Git

May 7, 2018

1 Search Intelligence Executive Summary

- **Top Source: Windows, AdsBot-Goolge, iphone** : The Top traffic sources are by `Windows, AdsBot-Goolge, iphone, and Googlebot, Android, Macintosh, and iPad`, and the difference among each sources are very obvious. However, later in question two, after focusing on PPC model, the important source is not the biggest source (Windows), but mobile ('Android', 'iPad', 'iPhone')
- **Homepage & AdsBot-Google** : AdsBot-Google has the largest chance to strangely lands on this page than any other systems. Even though Windows lands on 2651 times, but it has larger number (4903 times) on page ID `2116`.
- **Bot doesn't crawl forwarding page** : There is no bot system showing records on PPC forwarding page
- **Mobile Device is good for Pay-per-click (PPC)** : Activities on `mobile devices` has higher traffic
- **Mobile activities look for closer departures** : ... and Desktop tends to be `right skew` (further departures)
- **Users mostly look for a certain type of room**
- **Most searches are for short booking (1 or 2 days)**

2 Question One

2.0.1 Loading and Cleaning

```
In [140]: import pandas as pd
          from dateutil.parser import parse
          import numpy as np
          from nltk.tokenize import TweetTokenizer,  word_tokenize
          import seaborn as sns
          import matplotlib.pyplot as plt
          %matplotlib inline
```

```
In [151]: # Reading user_agent information, and return the system info.
```

```
        def system(text):

            if text == "Carbon":
                return "Carbon"

            elif text == "AdsBot-Google (+http://www.google.com/adsbot.html)":
                return "AdsBot-Google"

            else:
                d= text.split('/')

                if d[0] == "Sogou web spider":
                    return "Sogou"

                elif d[0] == "WordPress":
                    return "WordPress"

                elif d[0] == "facebookexternalhit":
                    return "facebookexternalhit"

                else:
                    words = word_tokenize(d[1])

                    if words[2] == "compatible":

                        if words[4] is ")":
                            return ""
                        else:
                            if words[4] == "Linux" or words[4] == "X11":
                                return "Android"
                            else:
                                return words[4]

                    else:
                        if words[2] == "Linux" or words[2] == "X11":
                            return "Android"
                        else:
                            return words[2]
```

In [262]:
```
# Further categorize the system types into devices

        def device(input):
            bot = ['AdsBot-Google', 'BLEXBot', 'Carbon', 'Exabot',
                   'Facebot', 'Googlebot', 'MSIE', 'Sogou', 'WordPress',
                   'YandexBot', 'adidxbot', 'bingbot', 'coccocbot-web',
                   'facebookexternalhit']
            mobile = ['Android', 'iPad', 'iPhone']
            desktop = ['Macintosh','Windows']
```

```
    if input in mobile:
        return "mobile"
    elif input in desktop:
        return "desktop"
    else:
        return "bot"
```

In [152]:
```
# extraiting tagging information from target url, reconstructing them into a list

import sys

def url_sp(text):
    d = text.split('&')

    if len(d) > 1:
        try:
            k = d[0].split('?')
            nn = k[1].split('=')
            del d[0]
            doubled = [num.split('=') for num in d if '=' in num]
            return nn + sum(doubled, [])

        except:
            print(sys.exc_info()[0],"occured.")
            print(d)

    else:
        #print(text)
        k = d[0].split('?')
        if len(k) > 1:
            nn = k[1].split('=')
            return nn
```

