Python Portfolio Project - Waze

February 27, 2024

1 *Waze Project*

[1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
```

[2]:
```
df = pd.read_csv('/Users/samantarana/Downloads/waze_dataset.csv')
```

[3]:
```
df.head(10)
```

[3]:

	ID	label	sessions	drives	total_sessions	n_days_after_onboarding	\
0	0	retained	283	226	296.748273	2276	
1	1	retained	133	107	326.896596	1225	
2	2	retained	114	95	135.522926	2651	
3	3	retained	49	40	67.589221	15	
4	4	retained	84	68	168.247020	1562	
5	5	retained	113	103	279.544437	2637	
6	6	retained	3	2	236.725314	360	
7	7	retained	39	35	176.072845	2999	
8	8	retained	57	46	183.532018	424	
9	9	churned	84	68	244.802115	2997	

	total_navigations_fav1	total_navigations_fav2	driven_km_drives	\
0	208	0	2628.845068	
1	19	64	13715.920550	
2	0	0	3059.148818	
3	322	7	913.591123	
4	166	5	3950.202008	
5	0	0	901.238699	
6	185	18	5249.172828	
7	0	0	7892.052468	
8	0	26	2651.709764	
9	72	0	6043.460295	

	duration_minutes_drives	activity_days	driving_days	device
0	1985.775061	28	19	Android

```
1              3160.472914             13             11   iPhone
2              1610.735904             14              8  Android
3               587.196542              7              3   iPhone
4              1219.555924             27             18  Android
5               439.101397             15             11   iPhone
6               726.577205             28             23   iPhone
7              2466.981741             22             20   iPhone
8              1594.342984             25             20  Android
9              2341.838528              7              3   iPhone
```

[4]:
```
df.size
```

[4]:
```
194987
```

[5]:
```
df.describe()
```

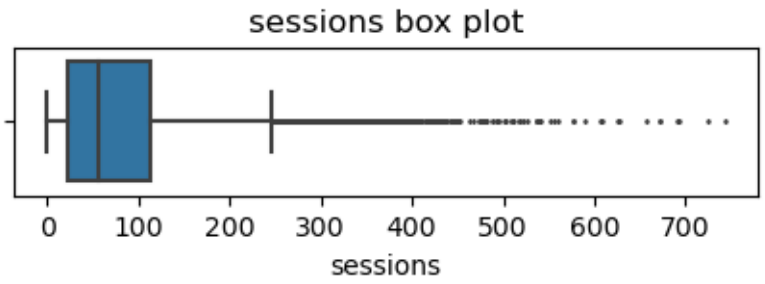
[5]:
```
                ID      sessions        drives  total_sessions  \
count  14999.000000  14999.000000  14999.000000    14999.000000
mean    7499.000000     80.633776     67.281152      189.964447
std     4329.982679     80.699065     65.913872      136.405128
min        0.000000      0.000000      0.000000        0.220211
25%     3749.500000     23.000000     20.000000       90.661156
50%     7499.000000     56.000000     48.000000      159.568115
75%    11248.500000    112.000000     93.000000      254.192341
max    14998.000000    743.000000    596.000000     1216.154633

       n_days_after_onboarding  total_navigations_fav1  \
count             14999.000000            14999.000000
mean               1749.837789              121.605974
std                1008.513876              148.121544
min                   4.000000                0.000000
25%                 878.000000                9.000000
50%                1741.000000               71.000000
75%                2623.500000              178.000000
max                3500.000000             1236.000000

       total_navigations_fav2  driven_km_drives  duration_minutes_drives  \
count            14999.000000      14999.000000             14999.000000
mean                29.672512       4039.340921              1860.976012
std                 45.394651       2502.149334              1446.702288
min                  0.000000         60.441250                18.282082
25%                  0.000000       2212.600607               835.996260
50%                  9.000000       3493.858085              1478.249859
75%                 43.000000       5289.861262              2464.362632
max                415.000000      21183.401890             15851.727160

       activity_days  driving_days
```

```
count   14999.000000  14999.000000
mean       15.537102     12.179879
std         9.004655      7.824036
min         0.000000      0.000000
25%         8.000000      5.000000
50%        16.000000     12.000000
75%        23.000000     19.000000
max        31.000000     30.000000
```

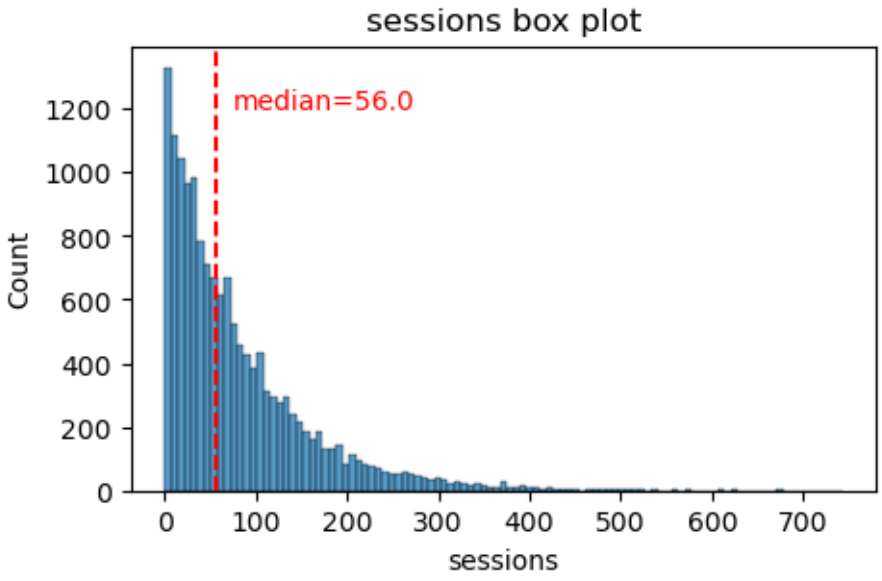
[6]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14999 entries, 0 to 14998
Data columns (total 13 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   ID                       14999 non-null  int64
 1   label                    14299 non-null  object
 2   sessions                 14999 non-null  int64
 3   drives                   14999 non-null  int64
 4   total_sessions           14999 non-null  float64
 5   n_days_after_onboarding  14999 non-null  int64
 6   total_navigations_fav1   14999 non-null  int64
 7   total_navigations_fav2   14999 non-null  int64
 8   driven_km_drives         14999 non-null  float64
 9   duration_minutes_drives  14999 non-null  float64
 10  activity_days            14999 non-null  int64
 11  driving_days             14999 non-null  int64
 12  device                   14999 non-null  object
dtypes: float64(3), int64(8), object(2)
memory usage: 1.5+ MB
```

[7]:
```
# Box plot
plt.figure(figsize=(5,1))
sns.boxplot(x=df['sessions'], fliersize=1)
plt.title('sessions box plot');
```

