch attribution. In addition, they found that higher-order Markov chains provided even better results. Koch et al. [2023] also used Markov chains for attribution, but instead did so on a rolling-basis, updating the Markov chain over time. On the other hand, Ma et al. [2008] used Markov chains as part of a larger model to estimate customer lifetime value, which enables businesses to focus on their best customers.

4 Conclusion

Markov chains are a powerful tool for modeling user interactions with marketing efforts, and thus they allow companies to analyze user trends and optimize their marketing. CTMCs are particularly useful since they allow for the modeling of not only the discrete state space of marketing assets, but also the continuous-time component of user behavior. As data becomes more detailed and extensive over time, these models will only become more useful.

Appendix: Python Code

```python
import pandas as pd
import numpy as np
from scipy.stats import gamma
import matplotlib.pyplot as plt

# Read data
data = pd.read_csv('./criteo_attribution_dataset/criteo_attribution_dataset.tsv.gz', sep='\t', compression='gzip')
data = data[['timestamp', 'uid', 'campaign', 'conversion', 'conversion_timestamp', 'conversion_id']]

# Calculate basic numbers
num_users = data['uid'].nunique()
num_conversions = data[data['conversion'] == 1]['conversion_id'].nunique()
num_converts = data[data['conversion'] == 1]['uid'].nunique()
percent_convert = num_converts / num_users
print(f'\n\nNumber␣of␣users:␣{num_users}')
print(f'Number␣of␣conversions:␣{num_conversions}')
print(f'Number␣of␣users␣that␣converted:␣{num_converts}')
print(f'Percent␣of␣users␣that␣converted:␣{round(100*percent_convert,␣2)}%')

# Sort data into user journies
journies = data.sort_values(by=['uid', 'timestamp']).reset_index(drop=True)

# Extract journies for users that converted
converts = journies[journies['conversion'] == 1]['uid'].unique()
```

```
converted_journies = journies[journies['uid'].isin(converts)]

# Find when each user first converted
conversion_times = converted_journies[converted_journies['conversion_timestamp'] != -1]
conversion_times = conversion_times.groupby('uid').first()
conversion_times = conversion_times['conversion_timestamp']
conversion_times.rename('first_conversion_timestamp', inplace=True)

# Delete touchpoints after the first conversion
converted_journies = converted_journies.join(conversion_times, on='uid', how='left')
converted_journies = converted_journies[converted_journies['timestamp'] <= converted_journies['first_conversion_timestamp']].reset_index(drop

# Replace the original conversion journies with the filtered ones
non_converted_journies = journies[~(journies['uid'].isin(converts))]
converted_journies.drop('first_conversion_timestamp', axis=1, inplace=True)
journies = pd.concat([non_converted_journies, converted_journies], ignore_index=True).sort_values(by=['uid', 'timestamp'])

# Find the first touchpoint's campaign for converted users
first_touches = converted_journies.groupby('uid').first()['campaign']
first_counts = first_touches.value_counts(normalize=True, sort=False)
first_counts.plot(kind='bar', xticks=range(0, len(first_counts), 50), title='First␣touchpoints', xlabel='Campaign␣ID', ylabel='Frequency')
plt.savefig('./first_counts.png')
plt.close()
plt.cla()
plt.clf()

# Find the last touchpoint's campaign for converted users
last_touches = converted_journies.groupby('uid').last()['campaign']
last_counts = last_touches.value_counts(normalize=True, sort=False)
last_counts.plot(kind='bar', xticks=range(0, len(last_counts), 50), title='Last␣touchpoints', xlabel='Campaign␣ID', ylabel='Frequency')
plt.savefig('./last_counts.png')
plt.close()
plt.cla()
plt.clf()

# Find the starting times for converted users' journies
initial_times = converted_journies.groupby('uid').first()
initial_times = initial_times['timestamp']

# Find the total time to conversion for converted users
total_conversion_times = conversion_times.subtract(initial_times)
total_conversion_times /= 86400
total_conversion_times.plot.hist(bins=30, title='Total␣times␣to␣conversion', xlabel='Total␣time␣of␣user␣journey', ylabel='Frequency')
plt.savefig('./total_conversion_times.png')
plt.close()
plt.cla()
plt.clf()
print('\n\nTotal␣times␣to␣conversion:')