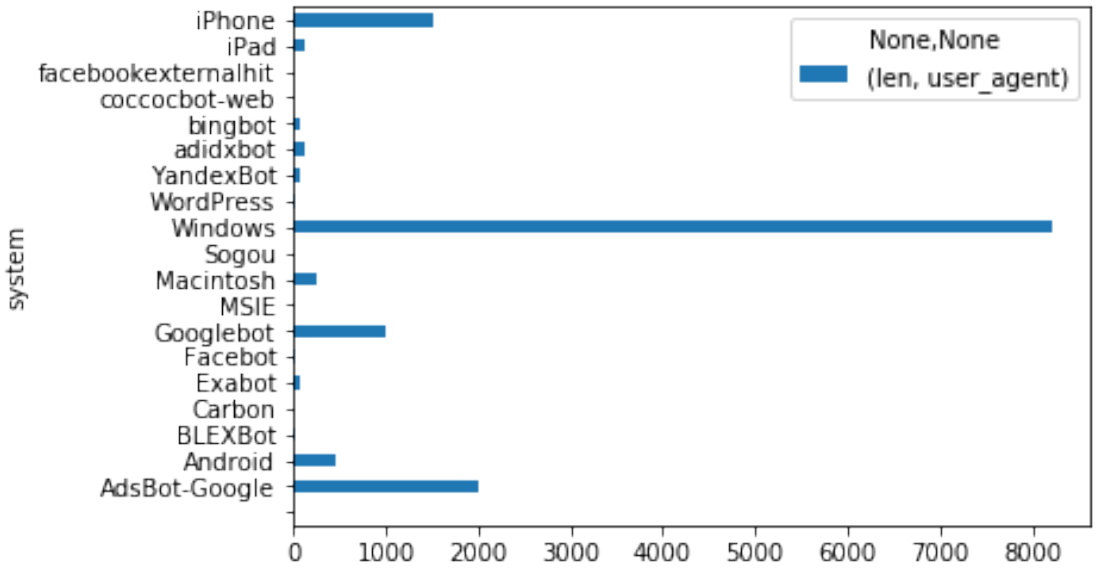
In [263]:
```
# read the data and add new attributes
data['time'] = data['@timestamp'].apply(lambda x: parse(x))
data['system'] = data['user_agent'].apply(lambda x: system(x))
data['trait'] = data['target_url'].apply(lambda x: url_sp(x))
data['device'] = data['system'].apply(lambda x: device(x))
```

2.0.2 System

After sorting and cleaning, we can see from the below compiling results that there are three types of devices, Desktop, Mobile, and Bot.

The Top traffic sources are by `Windows`, `AdsBot-Goolge`, `iphone`, and `Googlebot`, `Android`, `Macintosh`, and `iPad`, and the difference among each sources are very obvious. However, later in question two, after focusing on PPC model, the important source is not the biggest source (Windows), but mobile ('Android', 'iPad', 'iPhone')

```
In [261]: np.unique(data['system'])

Out[261]: array(['', 'AdsBot-Google', 'Android', 'BLEXBot', 'Carbon', 'Exabot',
                 'Facebot', 'Googlebot', 'MSIE', 'Macintosh', 'Sogou', 'Windows',
                 'WordPress', 'YandexBot', 'adidxbot', 'bingbot', 'coccocbot-web',
                 'facebookexternalhit', 'iPad', 'iPhone'], dtype=object)

In [7]: pd.pivot_table(data,index=["system"], values=["user_agent"],
                       aggfunc=[len]).plot(kind="barh")

Out[7]: <matplotlib.axes._subplots.AxesSubplot at 0x188dac8d400>
```



2.0.3 Page ID distribution by sources

The illustration of findings are shown by table and graph.

2100 & AdsBot-Google This is the homepage that has a search bar for user to enter their desired destination.

AdsBot-Google has the largest chance to strangely lands on this page than any other systems.

Even though Windows lands on 2651 times, but it has larger number (4903 times) on page ID 2116.

Bot doesn't crawl forwarding page There is no bot system showing records on 8001 page ID, but mostly landing onto 2111 - 2116.