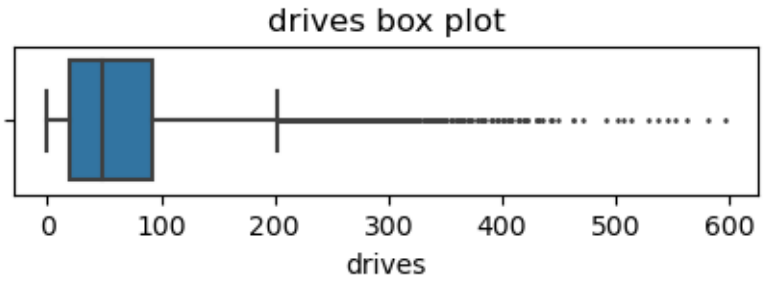


[8]:
```
# Histogram
plt.figure(figsize=(5,3))
sns.histplot(x=df['sessions'])
median = df['sessions'].median()
plt.axvline(median, color='red', linestyle='--')
plt.text(75,1200, 'median=56.0', color='red')
plt.title('sessions box plot');
```



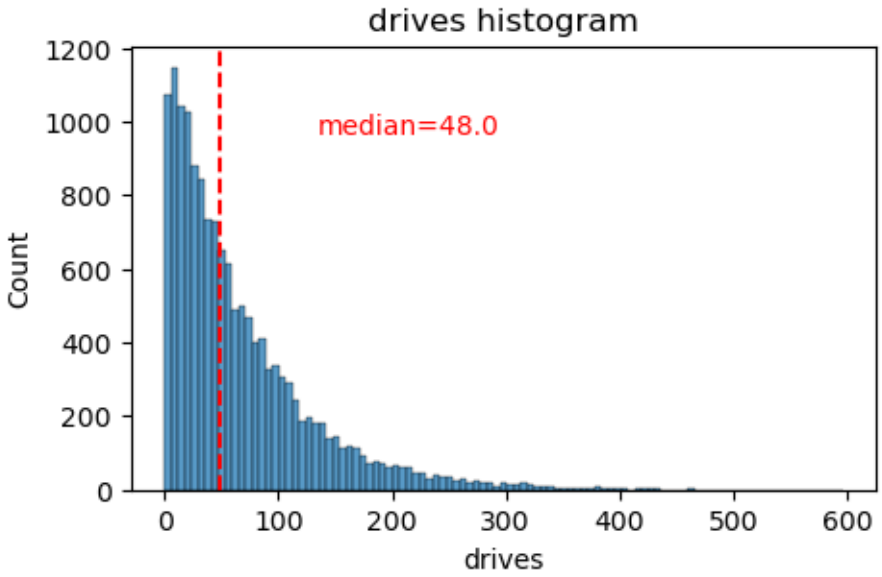
The sessions variable is a right-skewed distribution with half of the observations having 56 or fewer sessions. However, as indicated by the boxplot, some users have more than 700.

[9]:
```
# Box plot
plt.figure(figsize=(5,1))
sns.boxplot(x=df['drives'], fliersize=1)
plt.title('drives box plot');
```



[10]:
```
#Helper function to plot histograms based on the format of the `sessions`␣
↪histogram


def histogrammer(column_str, median_text=True, **kwargs):     # **kwargs = any␣
↪keyword arguments
                                                              # from the sns.
↪histplot() function
    median=round(df[column_str].median(), 1)
    plt.figure(figsize=(5,3))
    ax = sns.histplot(x=df[column_str], **kwargs)             # Plot the␣
↪histogram
    plt.axvline(median, color='red', linestyle='--')          # Plot the median␣
↪line
    if median_text==True:                                     # Add median text␣
↪unless set to False
        ax.text(0.25, 0.85, f'median={median}', color='red',
            ha='left', va='top', transform=ax.transAxes)
    else:
        print('Median:', median)
    plt.title(f'{column_str} histogram');
```

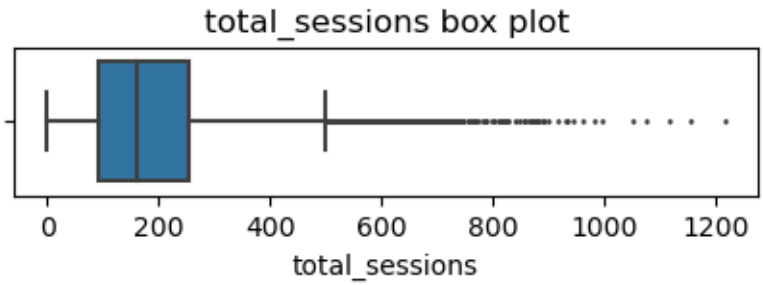
[11]:
```
# Histogram
histogrammer('drives')
```



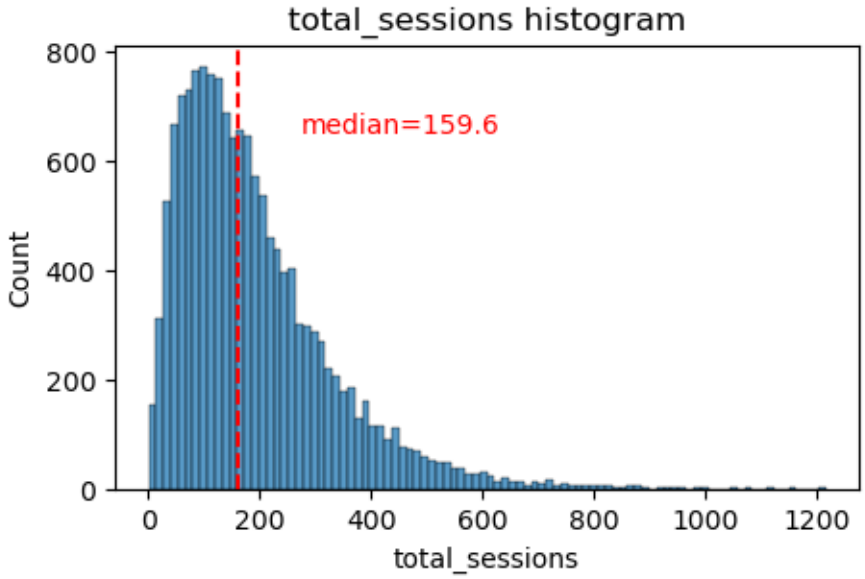
The drives information follows a distribution similar to the sessions variable. It is right-skewed, approximately log-normal, with a median of 48. However, some drivers had over 400 drives in the

last month.

