Amazon Baby Products Sentiment Analysis

In [1]:

```
from IPython.display import Image
Image(filename ='C:/amazon logo.png')
```

Out[1]:

I. Introduction

```
        As per one survey, 72% of consumers will take action only af
ter reading a positive review.
```

As a frequent Amazon user, I have always been curious about the online review system and how do they impact any consumer's purchasing decision. With this work I am interested in exploring the structure of a large database of Amazon reviews and analysing this information through effective visualization so as to be a smarter consumer as well as reviewer. For the sake of simplicity I am going to limit my analysis to only Amazon baby dataset.

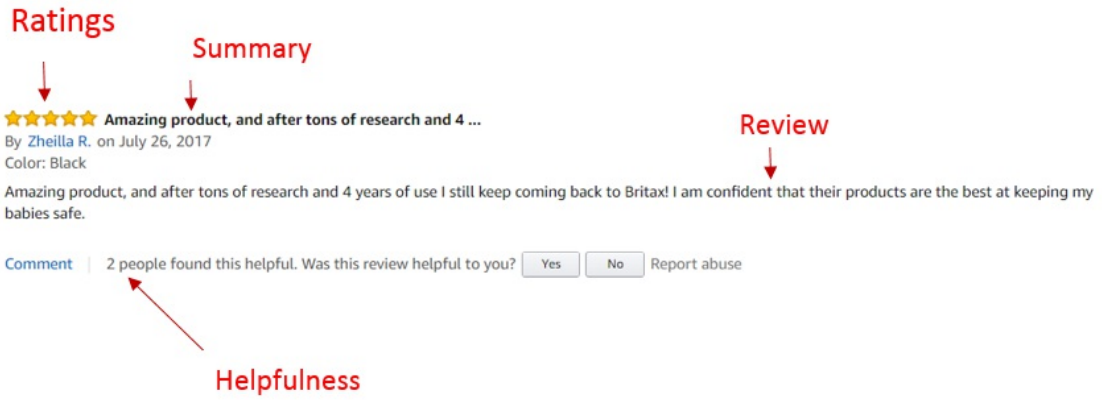
Below is a sample Amazon review. It consists of the following information:

- Rating (1 - 5 stars)
- The review
- A summary of the review
- The number of people who have voted if this review is helpful or not.

In [2]:

```
from IPython.display import Image
Image(filename ='C:/Amazon Rev.png')
# ![image](imagename.png "Title")
```

Out[2]:



II. Objective

I am going to use a data of over 59,000 reviews of Amazon Baby products that is available via this link here. http://jmcauley.ucsd.edu/data/amazon/links.html. This database contains 19 different features along with each of the elements of a review pictured above. So our initial goals would be to

- Perform some basic exploratory data analysis to better understand reviews.
- What are the properties of helpful reviews?
- How reviewText correlate to overall ratings.

In [3]:

```
%matplotlib inline
import pandas as pd
import numpy as np
from wordcloud import WordCloud , STOPWORDS

from nltk.corpus import stopwords
from nltk import word_tokenize

from string import punctuation

from sklearn.cross_validation import train_test_split, cross_val_score
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn import svm
from sklearn.linear_model import LogisticRegression
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.decomposition import TruncatedSVD
from sklearn.metrics import precision_score, confusion_matrix
from textblob import TextBlob
from sklearn.metrics import f1_score
from sklearn import metrics

import matplotlib.pyplot as plt
# from gensim.models import Word2Vec
import itertools
import seaborn as sns
import missingno as msno
```

```
import pandasql as pdsql
from pandasql import sqldf
```

```
C:\Users\anands\AppData\Local\Continuum\Anaconda3\lib\site-packages\sklearn\cross_validation.py:44: DeprecationWarning: This module was deprecated in version 0.18 in favor of the model_selection module into which all the refactored classes and functions are moved. Also note that the interface of the new CV iterators are different from that of this module. This module will be removed in 0.20.
  "This module will be removed in 0.20.", DeprecationWarning)
```

III. Reading Data

In [4]:

```
df = pd.read_csv('C:\Baby_review.csv',low_memory=False)
```

In [5]:

```
pd.read_csv('C:\Baby_review.csv',low_memory=False)
```

Out[5]:

	reviewerID	asin	reviewerName	helpful	helpful_num	helpful_den
0	A3NMPMELAZC8ZY	097293751X	Jakell	[3, 3]	3	3
1	A3O4ATU0ENBKTU	097293751X	MAPN	[1, 1]	1	1
2	A2SYNL4YX73KNY	097293751X	R. Davidson "Jrdpa"	[2, 2]	2	2
3	A2Q2A6JKY95RTP	097293751X	R. Garrelts	[2, 2]	2	2
4	A21I33AWNOWMK8	9729375011	EmilyS	[1, 2]	1	2
5	ALRN58JO86V5E	9729375011	John Ramahlo Jr.	[2, 3]	2	3