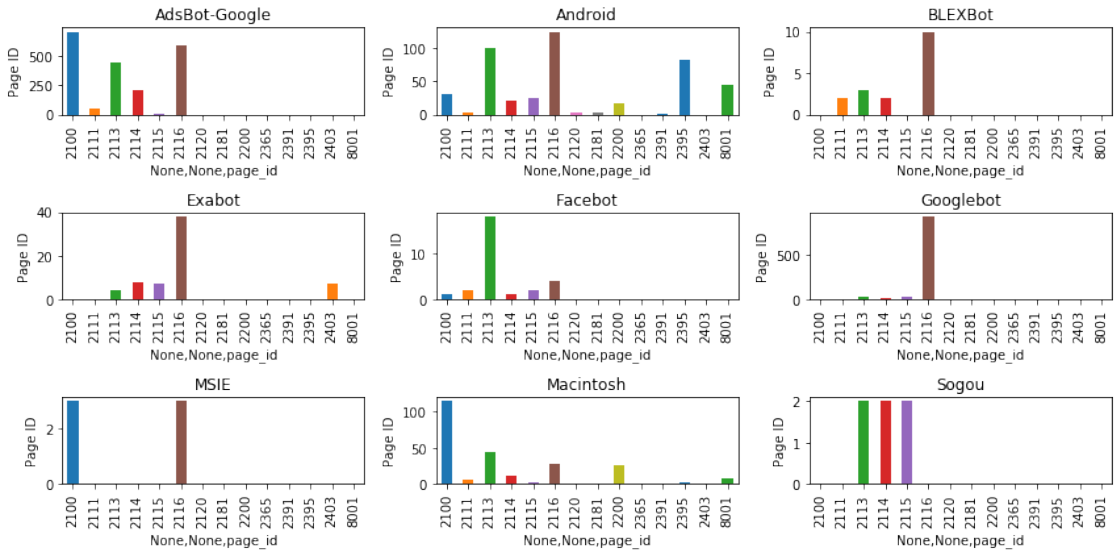
```
In [289]: pd.pivot_table(data,index=["system"], columns = ["page_id"],
                          values=["user_agent"], aggfunc=[len], margins=True
                         ).query('system != ["coccocbot-web", "facebookexternalhit"]')
```

Out[289]:

	len							
	user_agent							
page_id	2100	2111	2113	2114	2115	2116	2120	2181
system								
	2.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
AdsBot-Google	703.0	48.0	443.0	209.0	11.0	586.0	NaN	NaN
Android	30.0	4.0	100.0	21.0	24.0	125.0	4.0	4.0
BLEXBot	NaN	2.0	3.0	2.0	NaN	10.0	NaN	NaN
Carbon	1.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Exabot	NaN	NaN	4.0	8.0	7.0	38.0	NaN	NaN
Facebot	1.0	2.0	18.0	1.0	2.0	4.0	NaN	NaN
Googlebot	NaN	2.0	26.0	16.0	26.0	930.0	NaN	NaN
MSIE	3.0	NaN	NaN	NaN	NaN	3.0	NaN	NaN
Macintosh	114.0	6.0	45.0	11.0	3.0	28.0	NaN	NaN
Sogou	NaN	NaN	2.0	2.0	2.0	NaN	NaN	NaN
Windows	2651.0	11.0	137.0	18.0	41.0	4903.0	NaN	NaN
WordPress	NaN	NaN	8.0	NaN	NaN	NaN	NaN	NaN
YandexBot	NaN	NaN	1.0	1.0	20.0	38.0	NaN	NaN
adidxbot	3.0	6.0	52.0	11.0	11.0	41.0	NaN	NaN
bingbot	NaN	NaN	5.0	5.0	5.0	50.0	NaN	NaN
iPad	2.0	1.0	45.0	11.0	15.0	11.0	NaN	NaN
iPhone	27.0	79.0	455.0	192.0	43.0	624.0	NaN	NaN
All	3537.0	161.0	1344.0	508.0	210.0	7393.0	4.0	4.0