[12]:
```
# Box plot
plt.figure(figsize=(5,1))
sns.boxplot(x=df['total_sessions'], fliersize=1)
plt.title('total_sessions box plot');
```

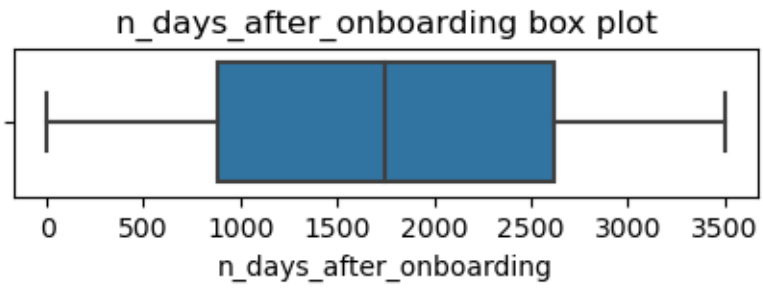


[13]:
```
# Histogram
histogrammer('total_sessions')
```



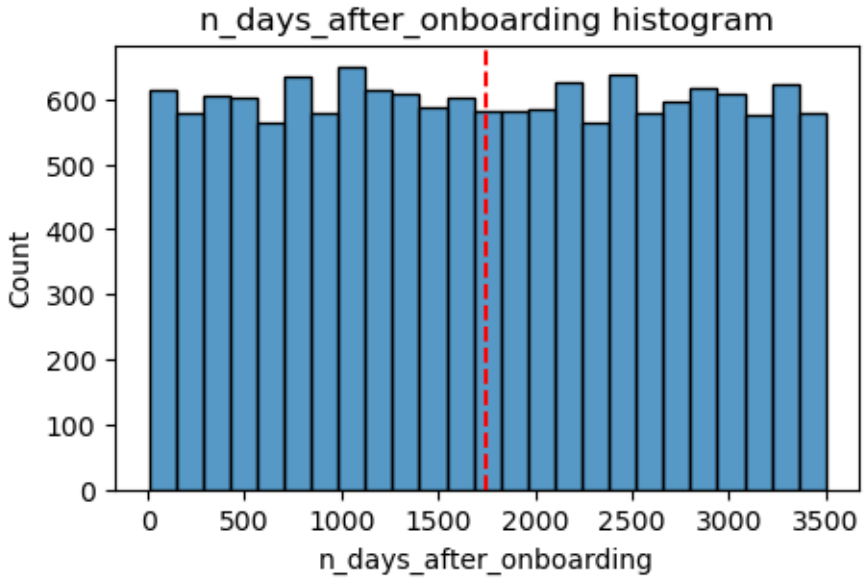
The total_sessions is a right-skewed distribution. The median total number of sessions is 159.6. This is interesting information because, if the median number of sessions in the last month was 56 and the median total sessions was ~160, then it seems that a large proportion of a user's (estimated) total drives might have taken place in the last month. This is something you can examine more closely later.

[14]:
```
# Box plot
plt.figure(figsize=(5,1))
sns.boxplot(x=df['n_days_after_onboarding'], fliersize=1)
plt.title('n_days_after_onboarding box plot');
```



[15]:
```
# Histogram
histogrammer('n_days_after_onboarding', median_text=False)
```

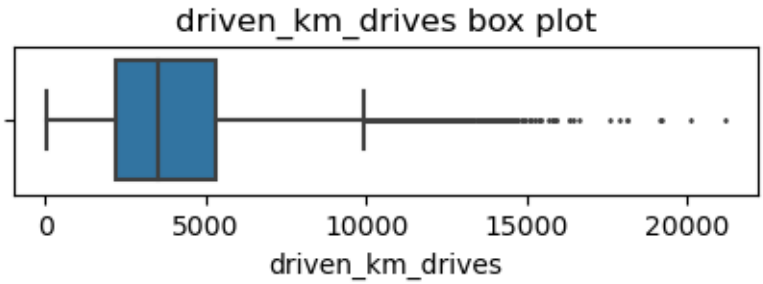
```
Median: 1741.0
```



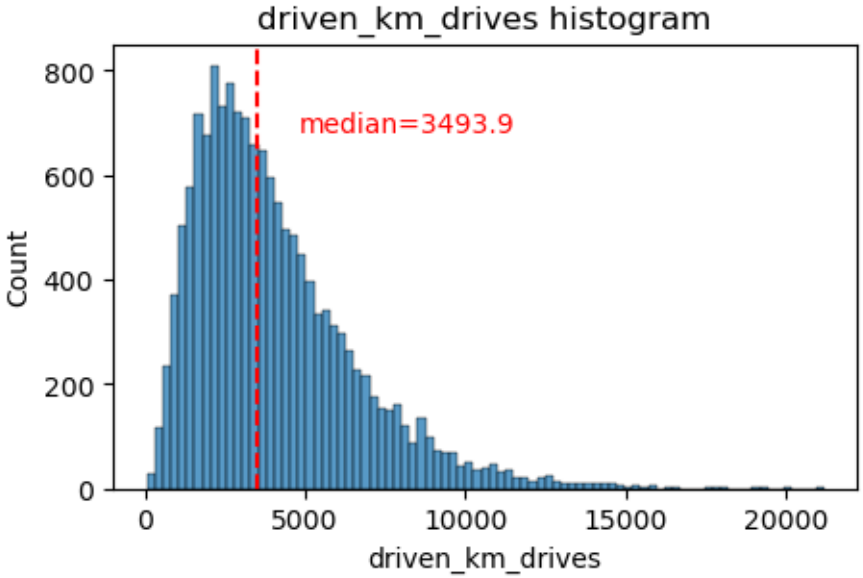
The total user tenure (i.e., number of days since onboarding) is a uniform distribution with values ranging from near-zero to ~3,500 (~9.5 years).

[16]:
```
# Box plot
plt.figure(figsize=(5,1))
sns.boxplot(x=df['driven_km_drives'], fliersize=1)
```

```
plt.title('driven_km_drives box plot');
```