print(total_conversion_times.describe())

# Find the number of touchpoints for each converted user
num_touchpoints = converted_journies.groupby('uid').count()
num_touchpoints = num_touchpoints['timestamp']
q = num_touchpoints.quantile(0.99) # remove outlier
num_touchpoints[num_touchpoints < q].plot.hist(bins=20, title='Total␣number␣of␣touchpoints', xlabel='#␣of␣touchpoints␣in␣user␣journey', ylabe
plt.savefig('./num_touchpoints.png')
plt.close()
plt.cla()
plt.clf()
print('\n\nNumber␣of␣touchpoints:')
print(num_touchpoints.describe())

# Create a list of states for the Markov chain
states = list(journies['campaign'].unique())
states.sort()
states.extend(['conversion', 'null'])

# Filter out journies of length 1 (only 1 touchpoint)
journies = journies[journies.groupby('uid').transform('size') > 1]

# For each touchpoint for each user, add the previous touchpoint's campaign
journies['prev_campaign'] = journies.groupby('uid')['campaign'].shift(periods=1).astype('Int64')

# For each touchpoint for each user, add the time since the last touchpoint
journies['time_since_last'] = journies.groupby('uid')['timestamp'].diff(periods=1).astype('Int64')

# Extract the transition times
transition_times = journies[['prev_campaign', 'campaign', 'time_since_last']].copy()
transition_times.sort_values(by=['prev_campaign', 'campaign', 'time_since_last'], inplace=True)

# Find the number of transitions between each pair of states (campaigns)
num_transitions = transition_times.groupby(['prev_campaign', 'campaign']).count()

# Plot transition times for pair of states with the most transitions
example = num_transitions.idxmax()
print(f'Pair␣of␣campaigns␣with␣the␣most␣transitions:␣{example}')
example_times = transition_times[(transition_times['prev_campaign'] == example[0][0]) & (transition_times['campaign'] == example[0][1])]
(example_times['time_since_last'] / 86400).plot.hist(bins=30, title='Example␣transition␣times', xlabel='Transition␣time', ylabel='Frequency')
plt.savefig('./example_transition_times.png')
plt.close()
plt.cla()
plt.clf()
```

```
# For converted users, find the transition time from their last campaign to conversion
last_touches = converted_journies.groupby('uid').last()[['campaign', 'timestamp']]
last_touches.sort_values(by=['campaign', 'timestamp'], inplace=True)
last_touches = last_touches.join(conversion_times, how='left')
conversion_times = pd.DataFrame()
conversion_times['prev_campaign'] = last_touches['campaign']
conversion_times['campaign'] = 'conversion'
conversion_times['time_since_last'] = last_touches['first_conversion_timestamp'] - last_touches['timestamp']

# For non-converted users, fill in a transition time from their last campaign to "null"
last_touches = non_converted_journies.groupby('uid').last()['campaign']
last_touches.sort_values()
null_times = pd.DataFrame()
null_times['prev_campaign'] = last_touches
null_times['campaign'] = 'null'
null_times['time_since_last'] = 3*86400

# Put all the transition times together
transition_times = pd.concat([transition_times, conversion_times, null_times], ignore_index=True)
transition_times.sort_values(by=['prev_campaign', 'campaign', 'time_since_last'])

# Calculate the number of transitions between each pair of states (campaigns)
num_transitions = transition_times.groupby(['prev_campaign', 'campaign']).count()
print('\n\nNumber␣of␣transitions␣between␣pairs␣of␣states:')
print(num_transitions.describe())

# Sum the transition times for each pair of states
transition_time_sums = transition_times.groupby(['prev_campaign', 'campaign']).sum()

# Calculate the posterior parameters for the transition rate between each pair of states
alpha = 1
beta = 3*86400
posterior_params = {state1: {state2: () for state2 in states} for state1 in states}
for state1 in states:
    for state2 in states:
        try:
            n = num_transitions.loc[(state1, state2), 'time_since_last']
            tt_sum = transition_time_sums.loc[(state1, state2), 'time_since_last']
        except:
            n = 0
            tt_sum = 5000000
        posterior_params[state1][state2] = (alpha+n, beta+tt_sum)

# Plot the common prior distribution for the transition rates
x = np.arange(1e-8, 3e-5, 1e-8)
y = gamma.pdf(x, alpha, scale=1/beta)
plt.plot(x, y, 'b-')
plt.xlabel('Transition␣rate␣q_ij')