	reviewerID	asin	reviewerName	helpful	helpful_num	helpful_den
6	AXEEHEUKQILR0	9729375011	K. Kadrmas	[2, 2]	2	2
7	A3KZ91O8KA1IAZ	9729375011	LMCR "HauteDiva"	[1, 2]	1	2
8	A1OG2X4KR2U1GE	9729375011	M&M	[1, 1]	1	1
9	A3EG1F4UBURE6O	9729375011	Maria M. Shaw	[6, 6]	6	6
10	A2BGDV09HCHX8E	9729375011	msb175 "msb"	[0, 1]	0	1
11	A10STX704CY2IH	9729375011	Rangers68	[2, 3]	2	3
12	A36OZM5CENQ7LW	B00000IZQI	Allison	[1, 1]	1	1
13	A1Z54EM24Y40LL	B00000IZQI	csm	[17, 18]	17	18
14	AZK9CCTYQNRNL	B00000IZQI	Deanna Hockey	[2, 2]	2	2
15	API5JLBRT6FIN	B00000IZQI	KNSudha	[7, 7]	7	7
16	A1DNWA98IV5PMW	B00000IZQI	Madison and Nathan's Mom	[2, 2]	2	2

17	AFM7DWO762149	B00000IZQI	S. Hughes "s_hughes"	[98, 100]	98	100
	reviewerID	**asin**	**reviewerName**	**helpful**	**helpful_num**	**helpful_den**
18	A3DETJ3SIGHPWI	B00000IZQI	Sue	[7, 7]	7	7
19	A3NVSUFF2RPXOV	B00000IZQI	William Moor "William Moor"	[1, 1]	1	1
20	A30H2335OM7RD6	B00000J3LL	apoem "apoem"	[5, 5]	5	5
21	A2EP68HH1PXAN3	B00000J3LL	Tamara	[1, 1]	1	1
22	A3NKRXQLI3FQML	B00002JV9S	azgal	[1, 1]	1	1
23	A9RTFYY13I0GP	B00002JV9S	Buzzbee	[1, 1]	1	1
24	A3B19AL2Q4KZAG	B00002JV9S	Cece "Not So Usual"	[3, 4]	3	4
25	A269WG8C9O2B0C	B00002JV9S	Gee Geronimo	[1, 1]	1	1
26	A37KH5N4TE0RPR	B00002JV9S	LisaD	[1, 2]	1	2
27	A14RUX9GCBX49B	B00002JV9S	Rebecca Tucker	[0, 2]	0	2
28	A2QFBZKATFQCHC	B00002JV9S	RGillund	[4, 4]	4	4

28	A2QEPZKAYEOCOG	B00002JV9S	KGillund	[4, 4]	4	4
	reviewerID	**asin**	**reviewerName**	**helpful**	**helpful_num**	**helpful_den**
29	A3IUDIPEMB2EE8	B00002JV9S	TwinMom	[54, 57]	54	57
...	...	...	...	...	...	...
56920	A383CVDQ66BP86	B00JLHWDO4	Brookel	[5, 9]	5	9
56921	A3045OYEAMGB7Z	B00JLHWDO4	Elise	[1, 1]	1	1
56922	A1Y0G2WBHTXPHY	B00JLHWDO4	K. Fowler "Mom of twins"	[3, 3]	3	3
56923	A3UQW8PYBBQI9P	B00JLHWDO4	RSD	[3, 5]	3	5
56924	A2XNKMGEYHLUK7	B00JLI73ZM	Amazon Customer	[2, 5]	2	5
56925	A27JH7C18JQIJB	B00JLI73ZM	Angie	[2, 3]	2	3
56926	A3AOVTNCJ73WZU	B00JLI73ZM	CNye	[3, 4]	3	4
56927	A37VACOJ7WWJR4	B00JLI73ZM	Dani A.	[1, 6]	1	6
56928	A1HKBA28E0BZF8	B00JLI73ZM	Leaping Trout	[1, 3]	1	3

	reviewerID	asin	reviewerName	helpful	helpful_num	helpful_den
56929	A2D5LCNAIW2BP7	B00ULI73ZM	Mamafeesa	[20, 21]	20	21
56930	A2RIAXGSBP65BJ	B00JRYRYS6	Amanda Hamm "writer"	[2, 3]	2	3
56931	A2RX62V4E2BF5Z	B00JRYRYS6	Celeste "Vodka, Apple Pucker & Sweet 'n' Sour"	[1, 1]	1	1
56932	A268QM4AOCO9NI	B00JRYRYS6	C. Weaver "PsychoDoc"	[3, 3]	3	3
56933	A2ME89MSWVG9NF	B00JRYRYS6	donny "don130"	[1, 1]	1	1
56934	A3C1QYGEET3BVY	B00JRYRYS6	PC Mountain	[1, 2]	1	2
56935	A3IMK08UX0I46A	B00JRYRYS6	TOPJOB7 "topjob7"	[1, 1]	1	1
56936	A1ZILONLWX15N2	B00L13XFIE	Ally	[1, 1]	1	1
56937	A2K0QLST7946WU	B00L13XFIE	AQBoston "Allie"	[1, 1]	1	1
56938	A1HH6P7G3GH9EL	B00L13XFIE	BtrflyBlueStar	[6, 6]	6	6
56939	A17SL464CLQAT8	B00L13XFIE	ca.roybal	[3, 5]	3	5