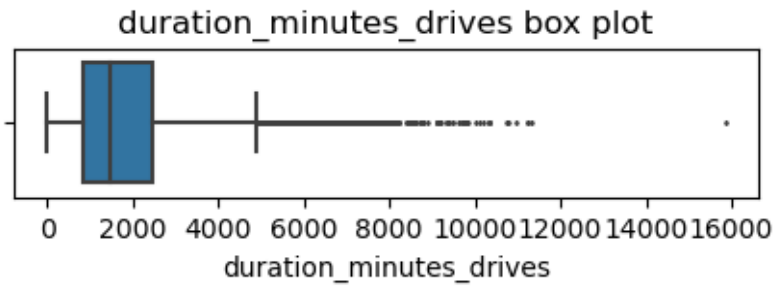


[17]:
```
# Histogram
histogrammer('driven_km_drives')
```

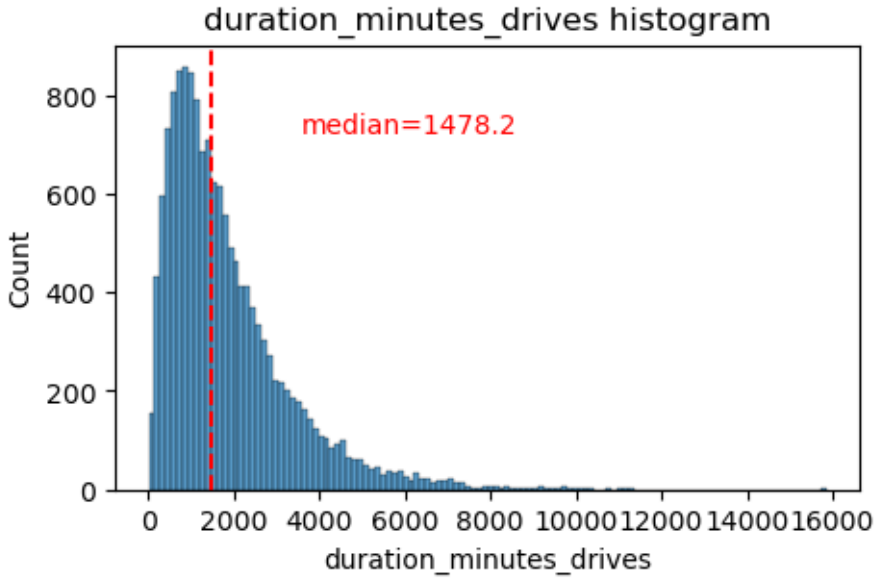


The number of drives driven in the last month per user is a right-skewed distribution with half the users driving under 3,495 kilometers. As you discovered in the analysis from the previous course, the users in this dataset drive a lot. The longest distance driven in the month was over half the circumferene of the earth.

[18]:
```
# Box plot
plt.figure(figsize=(5,1))
sns.boxplot(x=df['duration_minutes_drives'], fliersize=1)
plt.title('duration_minutes_drives box plot');
```

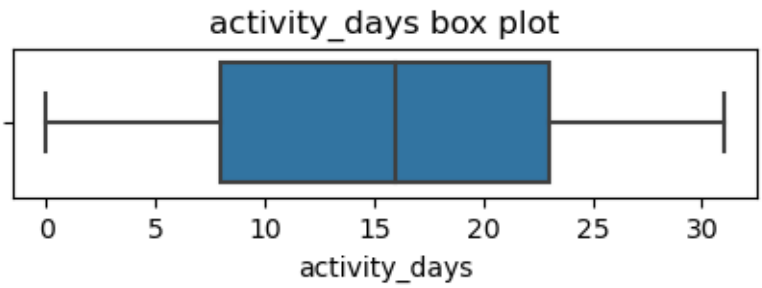


[19]:
```
# Histogram
histogrammer('duration_minutes_drives')
```



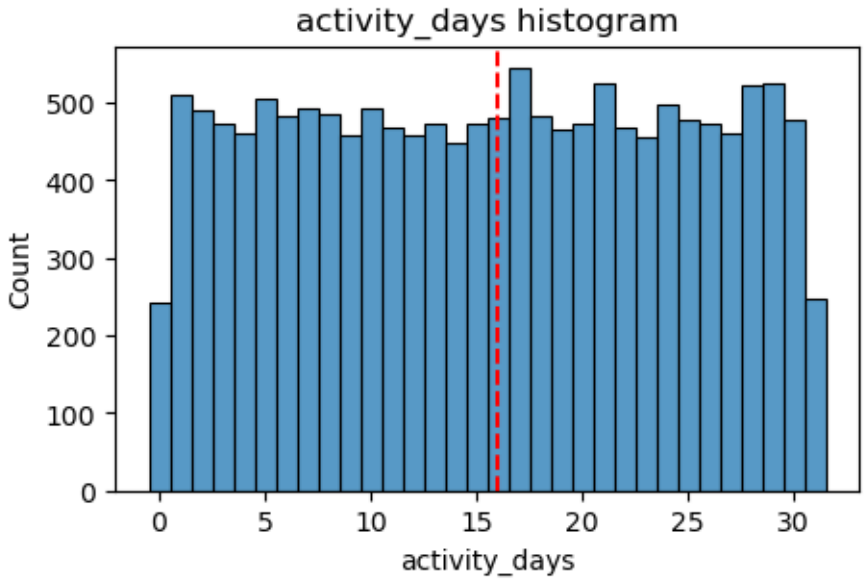
The duration_minutes_drives variable has a heavily skewed right tail. Half of the users drove less than ~1,478 minutes (~25 hours), but some users clocked over 250 hours over the month.

[20]:
```
# Box plot
plt.figure(figsize=(5,1))
sns.boxplot(x=df['activity_days'], fliersize=1)
plt.title('activity_days box plot');
```



[21]:
```
# Histogram
histogrammer('activity_days', median_text=False, discrete=True)
```

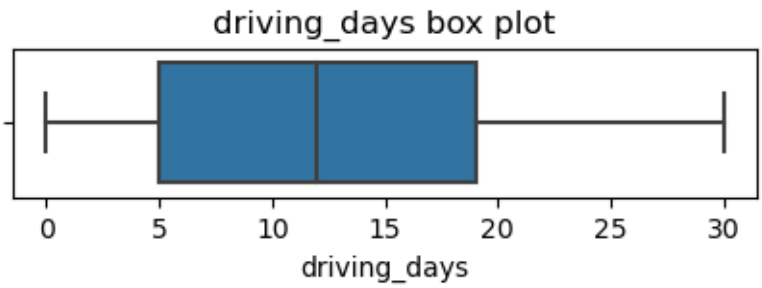
```
Median: 16.0
```



Within the last month, users opened the app a median of 16 times. The box plot reveals a centered distribution. The histogram shows a nearly uniform distribution of ~500 people opening the app on each count of days. However, there are ~250 people who didn't open the app at all and ~250 people who opened the app every day of the month.