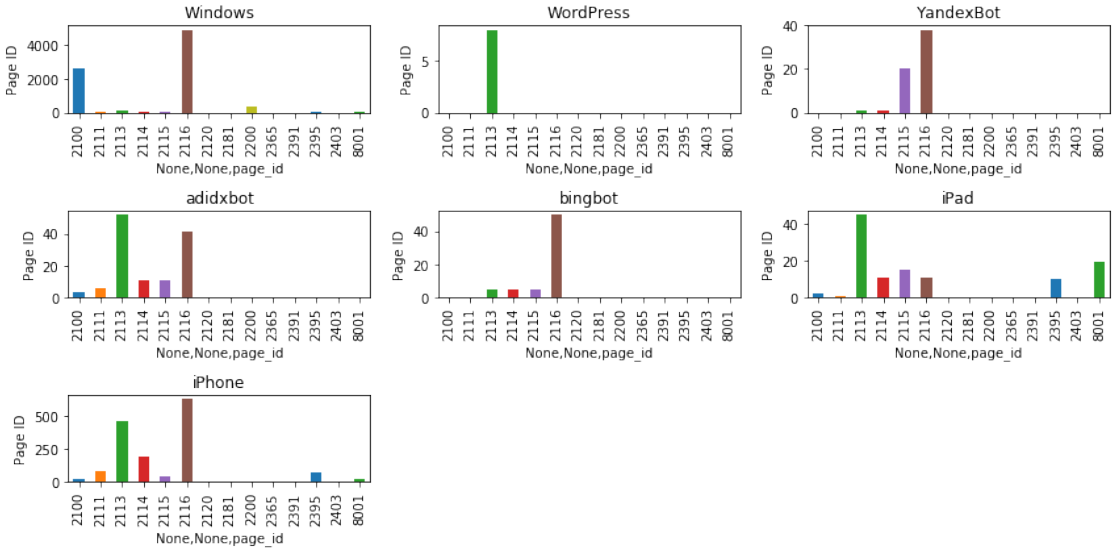
page_id	2200	2365	2391	2395	2403	8001	All
system							
	NaN	NaN	NaN	NaN	NaN	NaN	2
AdsBot-Google	NaN	NaN	NaN	NaN	NaN	NaN	2000
Android	17.0	NaN	1.0	83.0	NaN	44.0	457
BLEXBot	NaN	NaN	NaN	NaN	NaN	NaN	17
Carbon	NaN	NaN	NaN	NaN	NaN	NaN	1
Exabot	NaN	NaN	NaN	NaN	7.0	NaN	64
Facebot	NaN	NaN	NaN	NaN	NaN	NaN	28
Googlebot	NaN	NaN	NaN	NaN	NaN	NaN	1000
MSIE	NaN	NaN	NaN	NaN	NaN	NaN	6
Macintosh	26.0	NaN	NaN	3.0	NaN	8.0	244
Sogou	NaN	NaN	NaN	NaN	NaN	NaN	6
Windows	379.0	1.0	1.0	20.0	NaN	57.0	8219
WordPress	NaN	NaN	NaN	NaN	NaN	NaN	8
YandexBot	NaN	NaN	NaN	NaN	NaN	NaN	60
adidxbot	NaN	NaN	NaN	NaN	NaN	NaN	124
bingbot	NaN	NaN	NaN	NaN	NaN	NaN	65
iPad	NaN	NaN	NaN	10.0	NaN	19.0	114
iPhone	NaN	NaN	NaN	69.0	NaN	28.0	1517
All	422.0	1.0	2.0	185.0	7.0	156.0	13934

In [135]:
```
fig = plt.figure(figsize=(12, 6))
# 4 x 5 (3 x 3)

pd_table = pd.pivot_table(data,index=["system"],
             columns = ["page_id"], values=["user_agent"], aggfunc=[len]
            ).query('system != ["Carbon", "coccocbot-web", "facebookexternalhit"]')

for num in range(1,10):

    xtick = pd_table.iloc[num,].index.levels[:][2].tolist()
    name = pd_table.iloc[num,].name

    position = int('33'+ str(num) )
    ax = fig.add_subplot(position)
    pd_table.iloc[num,].plot(kind=' bar',
    title = name).set_xticklabels(xtick)
    plt.ylabel('Page ID')

plt.tight_layout()
plt.show()
```



In [138]:
```
fig = plt.figure(figsize=(12, 6))

for num in range(1,8):
    nu = num + 9
    xtick = pd_table.iloc[nu,].index.levels[:][2].tolist()
    name = pd_table.iloc[nu,].name

    position = int('33'+ str(num) )
```

```
        ax = fig.add_subplot(position)
        pd_table.iloc[nu,].plot(kind=' bar', title = name).set_xticklabels(xtick)
        plt.ylabel('Page ID')

    plt.tight_layout()
    plt.show()
```



3 Question Two

```
In [266]: #Slicing the data - only page_id in 8001

          data_8001 = data[data['page_id']== 8001]
          data_8001['count'] = 1
```

```
C:\ProgramData\Anaconda3\lib\site-packages\ipykernel_launcher.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html
```

3.1 Which Device is good for Pay-per-click (PPC)?

Given the target url, we can see the page ID 8001 is the forwarding page to hoteliers which bring PPC value. THerefore