	reviewerID	asin	reviewerName	helpful	helpful_num	helpful_den
56940	A32IUKDAS4THJD	B00L13XFIE	Danielle Warren	[2, 4]	2	4
56941	A2RDQBFPIJXNOW	B00L13XFIE	Fiona'sMom	[16, 18]	16	18
56942	A3GJJZV6K0F7IY	B00L13XFIE	GreenEyedGirl	[2, 2]	2	2
56943	A1URMXEEWEQR6V	B00L13XFIE	Jeremiah L. Olson	[2, 3]	2	3
56944	A2VCMK1USHIEL9	B00L13XFIE	Marcela	[1, 1]	1	1
56945	A34T0JYVRU1M2B	B00L13XFIE	Nukke	[1, 1]	1	1
56946	A1AFNMTDISXUJE	B00L13XFIE	Qutie	[1, 2]	1	2
56947	A1ZS6UQ9RVUX97	B00L13XFIE	R. Nafziger	[1, 2]	1	2
56948	AG4E44KM93P4L	B00L13XFIE	Silofish	[0, 1]	0	1
56949	A3CIIOMK18CHXM	B00L13XFIE	Viviana	[1, 1]	1	1

56950 rows × 19 columns

In [6]:

```
df.shape
```

Out[6]:

```
(56950, 19)
```

In [7]:

```
df.columns.tolist()
```

Out[7]:

```
['reviewerID',
 'asin',
 'reviewerName ',
 'helpful',
 'helpful_num',
 'helpful_den',
 'reviewText',
 'overall',
 'summary',
 'unixReviewTime',
 'reviewTime',
 'exclamationcount',
 'questioncount',
 'charcount',
 'wordcount',
 'capcount',
 'avgrating',
 'diffrating',
 'ishelpful']
```

In [8]:

```
df.head(2)
```

Out[8]:

	reviewerID	asin	reviewerName	helpful	helpful_num	helpful_den	reviewT
0	A3NMPMELAZC8ZY	097293751X	Jakell	[3, 3]	3	3	This boo is perfec I'm a firs time nev mo...
1	A3O4ATU0ENBKTU	097293751X	MAPN	[1, 1]	1	1	I use thi so that c babysitt (grandm ca...

In [9]:

```
df.sort_values(by='overall', ascending=False)[:3]
```

Out[9]:

	reviewerID	asin	reviewerName	helpful	helpful_num	helpful_den	re
0	A3NMPMELAZC8ZY	097293751X	Jakell	[3, 3]	3	3	T is I' ti m
31934	A37QWKCN6XQV1I	B003FLLQX6	K Bosh	[1, 1]	1	1	B fa b c ra
31939	A1ER8RH6UFTD1W	B003FLLQXG	Melanie Cintron	[10, 11]	10	11	T is T o it.

In [10]:

```
df.describe()
```

Out[10]:

	helpful_num	helpful_den	overall	unixReviewTime	exclamationcount	questi
count	56950.000000	56950.000000	56950.000000	5.695000e+04	56950.000000	56950.
mean	4.939701	5.987199	3.951975	1.329489e+09	0.879385	0.1024
std	23.655852	24.799907	1.308374	7.046482e+07	1.814582	0.4932
min	0.000000	1.000000	1.000000	9.824544e+08	0.000000	0.0000
25%	1.000000	1.000000	3.000000	1.306454e+09	0.000000	0.0000
50%	1.000000	2.000000	4.000000	1.353802e+09	0.000000	0.0000
75%	3.000000	4.000000	5.000000	1.376006e+09	1.000000	0.0000
max	1206.000000	1214.000000	5.000000	1.405987e+09	74.000000	18.000

IV. Exploratory Data Analysis

IV.I- Distribution of overall ratings

In [11]:

```
df['overall'].value_counts()
```

Out[11]:

```
5    28368
4    11717
```

```
3      7463
1      4836
2      4566
Name: overall, dtype: int64
```

In [12]:

```
df1 = df.groupby(['overall']).agg({'reviewerID': 'count'})
df1['% of total']= df1['reviewerID']/df1['reviewerID'].sum()*100
# df1['% of total'] = df1['% of total'].map('{:,.2f}%'.format)
df1
```

Out[12]:

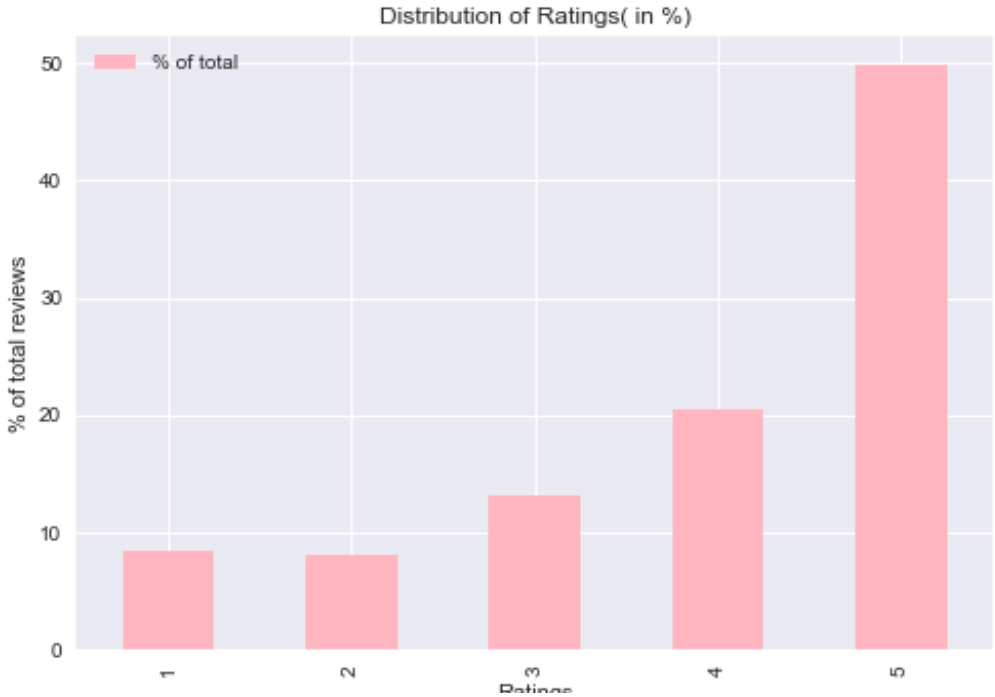
	reviewerID	% of total
overall		
1	4836	8.491659
2	4566	8.017559
3	7463	13.104478
4	11717	20.574188
5	28368	49.812116

In [13]:

```
df1.reset_index(level = 0 , inplace = True)
df1
s = df1[['overall','% of total']]
s.set_index('overall', inplace = True)
s.plot(kind = 'bar', color = 'Lightpink')
plt.xlabel("Ratings")
plt.ylabel("% of total reviews")
plt.title("Distribution of Ratings( in %)")
```

Out[13]:

```
<matplotlib.text.Text at 0xe581cc0>
```



Ratings

Looking at the distribution of ratings, we see that 5-star reviews constitute a large proportion (50%) of all reviews. The next most prevalent rating is 4-stars(21%), followed by 3-star (13%), 1-star (8.5%), and finally 2-star reviews (8.0%).

IV.II- Reviews and their helpfulness

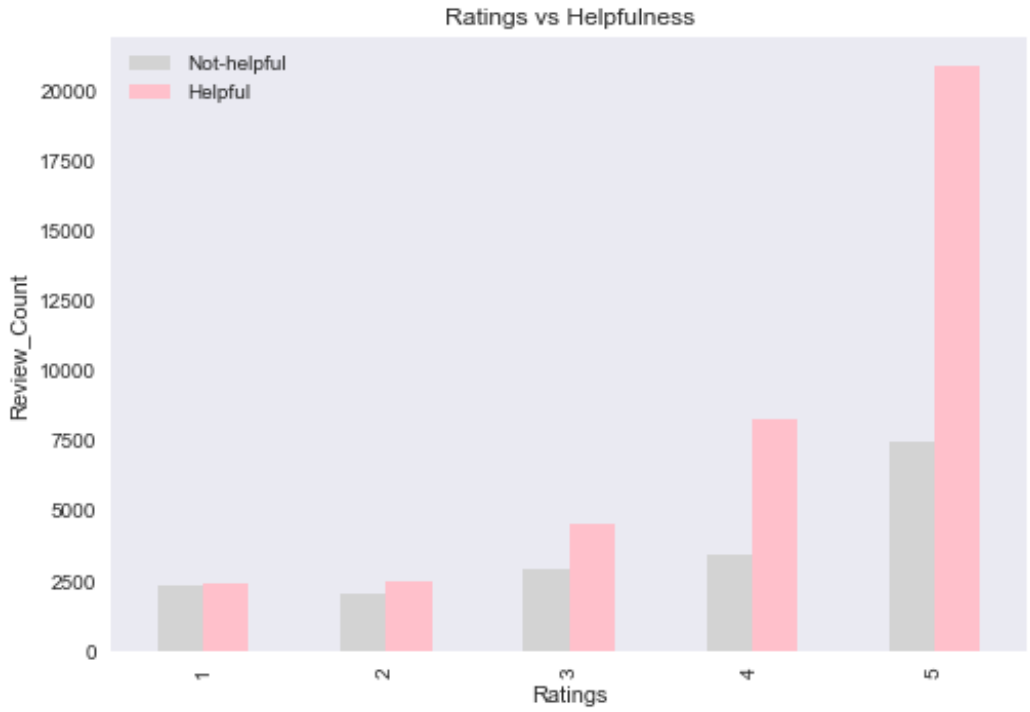
- Reviews are voted upon based on how helpful other reviewers find them. The most helpful reviews appear near the top of the list of reviews and are hence more visible. As such, I was interested in exploring the properties of helpful reviews.