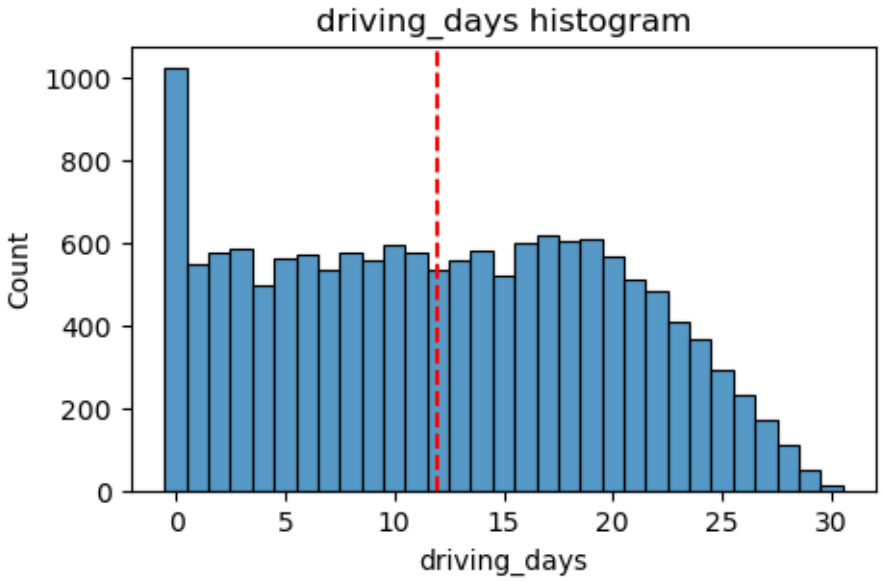
This distribution is noteworthy because it does not mirror the sessions distribution, which was thought to be closely correlated with activity_days.

[22]:
```
# Box plot
plt.figure(figsize=(5,1))
sns.boxplot(x=df['driving_days'], fliersize=1)
plt.title('driving_days box plot');
```



[23]:
```
# Histogram
histogrammer('driving_days', median_text=False, discrete=True)
```

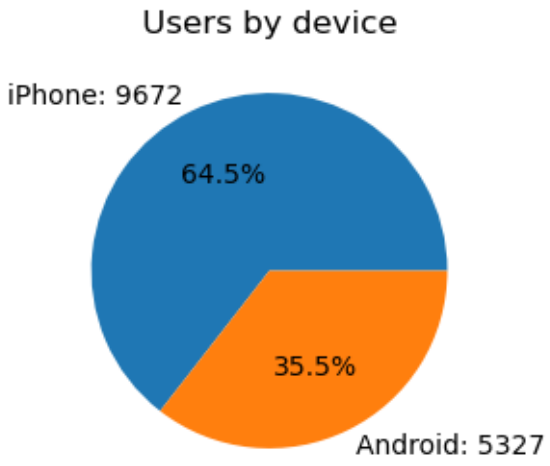
```
Median: 12.0
```



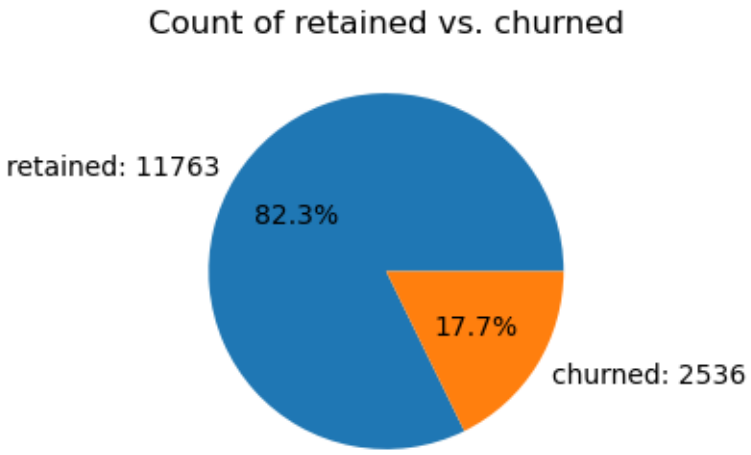
The number of days users drove each month is almost uniform, and it largely correlates with the number of days they opened the app that month, except the driving_days distribution tails off on the right.

However, there were almost twice as many users (~1,000 vs. ~550) who did not drive at all during the month. This might seem counterintuitive when considered together with the information from activity_days. That variable had ~500 users opening the app on each of most of the day counts, but there were only ~250 users who did not open the app at all during the month and ~250 users who opened the app every day. Flagging this for further investigation later.

[24]:
```
# Pie chart
fig = plt.figure(figsize=(3,3))
data=df['device'].value_counts()
plt.pie(data,
        labels=[f'{data.index[0]}: {data.values[0]}',
                f'{data.index[1]}: {data.values[1]}'],
        autopct='%1.1f%%'
        )
plt.title('Users by device');
```



[25]:
```
# Pie chart
fig = plt.figure(figsize=(3,3))
data=df['label'].value_counts()
plt.pie(data,
        labels=[f'{data.index[0]}: {data.values[0]}',
                f'{data.index[1]}: {data.values[1]}'],
        autopct='%1.1f%%'
        )
plt.title('Count of retained vs. churned');
```



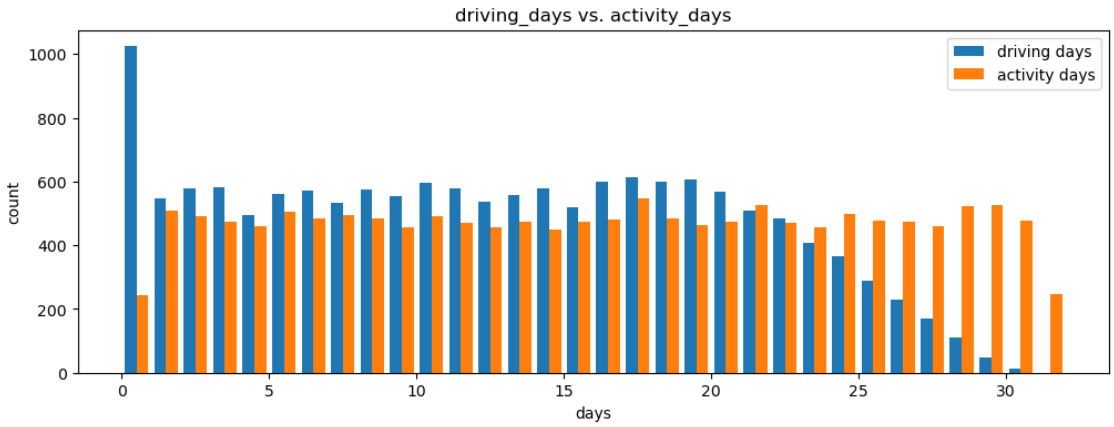
Less than 18% of the users churned.

driving_days vs. activity_days

Because both driving_days and activity_days represent counts of days over a month and they're also closely related, graph can be plotted on a single histogram. This will help to better understand how they relate to each other without having to scroll back and forth comparing histograms in two different places.

Plotting a histogram that, for each day, has a bar representing the counts of driving_days and user_days.

[26]:
```
# Histogram
plt.figure(figsize=(12,4))
label=['driving days', 'activity days']
plt.hist([df['driving_days'], df['activity_days']],
         bins=range(0,33),
         label=label)
plt.xlabel('days')
plt.ylabel('count')
plt.legend()
plt.title('driving_days vs. activity_days');
```



As observed previously, this might seem counterintuitive. After all, why are there fewer people who didn't use the app at all during the month and more people who didn't drive at all during the month?