- mobile includes 'Android', 'iPad', and 'iPhone'

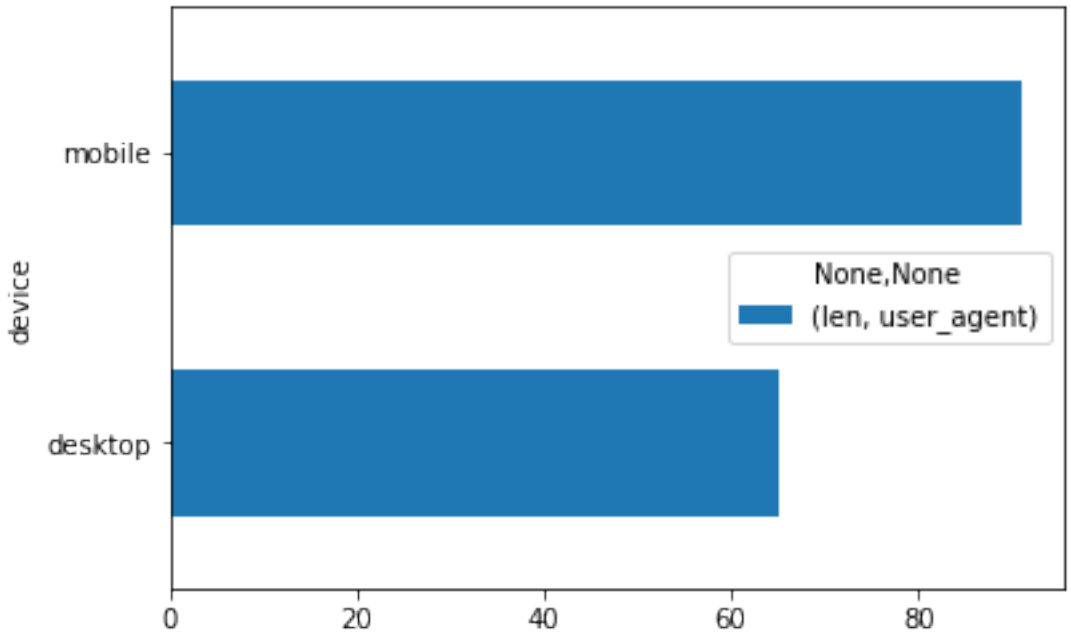
- desktop includes 'Macintosh', and 'Windows'
- rest is regarded as 'bot'

According to the below plot, activities on `mobile devices` has higher traffic.

Since the characteristics of this page type, it is understandable that `there is no roaming bot records`.

```
In [286]: pd.pivot_table(data_8001,index=["device"],
                         values=["user_agent"], aggfunc=[len]
                        ).plot(kind="barh")
```

```
Out[286]: <matplotlib.axes._subplots.AxesSubplot at 0x188dfaca9b0>
```



3.2 Aggregating data and insert new attribute

```
In [268]: # return the position of a specific string in a list
          def find(line, word):
              try:
                  k = [ i for i,x in enumerate(line) if x == word]

              except:
                  print(sys.exc_info()[0],"occured.")
                  print(word)
                  print(line)
              if len(k) is 0:
                  return True
```

```
            else:
                return k[0]

        # to be used in lamda to return the value by the key <- input(word)
        def query(dat, word):
            data_8001[word] = data_8001['trait'].apply(lambda x: x[1 + find(x, word)])
```

In [269]:
```
        # keyword tag in track url
        attribute = ['arrival', 'departure', 'path', 'room_type', 'cpt', 'clTi', 'mode']

        # insert new attributes
        for i in attribute:
            query(data_8001, i)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\ipykernel_launcher.py:15: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html
  from ipykernel import kernelapp as app
```

In [270]:
```
        # calculate the search travel length and
        # Calculate the difference between "search" and "arrival time"

        L1 = []
        L2 = []

        for nc in data_8001.index:
            t1 = parse(data_8001['arrival'][nc])
            t2 = parse(data_8001['departure'][nc])
            t3 = data_8001['time'][nc]
            diff1 = t2 - t1
            diff2 = t1 - t3
            L1.append(diff1.days)
            L2.append(diff2.days)

        data_8001['travel_length'] = L1
        data_8001['Depart_diff'] = L2
```

```
C:\ProgramData\Anaconda3\lib\site-packages\ipykernel_launcher.py:13: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html
  del sys.path[0]
C:\ProgramData\Anaconda3\lib\site-packages\ipykernel_launcher.py:14: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead
```

```
See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html
```