plt.ylabel('Probability')
plt.title('Prior␣distribution␣for␣transition␣rates')
plt.savefig('./prior_transition_rates.png')
plt.close()
plt.cla()
plt.clf()

# Plot the posterior for the pair with the most transitions
a, b = posterior_params[example[0][0]][example[0][1]]
x = np.arange(1e-8, 1e-5, 1e-8)
y = gamma.pdf(x, a, scale=1/b)
plt.plot(x, y, 'b-')
plt.xlabel('Transition␣rate␣q_ij')
plt.ylabel('Probability')
plt.title('Posterior␣distribution␣for␣example␣transition␣rate')
plt.savefig('./posterior_example_transition_rate.png')
plt.close()
plt.cla()
plt.clf()

# Calculate posterior means for transition rates
posterior_means = {state1: {state2: 0 for state2 in states} for state1 in states}
for state1 in states:
    for state2 in states:
        a, b = posterior_params[state1][state2]
        posterior_means[state1][state2] = a / b
for state in states:
    posterior_means['conversion'][state] = 0
    posterior_means['null'][state] = 0

# Calculate holding time parameters and expected holding times
holding_time_params = {state: 0 for state in states}
for state1 in states:
    for state2 in states:
        holding_time_params[state1] += posterior_means[state1][state2]
holding_times = {state: (1/holding_time_params[state])/86400 for state in states if state not in ['conversion', 'null']}

# Plot expected holding times
plt.bar(range(len(holding_times)), list(holding_times.values()))
plt.xticks(range(0, len(holding_times), 100))
plt.xlabel('Campaign␣ID')
plt.ylabel('Expected␣holding␣time␣(days)')
plt.title('Holding␣times␣for␣each␣campaign')
plt.savefig('./holding_times.png')
plt.close()
plt.cla()
plt.clf()

# Create generator
Q = np.zeros((len(states), len(states)))
for i in range(len(states)):
    if states[i] not in ['conversion', 'null']:
        for j in range(len(states)):
            if i == j:
                Q[i][i] = -1 * holding_time_params[states[i]]
            else:
                Q[i][j] = posterior_means[states[i]][states[j]]

# Calculate fundamental matrix and expected absorption times
V = Q[:-2, :-2]
F = -1 * np.linalg.inv(V)
absorption_times = np.sum(F, axis=1) / 86400
print('\n\nExpected␣absorption␣times:')
print(pd.Series(absorption_times).describe())

# Plot expected absorption times
plt.figure(figsize=(6, 6))
plt.bar(range(len(F)), absorption_times)
plt.xticks(range(0, len(F), 100))
plt.xlabel('Campaign␣ID')
plt.ylabel('Expected␣time␣to␣absorption␣(days)')
plt.title('Expected␣absorption␣time␣by␣campaign')
plt.savefig('./absorption_times.png')
plt.close()
plt.cla()
plt.clf()

# Calculate arrival times for conversions
arrival_times = data[['uid', 'conversion_timestamp']].copy()
arrival_times = arrival_times[arrival_times['conversion_timestamp'] != -1]
arrival_times.drop_duplicates(inplace=True, ignore_index=True)
arrival_times = arrival_times['conversion_timestamp'].copy()
arrival_times.sort_values(inplace=True)

# Calculate and plot interarrival times
interarrival_times = arrival_times.diff(periods=1).dropna()
q = interarrival_times.quantile(0.99) # remove outlier
interarrival_times[interarrival_times < q].plot.hist(bins=30, title='Interarrival␣times␣of␣conversions', xlabel='Interarrival␣time', ylabel='
plt.savefig('./interarrival_times.png')
plt.close()
plt.cla()
plt.clf()
print('\n\nInterarrival␣times:')
print(interarrival_times.describe())

# Plot prior distribution for lambda (Poisson process parameter)
alpha = 1
beta = 300
x = np.arange(1e-4, 3e-2, 1e-4)
y = gamma.pdf(x, alpha, scale=1/beta)
plt.plot(x, y, 'b-')
plt.xlabel('Poisson␣parameter␣lambda')
plt.ylabel('Probability')
plt.title('Prior␣distribution␣for␣Poisson␣parameter')
plt.savefig('./prior_lambda.png')
plt.close()
plt.cla()
plt.clf()

# Plot posterior distribution for lambda
a = alpha + len(interarrival_times)
b = beta + np.sum(interarrival_times)
x = np.arange(1e-6, 2e-1, 1e-6)
y = gamma.pdf(x, a, scale=1/b)
plt.plot(x, y, 'b-')
plt.xlabel('Poisson␣parameter␣lambda')
plt.ylabel('Probability')
plt.title('Posterior␣distribution␣for␣Poisson␣parameter')
plt.savefig('./posterior_lambda.png')
plt.close()
plt.cla()
plt.clf()

# Calculate expected interarrival time and expected wait time (Parting Paradox)
posterior_mean = a / b
print(f'\n\nExpected␣interarrival␣time:␣{1/posterior_mean}')
print(f'Expected␣waiting␣time:␣{2/posterior_mean}')
```