On the other hand, it could just be illustrative of the fact that, while these variables are related to each other, they're not the same. People probably just open the app more than they use the app to drive—perhaps to check drive times or route information, to update settings, or even just by mistake.

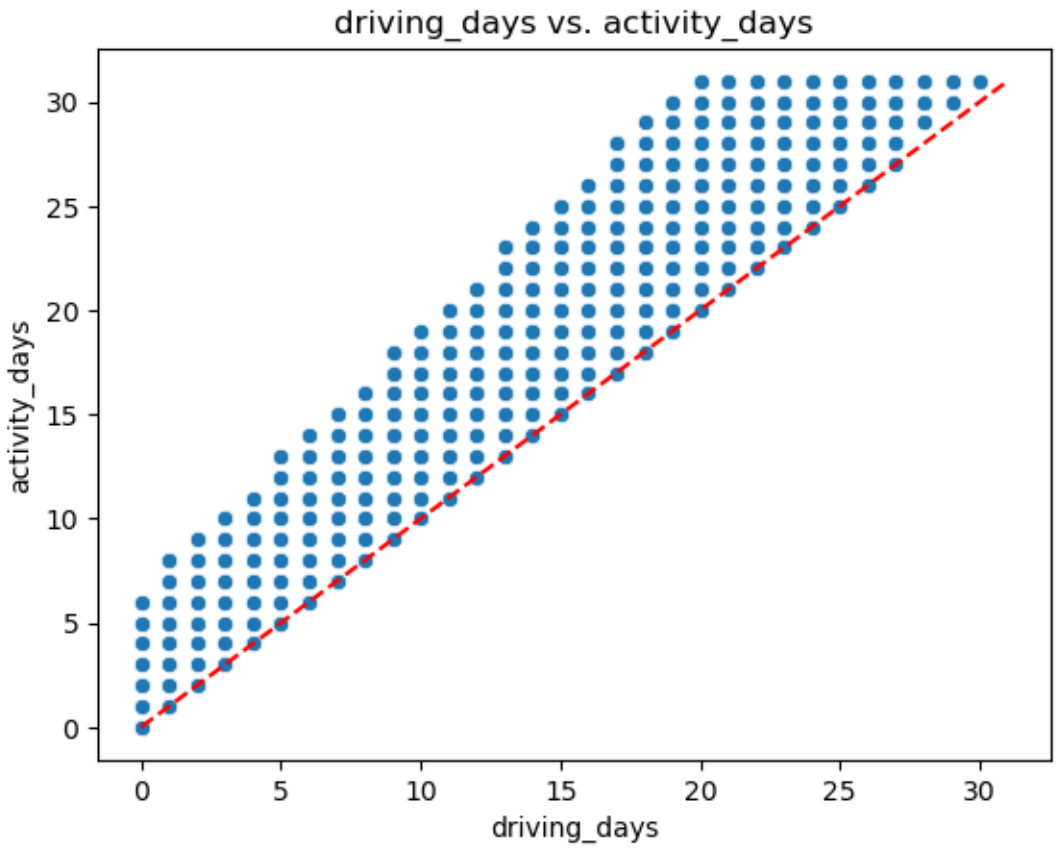
[27]:
```
print(df['driving_days'].max())
print(df['activity_days'].max())
```

```
30
31
```

It's true. Although it's possible that not a single user drove all 31 days of the month, it's highly unlikely, considering there are 15,000 people represented in the dataset.

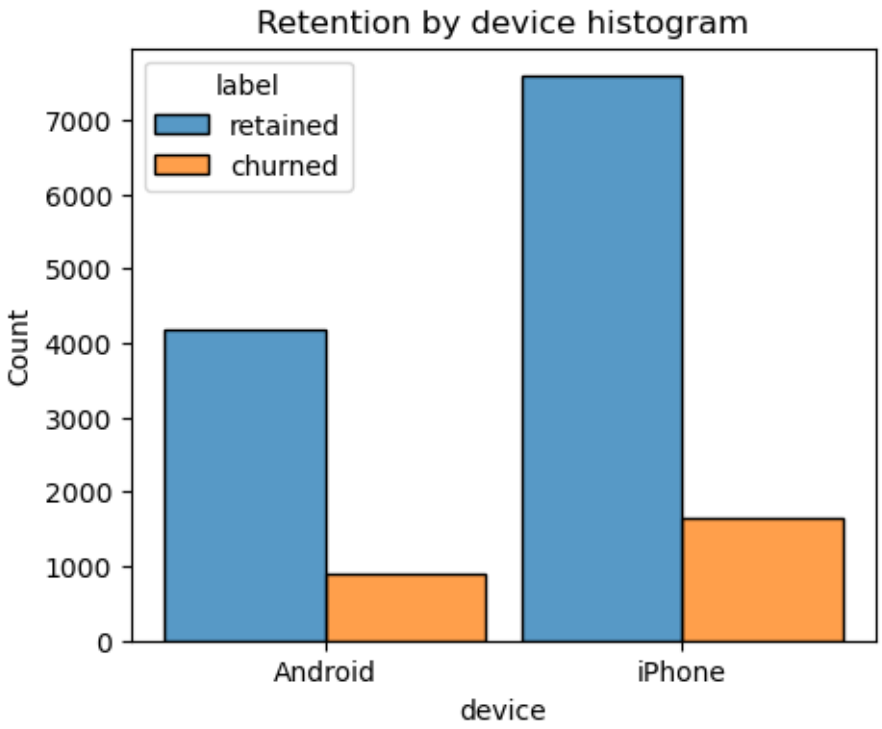
One other way to check the validity of these variables is to plot a simple scatter plot with the x-axis representing one variable and the y-axis representing the other.

[28]:
```
# Scatter plot
sns.scatterplot(data=df, x='driving_days', y='activity_days')
plt.title('driving_days vs. activity_days')
plt.plot([0,31], [0,31], color='red', linestyle='--');
```



Notice that there is a theoretical limit. If you use the app to drive, then by definition it must count as a day-use as well. In other words, you cannot have more drive-days than activity-days. None of the samples in this data violate this rule, which is good.

[29]:
```
## To check retention ny device

# Histogram
plt.figure(figsize=(5,4))
sns.histplot(data=df,
             x='device',
             hue='label',
             multiple='dodge',
             shrink=0.9
             )
plt.title('Retention by device histogram');
```



The proportion of churned users to retained users is consistent between device types.