In [14]:

```
t2 = pd.crosstab(df['overall'], df['ishelpful'])
t2.columns = ["Not-helpful","Helpful"]
t2.index = [1,2,3,4,5]

t2.plot(kind='bar', stacked=False, color=['Lightgrey','pink'], grid=False)
plt.xlabel("Ratings")
plt.ylabel("Review_Count")
plt.title("Ratings vs Helpfulness")
```

Out[14]:

```
<matplotlib.text.Text at 0x119d10b8>
```



In [15]:

```
t1 = pd.crosstab(df['overall'], df['ishelpful'], margins = True)
t1.columns = ["Not-helpful","helpful","Total"]
t1.index = [1,2,3,4,5,"Total reviews"]
t0 = t1/t1.ix["Total reviews","Total"]*100
t0['Total'] = t0['Total'].map('{:,.2f}%'.format)
t0
```

Out[15]:

	Not-helpful	helpful	Total
1	4.180860	4.310799	8.49%
2	3.622476	4.395083	8.02%
3	5.165935	7.938543	13.10%
4	6.015803	14.558385	20.57%
5	13.190518	36.621598	49.81%
Total reviews	32.175593	67.824407	100.00%

I looked at the percentage of those reviews that users found helpful or not helpful for each Star rating. And we notice that as the ratings increase, the reviews become more helpful. For 5-star reviews, 36% reviews were found helpful and 13% not helpful.

IV.III- Avg word_cnt per review

It will be interesting to visualize how word counts vary for reviews. Further, I would also like to explore the correlation between word count and other characteristics of a given review like helpfulness.

Word cnt wrt Rating(overall)

In [16]:

```
pysql = lambda q: pdsql.sqldf(q, globals())
str1= """SELECT overall as Ratings,avg(wordcount), min(wordcount), max(word
count)
from df
group by overall
"""
p1 = pysql(str1)
p1.head(10)
p1
# p1.plot(kind = 'box')
```

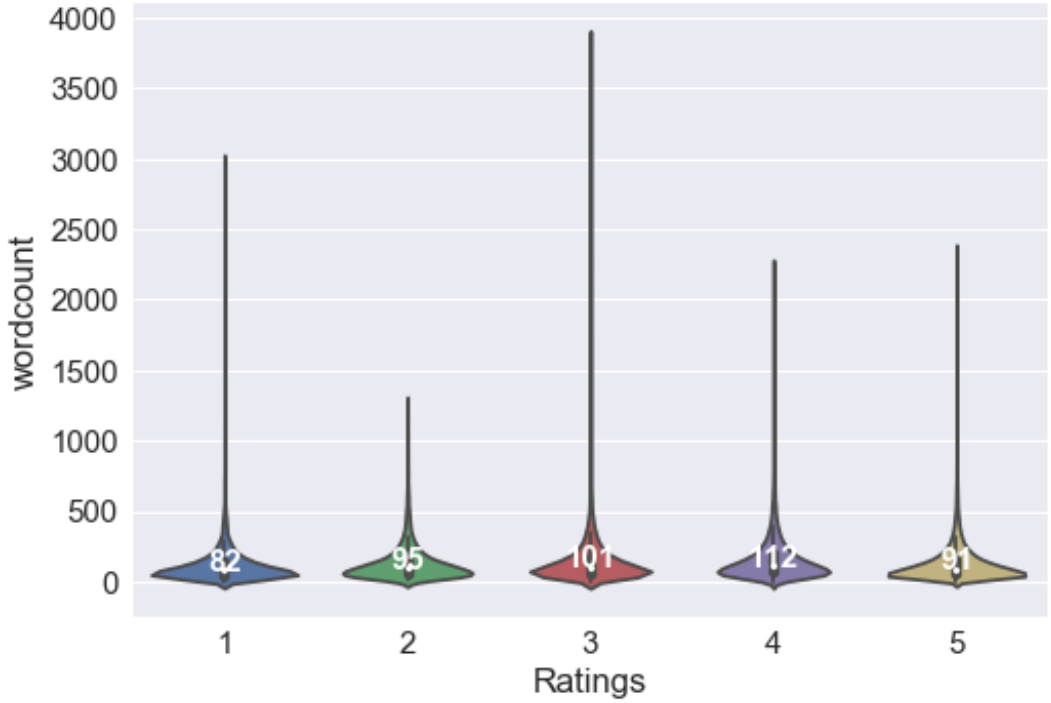
Out[16]:

	Ratings	avg(wordcount)	min(wordcount)	max(wordcount)
0	1	117.148677	2	2978
1	2	132.210031	12	1262
2	3	144.172451	4	3855
3	4	159.316293	2	2232
4	5	130.617844	1	2352

In [17]:

```
sns.set_context('notebook', font_scale=1.5, rc={'line.linewidth': 3})

pysql = lambda q: pdsql.sqldf(q, globals())
str1= """SELECT overall as Ratings, wordcount
```

```
from df
"""
p2 = pysql(str1)
color = dict(boxes='DarkGreen', whiskers='DarkOrange', medians='DarkBlue',
caps='Gray')
ax = sns.violinplot( p2.Ratings,p2.wordcount, palette = 'deep', color =
color)
medians = p2.groupby(['Ratings'])['wordcount'].median().values
median_labels = [str(np.round(s, 2)) for s in medians]
pos = range(len(medians))
for tick,label in zip(pos,ax.get_xticklabels()):
    ax.text(pos[tick], medians[tick] + 0.25, median_labels[tick],
            horizontalalignment='center', size='small', color='w', weight='b
old')
```



5-star reviews have the second lowest median word count (91 words), while 3-star reviews have relatively higher median word count (101 words).