References

Robert P. Dobrow. *Introduction to stochastic processes with R*. Wiley, Hoboken, New Jersey, 2016. ISBN 1118740653. URL `http://swbplus.bsz-bw.de/bsz468986243cov.htm`.

Lukas Kakalejcik, Jozef Bucko, and Paulo Resende. Multichannel marketing attribution using markov chains for e-commerce. *Statistika (Prague, Czech Republic)*, 101(2):142–158, Jan 1, 2021. URL `https://search.proquest.com/docview/2688054400`.

Christian Koch, Benedikt Lindenbeck, and Rainer Olbrich. Dynamic customer journey analysis and its advertising impact. *Journal of strategic marketing*, ahead-of-print(ahead-of-print):1–20, Jan 25, 2023. URL `https://www.tandfonline.com/doi/abs/10.1080/0965254X.2023.2171475`.

Criteo AI Lab. Criteo attribution modeling for bidding dataset, Oct 10, 2017. URL `http://ailab.criteo.com/criteo-attribution-modeling-bidding-dataset/`.

Sanghee Lim, Dobin Yim, Jiban Khuntia, and Mohan Tanniru. A continuous-time markov chain model–based business analytics approach for estimating patient transition states in online health infomediary. *Decision sciences*, 51(1):181–208, Feb 2020. URL `https://onlinelibrary.wiley.com/doi/abs/10.1111/deci.12430`.

Ming Ma, Zehui Li, and Jinyuan Chen. Phase-type distribution of customer relationship with markovian response and marketing expenditure decision on the customer lifetime value. *European journal of operational research*, 187(1):313–326, May 16, 2008. URL `https://dx.doi.org/10.1016/j.ejor.2007.03.018`.

Carsten Schultz and Andreas Dellnitz. *Attribution Modeling in Online Advertising*, pages 226–249. 12 2017. ISBN 9781522531142. doi: 10.4018/978-1-5225-3114-2.ch009.

Susan Vermeer, Damian Trilling, Sanne Kruikemeier, and Claes de Vreese. Online news user journeys: The role of social media, news websites, and topics. *Digital journalism*, 8(9):1114–1141, Oct 20, 2020. URL `https://www.tandfonline.com/doi/abs/10.1080/21670811.2020.1767509`.