[30]:
```
# 1. Create `km_per_driving_day` column
df['km_per_driving_day'] = df['driven_km_drives'] / df['driving_days']

# 2. Call `describe()` on the new column
df['km_per_driving_day'].describe()
```

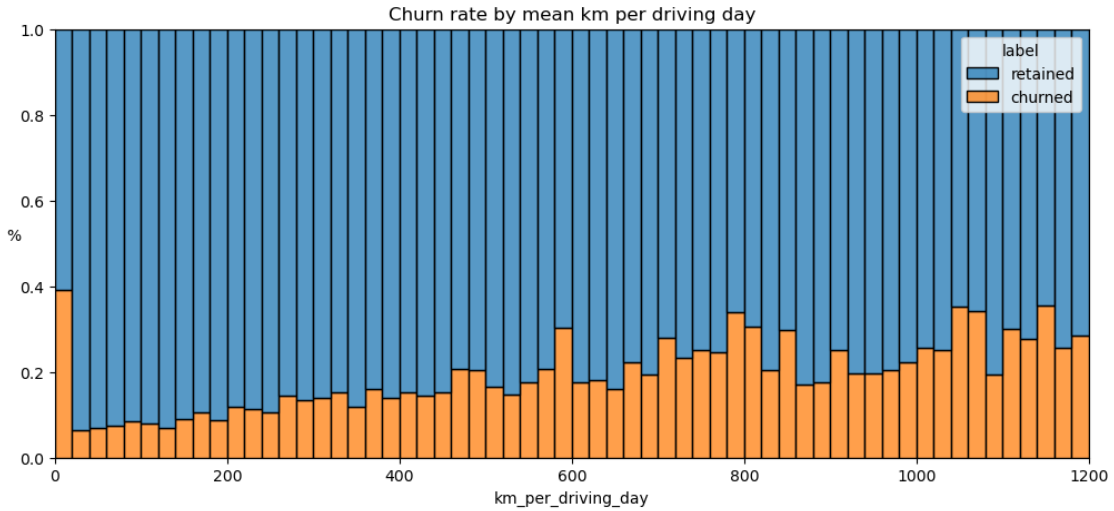
[30]:
```
count    1.499900e+04
mean              inf
std               NaN
min      3.022063e+00
25%      1.672804e+02
50%      3.231459e+02
75%      7.579257e+02
max               inf
Name: km_per_driving_day, dtype: float64
```

[31]:
```
# 1. Convert infinite values to zero
df.loc[df['km_per_driving_day']==np.inf, 'km_per_driving_day'] = 0

# 2. Confirm that it worked
df['km_per_driving_day'].describe()
```

[31]:
```
count    14999.000000
mean       578.963113
std       1030.094384
min          0.000000
25%        136.238895
50%        272.889272
75%        558.686918
max      15420.234110
Name: km_per_driving_day, dtype: float64
```

The maximum value is 15,420 kilometers per drive day. This is physically impossible. Driving 100 km/hour for 12 hours is 1,200 km. It's unlikely many people averaged more than this each day they drove, so, for now, disregard rows where the distance in this column is greater than 1,200 km.

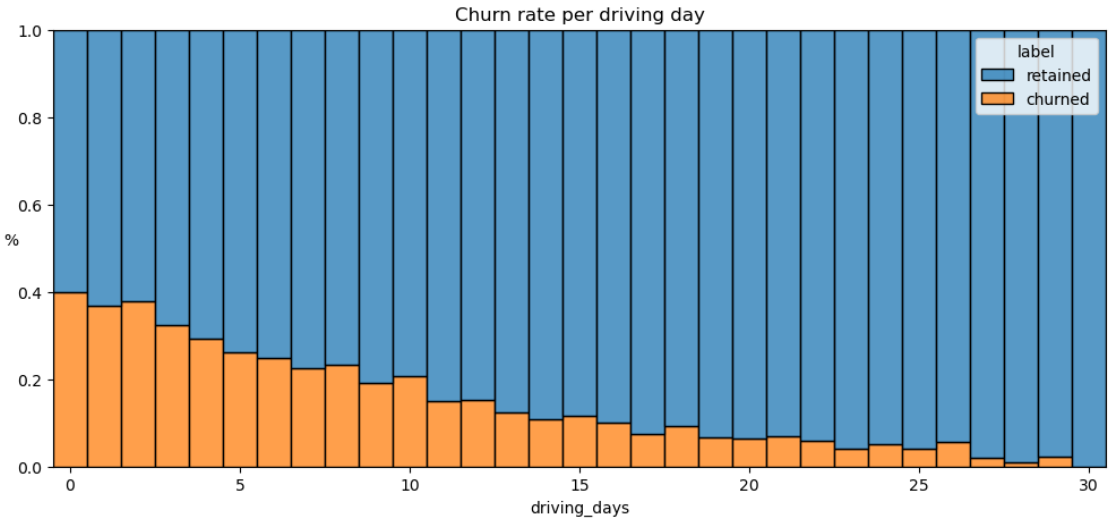
[32]:
```
# Histogram
plt.figure(figsize=(12,5))
sns.histplot(data=df,
             x='km_per_driving_day',
             bins=range(0,1201,20),
             hue='label',
             multiple='fill')
plt.ylabel('%', rotation=0)
plt.title('Churn rate by mean km per driving day');
```



The churn rate tends to increase as the mean daily distance driven increases, confirming what was found in the previous course. It would be worth investigating further the reasons for long-distance users to discontinue using the app.

[33]:
```
# Histogram
plt.figure(figsize=(12,5))
```

```
sns.histplot(data=df,
             x='driving_days',
             bins=range(1,32),
             hue='label',
             multiple='fill',
             discrete=True)
plt.ylabel('%', rotation=0)
plt.title('Churn rate per driving day');
```



The churn rate is highest for people who didn't use Waze much during the last month. The more times they used the app, the less likely they were to churn. While 40% of the users who didn't use the app at all last month churned, nobody who used the app 30 days churned.

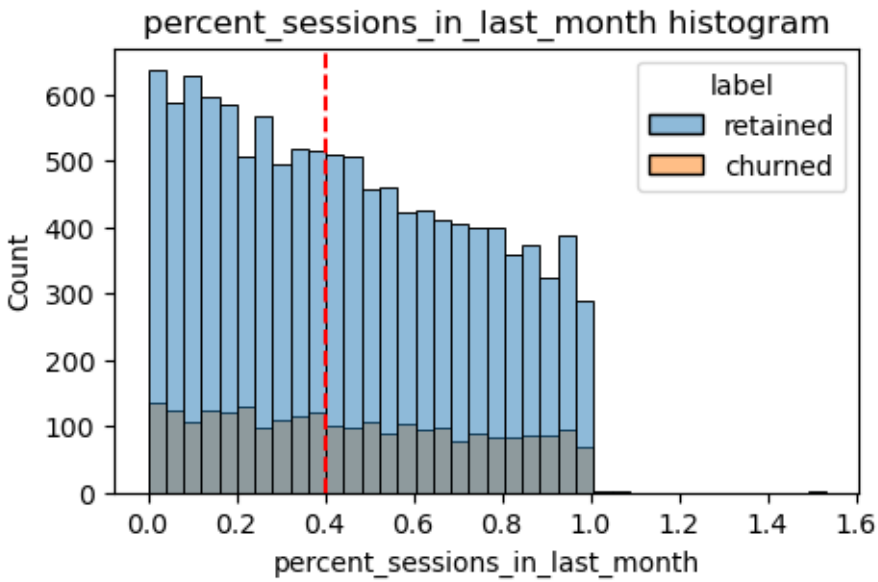
[34]:
```
df['percent_sessions_in_last_month'] = df['sessions'] / df['total_sessions']
```