IV.IV- Avg word_cnt per review on helpfulness index

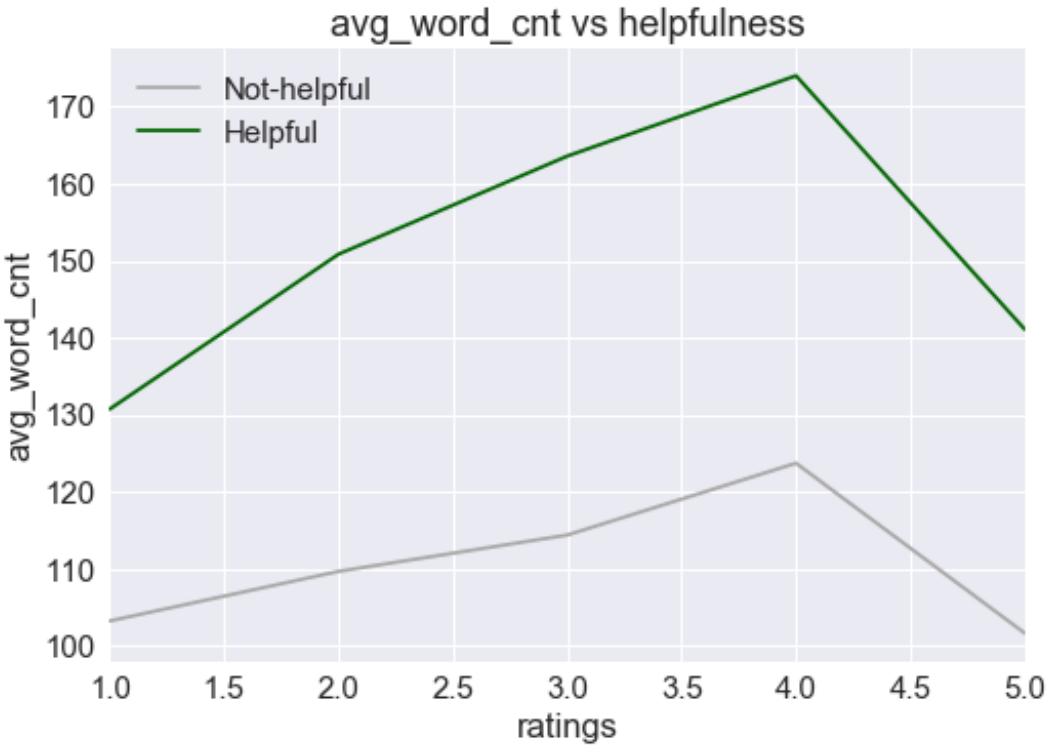
In [18]:

```
pysql = lambda q: pdsql.sqldf(q, globals())

str1= """SELECT overall, Avg(wordcount) as avg_wordcount
from df
where ishelpful = 0
group by overall, ishelpful
"""
df1 = pysql(str1)
# df1.head(10)
# ax = df1.plot(x='overall', y='avg_wordcount',  color='g')

str2= """SELECT overall, Avg(wordcount) as avg_wordcount
from df
```

```
where ishelpful = 1
group by overall, ishelpful
"""
df1 = pysql(str1)
df2 = pysql(str2)
ax = df1.plot(x='overall', y='avg_wordcount', color='Darkgrey', label =
'Not-helpful')
df2.plot(x='overall', y='avg_wordcount', color='Darkgreen', label =
'Helpful', ax = ax)
plt.ylabel('avg_word_cnt')
plt.xlabel('ratings')
plt.title("avg_word_cnt vs helpfulness")
```

Out[18]:

```
<matplotlib.text.Text at 0x1be458d0>
```



The word counts for helpful reviews and not helpful reviews have a similar distribution. However, not helpful reviews have a larger concentration of reviews with low word count and helpful reviews have more longer reviews.

IV.V-Regular vs Non Regular Reviewers

We are interested to draw a comparision between the reviews of regular vs not regular reviewers. So who are regular reviewers? Our obvious choice would be those customers who have clearly reviewed more than once. the more the better! After analysing the review counts, we found out that there is a good distinction of review counts between <3 reviews and more than 3 reviews /customer in the dataset. So, we assigned reviewers as frequent reviewers who have more than 3 reviews and vice-versa. The goal here is to identify if there is any behavioral distinction between frequent and not frequent reviewer groups.