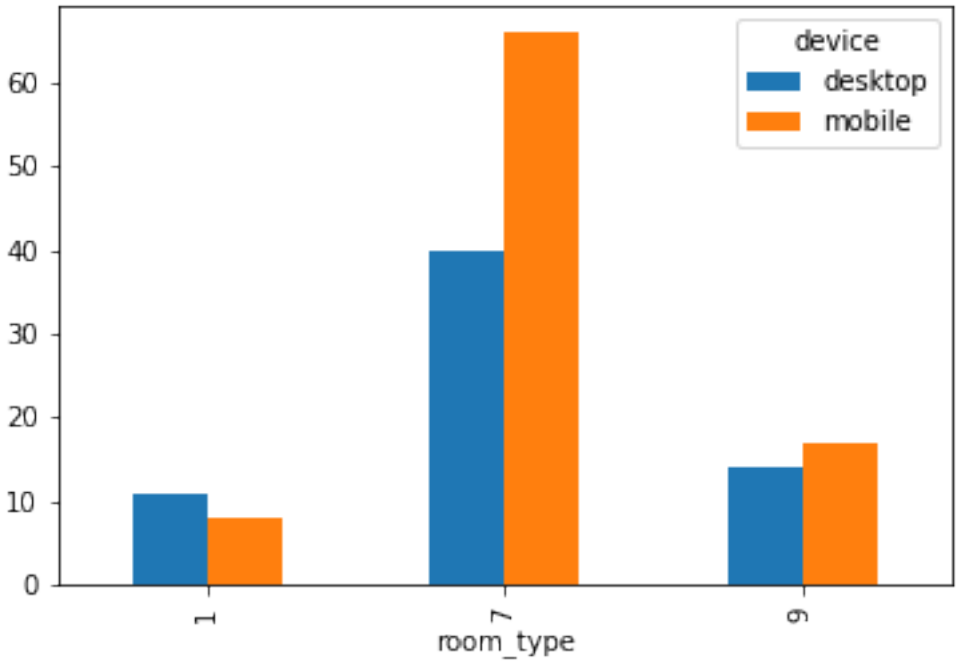
3.3 Results: What are the popular choices among users?

3.3.1 Room Type

Number 7 is the most popular room type, both in desktop or mobile.

```
In [272]: data_8001.pivot_table('count', index=['room_type'],
                                 columns='device', aggfunc='sum').plot(kind ='bar')
```

```
Out[272]: <matplotlib.axes._subplots.AxesSubplot at 0x188e21a7048>
```



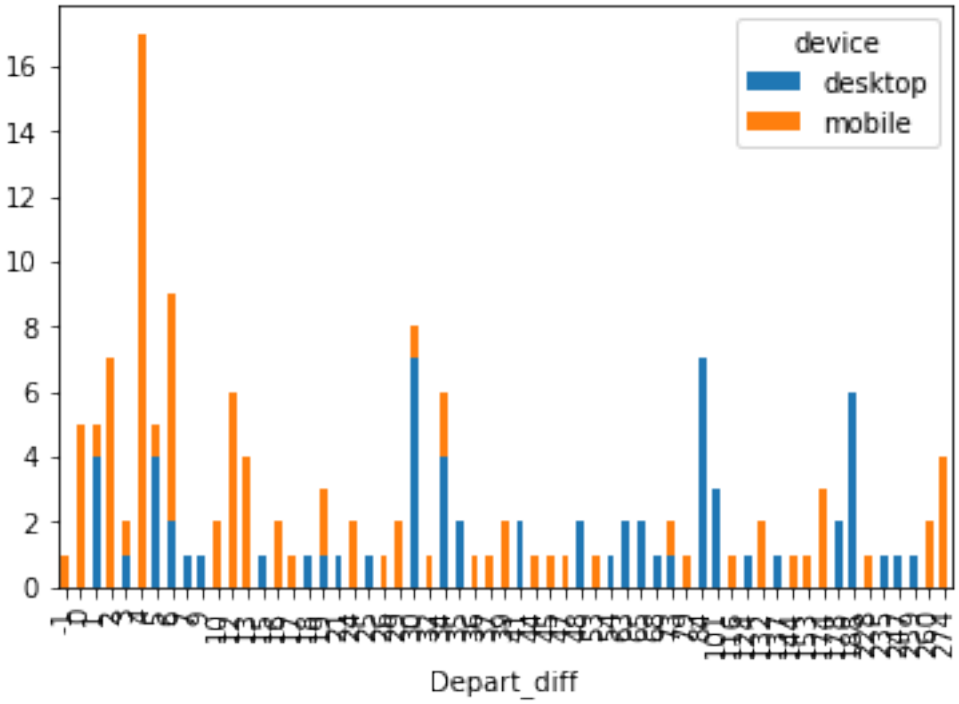
3.3.2 the difference between "search" and "arrival time"