[35]:
```
df['percent_sessions_in_last_month'].median()
```

[35]:
```
0.42309702992763176
```

[36]:
```
# Histogram
histogrammer('percent_sessions_in_last_month',
             hue=df['label'],
             multiple='layer',
             median_text=False)
```

```
Median: 0.4
```

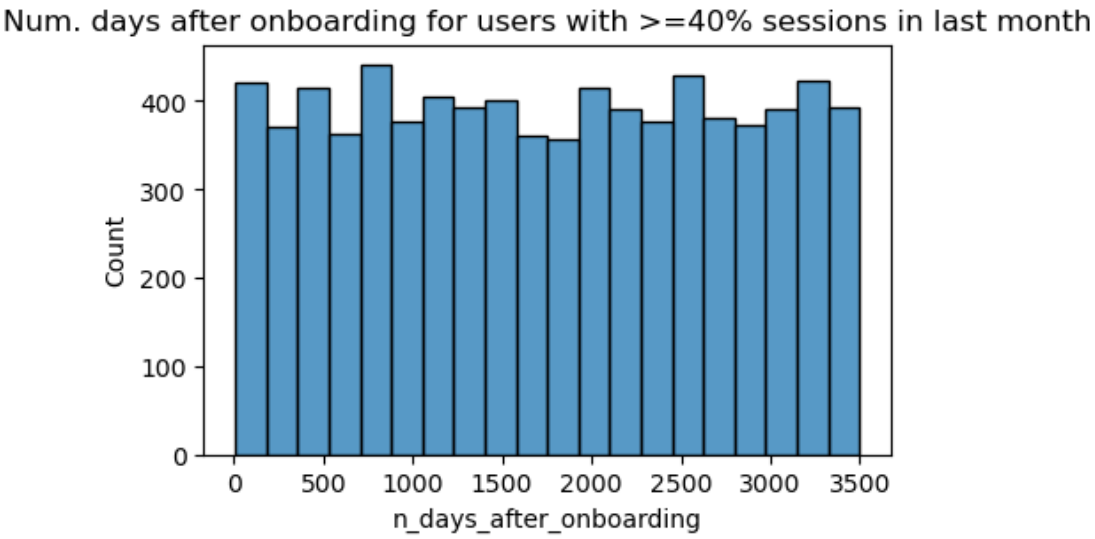


[37]:
```
df['n_days_after_onboarding'].median()
```

[37]:
```
1741.0
```

Half of the people in the dataset had 40% or more of their sessions in just the last month, yet the overall median time since onboarding is almost five years.

[38]:
```
# Histogram
data = df.loc[df['percent_sessions_in_last_month']>=0.4]
plt.figure(figsize=(5,3))
sns.histplot(x=data['n_days_after_onboarding'])
plt.title('Num. days after onboarding for users with >=40% sessions in last␣
 ↪month');
```



The number of days since onboarding for users with 40% or more of their total sessions occurring in just the last month is a uniform distribution. This is very strange. It's worth asking Waze why so many long-time users suddenly used the app so much in the last month.

[39]:
```
def outlier_imputer(column_name, percentile):
    # Calculate threshold
    threshold = df[column_name].quantile(percentile)
    # Impute threshold for values > than threshold
    df.loc[df[column_name] > threshold, column_name] = threshold

    print('{:>25} | percentile: {} | threshold: {}'.format(column_name,␣
 ↪percentile, threshold))
```

[40]:
```
for column in ['sessions', 'drives', 'total_sessions',
               'driven_km_drives', 'duration_minutes_drives']:
               outlier_imputer(column, 0.95)
```

```
                 sessions | percentile: 0.95 | threshold: 243.0
                   drives | percentile: 0.95 | threshold: 201.0
           total_sessions | percentile: 0.95 | threshold: 454.3632037399997
        driven_km_drives | percentile: 0.95 | threshold: 8889.7942356
  duration_minutes_drives | percentile: 0.95 | threshold: 4668.899348999998
```

[41]:
```
df.describe()
```

[41]:

	ID	sessions	drives	total_sessions	\
count	14999.000000	14999.000000	14999.000000	14999.000000	
mean	7499.000000	76.568705	64.058204	184.031320	
std	4329.982679	67.297958	55.306924	118.600463	

```
min        0.000000      0.000000      0.000000        0.220211
25%     3749.500000     23.000000     20.000000       90.661156
50%     7499.000000     56.000000     48.000000      159.568115
75%    11248.500000    112.000000     93.000000      254.192341
max    14998.000000    243.000000    201.000000      454.363204

       n_days_after_onboarding  total_navigations_fav1  \
count             14999.000000            14999.000000
mean               1749.837789              121.605974
std                1008.513876              148.121544
min                   4.000000                0.000000
25%                 878.000000                9.000000
50%                1741.000000               71.000000
75%                2623.500000              178.000000
max                3500.000000             1236.000000

       total_navigations_fav2  driven_km_drives  duration_minutes_drives  \
count            14999.000000      14999.000000             14999.000000
mean                29.672512       3939.632764              1789.647426
std                 45.394651       2216.041510              1222.705167
min                  0.000000         60.441250                18.282082
25%                  0.000000       2212.600607               835.996260
50%                  9.000000       3493.858085              1478.249859
75%                 43.000000       5289.861262              2464.362632
max                415.000000       8889.794236              4668.899349

       activity_days  driving_days  km_per_driving_day  \
count   14999.000000  14999.000000        14999.000000
mean       15.537102     12.179879          578.963113
std         9.004655      7.824036         1030.094384
min         0.000000      0.000000            0.000000
25%         8.000000      5.000000          136.238895
50%        16.000000     12.000000          272.889272
75%        23.000000     19.000000          558.686918
max        31.000000     30.000000        15420.234110

       percent_sessions_in_last_month
count                    14999.000000
mean                         0.449255
std                          0.286919
min                          0.000000
25%                          0.196221
50%                          0.423097
75%                          0.687216
max                          1.530637
```

Conclusion Analysis revealed that the overall churn rate is ~17%, and that this rate is consistent between iPhone users and Android users.