In [19]:

```
pysql = lambda q: pdsql.sqldf(q, globals())
```

```
str1= """SELECT reviewerID, count(overall) as frequency
from df
group by reviewerID
"""
df4 = pysql(str1)
df4['% of total review']= df4['frequency']/df4['frequency'].sum()*100
df4['% of total review'] = df4['% of total review'].map('{:,.2f}%'.format)
df4["frequency"].value_counts()
```

Out[19]:

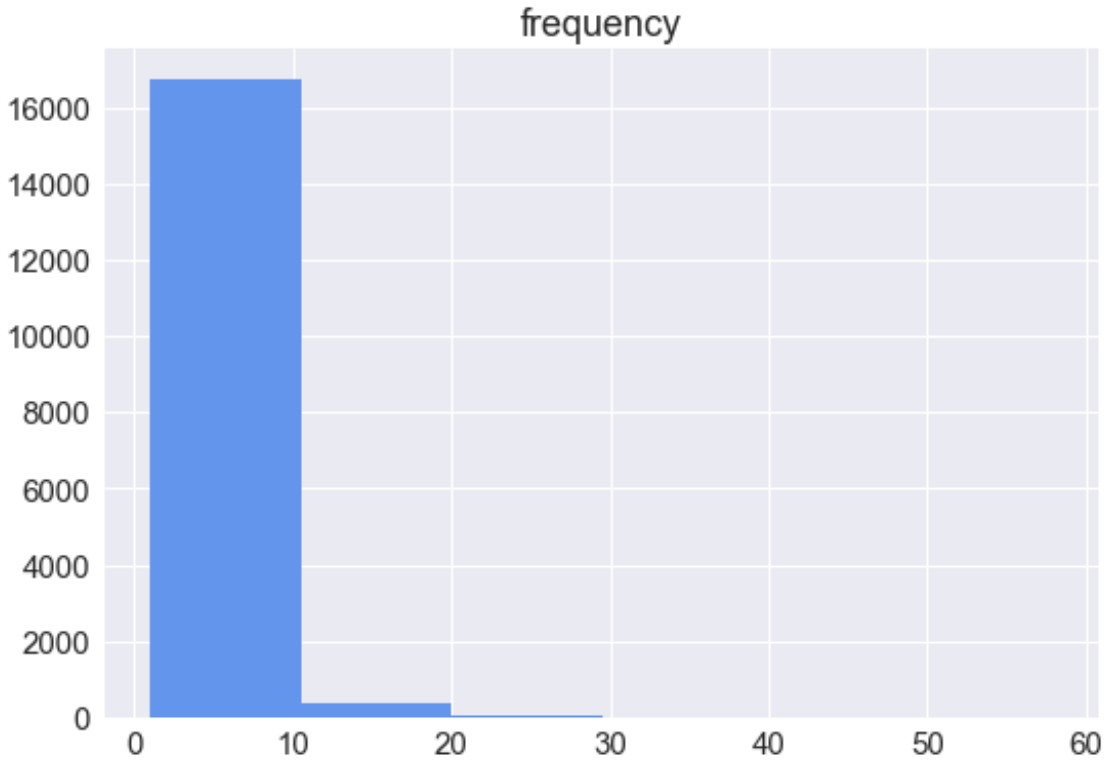
```
2     4143
1     4047
3     3267
4     2154
5     1395
6      739
7      431
8      251
9      178
10     135
11      90
12      71
13      57
14      31
17      29
15      28
16      25
18      14
20      12
19      11
21       8
22       8
32       6
26       6
23       5
24       4
27       3
36       2
25       2
38       2
31       1
49       1
48       1
30       1
45       1
33       1
58       1
44       1
34       1
35       1
57       1
37       1
53       1
47       1
Name: frequency, dtype: int64
```

In [20]:

```
df4.hist(layout=(1,2),bins = 6, figsize = [20,6], color = 'cornflowerblue')
```

Out[20]:

```
array([[<matplotlib.axes._subplots.AxesSubplot object at
0x0000000011C2A908>,
        <matplotlib.axes._subplots.AxesSubplot object at
0x0000000011AE26A0>]], dtype=object)
```



Here Frequency is the number of reviews completed by a given customer on the website. From the above histogram we see we have quite a good concentration of reviewers in the dataset reviewing in the range of 1-10 times. After analysing further we found below.

- We have 8977 Regular Customers with Review frequency > 2
- We have 8190 customers which are not so regular having frequency <= 2
- Majority of the reviews(7410) are done by customers who have reviewed atleast 2 or 3 times.

IV.VI Do "more reviews" mean "more helpful" ?