It has the same result that mobile activities is larger than desktop ones.
Moreover, we can find two distinct characteristics:

- mobile activities is `left skew` (closer departures)
- desktop tends to be `right skew` (further departures)

```
In [278]: data_8001.pivot_table('count', index=['Depart_diff'],
                                 columns='device', aggfunc='sum'
                                ).plot(kind ='bar', stacked=True)
```

```
Out[278]: <matplotlib.axes._subplots.AxesSubplot at 0x188dd2494a8>
```

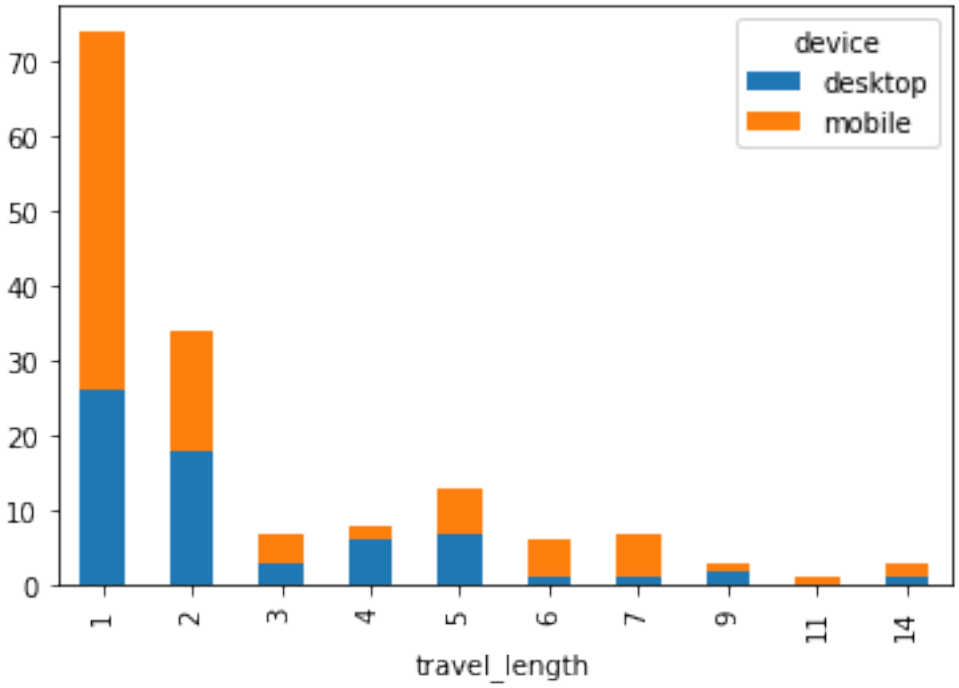


3.3.3 Prefered Stay Length

It is show obvious left skew distribution, showing that users prefers to look for 1 or 2 days bookings.

```
In [279]: data_8001.pivot_table('count', index=['travel_length'],
                                columns='device', aggfunc='sum'
                               ).plot(kind ='bar', stacked=True)
```

```
Out[279]: <matplotlib.axes._subplots.AxesSubplot at 0x188e1a9b860>
```


device
desktop
mobile
70
60
50
40
30
20
10
0
1
2
3
4
5
6
7
9
11
14
travel_length