In [21]:

```
str1="""Select reviewerID, count(overall) as Frequency, overall as Stars, i
shelpful as Helpful
from df
group by reviewerID, overall
"""
df5 = pysql(str1)
# df5['% of total review']= df5['frequency']/df4['frequency'].sum()*100
# df5['% of total review'] = df5['% of total
review'].map('{:,.2f}%'.format)
# df5['reg_nonreg']=df5['frequency'].apply(lambda x: "Non-Regular" if x < 3
else "Regular")

df5['TotalFrequency']= df5.groupby('reviewerID').Frequency.transform(np.sum
)
s = df5[['Helpful','TotalFrequency']]
```

```
s
t1 = pd.crosstab(s['TotalFrequency'], s['Helpful'], margins = True)
t1.columns = ['Not-Helpful', 'Helpful', 'Total Reviewers']
t1
# t1.plot(color = ['orange','green','grey'])
# plt.xlabel("No of times user reviewed")
# plt.ylabel("Total Reviews")
# plt.title("Review frequency vs Helpfulness")
```

Out[21]:

	Not-Helpful	Helpful	Total Reviewers
TotalFrequency			
1	1557	2490	4047
2	2579	4082	6661
3	2415	4303	6718
4	1752	3387	5139
5	1206	2583	3789
6	704	1472	2176
7	450	884	1334
8	258	562	820
9	199	399	598
10	171	312	483
11	95	230	325
12	79	174	253
13	71	132	203
14	41	78	119
15	40	69	109
16	38	61	99
17	36	77	113
18	24	31	55
19	13	29	42
20	19	28	47
21	9	24	33
22	7	23	30
23	5	17	22
24	5	12	17
25	1	7	8
26	7	16	23
27	2	11	13

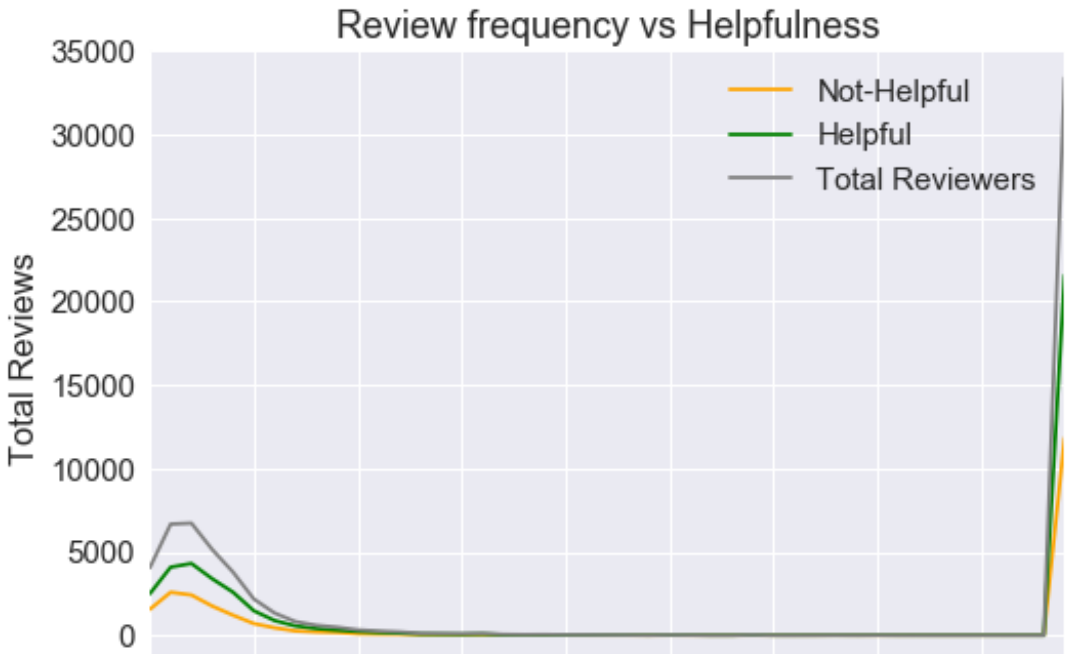
TotalFrequency	Not-Helpful	Helpful	Total Reviewers
30	0	3	3
31	2	2	4
32	7	19	26
33	2	3	5
34	1	3	4
35	0	5	5
36	4	6	10
37	0	5	5
38	0	9	9
44	0	4	4
45	2	3	5
47	1	3	4
48	1	3	4
49	2	3	5
53	2	2	4
57	3	2	5
58	5	0	5
All	11815	21568	33383

In [22]:

```
t1.plot(color = ['orange','green','grey'])
plt.xlabel("No of times user reviewed")
plt.ylabel("Total Reviews")
plt.title("Review frequency vs Helpfulness")
```

Out[22]:

```
<matplotlib.text.Text at 0x1a0dc3c8>
```



There is no striking pattern detected from the above plot generated for Total Frequecy of the reviews vs Helpfulness. However, we clearly see more helpful reviews for each frequency. Also, we notice more helpful reviews than not -useful ones for the users who have reviewed for about 2-8 times as opposed to higher frequency holders. Also, as the review frequency increases that is the number of times user gives review increases, so does the helpful index in general. So we cab say more reviews are better!

V. Inferences

- In general positive reviews are common in this dataset.
- We have more 50 % of the total reviews assigned as 5 -star.
- Best reviews (5-star) are relatively shorter.
- Longer reviews are more helpful.
- Frequent reviewers write longer and helpful reviews.

Thank You!