Conversion Rate

April 30, 2019

In [1]:
```
%matplotlib inline
```

In [2]:
```
import matplotlib.pyplot as plt
import pandas as pd
```

In [3]:
```
df = pd.read_csv('Data/bank-additional-full.csv', sep=';')
```

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

Out[4]:
```
(41188, 21)
```

In [5]:
```
df.head()
```

Out[5]:

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	...	campaign	pdays	previous	poutcome	emp.var.rate	cons.price.idx	cons.conf.idx	euribor3m	nr.employed	y
0	56	housemaid	married	basic.4y	no	no	no	telephone	may	mon	...	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191.0	no
1	57	services	married	high.school	unknown	no	no	telephone	may	mon	...	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191.0	no
2	37	services	married	high.school	no	yes	no	telephone	may	mon	...	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191.0	no
3	40	admin.	married	basic.6y	no	no	no	telephone	may	mon	...	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191.0	no
4	56	services	married	high.school	no	no	yes	telephone	may	mon	...	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191.0	no

[5 rows x 21 columns]

In [6]:
```
# Encode the y variable as 1 for 'yes' and as 0 for 'no'

df['conversion'] = df['y'].apply(lambda x:1 if x == 'yes' else 0)
df.head()
```

Out[6]:

	age	job	marital	education	default	housing	loan	contact	\
0	56	housemaid	married	basic.4y	no	no	no	telephone	
1	57	services	married	high.school	unknown	no	no	telephone	
2	37	services	married	high.school	no	yes	no	telephone	
3	40	admin.	married	basic.6y	no	no	no	telephone	
4	56	services	married	high.school	no	no	yes	telephone	

	month	day_of_week	...	pdays	previous	poutcome	emp.var.rate	\
0	may	mon	...	999	0	nonexistent	1.1	
1	may	mon	...	999	0	nonexistent	1.1	
2	may	mon	...	999	0	nonexistent	1.1	
3	may	mon	...	999	0	nonexistent	1.1	
4	may	mon	...	999	0	nonexistent	1.1	

	cons.price.idx	cons.conf.idx	euribor3m	nr.employed	y	conversion
0	93.994	-36.4	4.857	5191.0	no	0
1	93.994	-36.4	4.857	5191.0	no	0
2	93.994	-36.4	4.857	5191.0	no	0
3	93.994	-36.4	4.857	5191.0	no	0
4	93.994	-36.4	4.857	5191.0	no	0

[5 rows x 22 columns]

0.1 1. Aggregate Conversion Rate

```
In [7]: # Total number of conversions
        df.conversion.sum()
```

```
Out[7]: 4640
```

```
In [8]: # total number of clients in the data (= number of rows in the data)
        df.shape[0]
```

```
Out[8]: 41188
```

```
In [9]: print('total conversions: %i out of %i' % (df.conversion.sum(), df.shape[0]))
```

```
total conversions: 4640 out of 41188
```

```
In [10]: print('conversion rate: %0.2f%%' % (df.conversion.sum() / df.shape[0] * 100.0))
```

```
conversion rate: 11.27%
```

0.2 2. Conversion Rate by Age

```
In [11]: pd.DataFrame(
             df.groupby(
```

```
            by='age'
        )['conversion'].sum()
    ).head()
```

Out[11]:

age	conversion
17	2
18	12
19	20
20	23
21	29

```
In [12]: pd.DataFrame(
            df.groupby(
                by='age'
            )['conversion'].count()
        ).head()
```

Out[12]:

age	conversion
17	5
18	28
19	42
20	65
21	102

```
In [13]: conversions_by_age = df.groupby(
            by='age'
        )['conversion'].sum() / df.groupby(
            by='age'
        )['conversion'].count() * 100.0

In [14]: pd.DataFrame(conversions_by_age).head(10)
```

Out[14]:

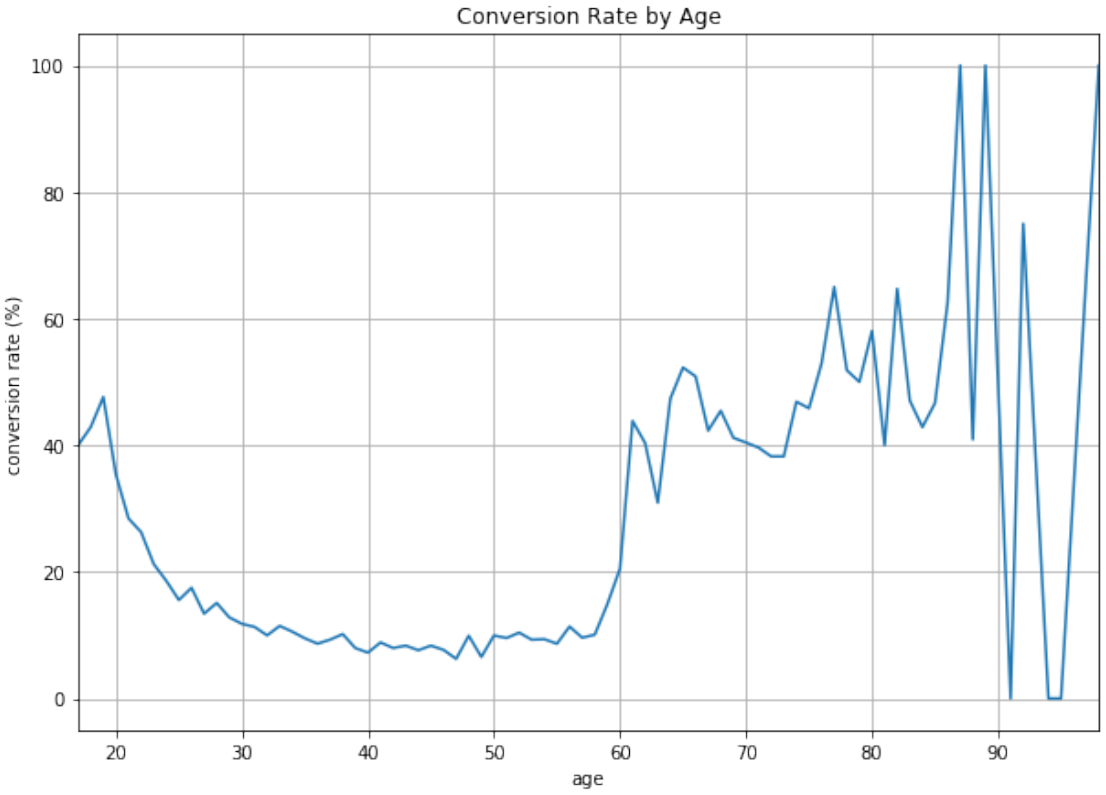
age	conversion
17	40.000000
18	42.857143
19	47.619048
20	35.384615
21	28.431373
22	26.277372
23	21.238938
24	18.574514
25	15.551839
26	17.478510

```
In [15]: # plot conversion rate by age
```

```
ax = conversions_by_age.plot(
    grid=True,
    figsize=(10, 7),
    title='Conversion Rate by Age'
)

ax.set_xlabel('age')
ax.set_ylabel('conversion rate (%)')

plt.show()
```



There seems to be a lot of noise in old age groups. In order to reduce this unwanted noise, we can group bank clients into 6 different groups, based on their age.

0.2.1 Create groups (classes) based on their age

```
In [16]: df['age_group'] = df['age'].apply(
             lambda x: '[18, 30)' if x < 30 else '[30, 40)' if x < 40 \
                 else '[40, 50)' if x < 50 else '[50, 60)' if x < 60 \
                 else '[60, 70)' if x < 70 else '70+'
         )

In [17]: df.head(10)
```

Out[17]:

	age	job	marital	education	default	housing	loan
0	56	housemaid	married	basic.4y	no	no	no
1	57	services	married	high.school	unknown	no	no
2	37	services	married	high.school	no	yes	no
3	40	admin.	married	basic.6y	no	no	no
4	56	services	married	high.school	no	no	yes
5	45	services	married	basic.9y	unknown	no	no
6	59	admin.	married	professional.course	no	no	no
7	41	blue-collar	married	unknown	unknown	no	no
8	24	technician	single	professional.course	no	yes	no
9	25	services	single	high.school	no	yes	no

	contact	month	day_of_week	...	previous	poutcome	emp.var.rate
0	telephone	may	mon	...	0	nonexistent	1.1
1	telephone	may	mon	...	0	nonexistent	1.1
2	telephone	may	mon	...	0	nonexistent	1.1
3	telephone	may	mon	...	0	nonexistent	1.1
4	telephone	may	mon	...	0	nonexistent	1.1
5	telephone	may	mon	...	0	nonexistent	1.1
6	telephone	may	mon	...	0	nonexistent	1.1
7	telephone	may	mon	...	0	nonexistent	1.1
8	telephone	may	mon	...	0	nonexistent	1.1
9	telephone	may	mon	...	0	nonexistent	1.1

	cons.price.idx	cons.conf.idx	euribor3m	nr.employed	y	conversion
0	93.994	-36.4	4.857	5191.0	no	0
1	93.994	-36.4	4.857	5191.0	no	0
2	93.994	-36.4	4.857	5191.0	no	0
3	93.994	-36.4	4.857	5191.0	no	0
4	93.994	-36.4	4.857	5191.0	no	0
5	93.994	-36.4	4.857	5191.0	no	0
6	93.994	-36.4	4.857	5191.0	no	0
7	93.994	-36.4	4.857	5191.0	no	0
8	93.994	-36.4	4.857	5191.0	no	0
9	93.994	-36.4	4.857	5191.0	no	0

	age_group
0	[50, 60)
1	[50, 60)
2	[30, 40)
3	[40, 50)
4	[50, 60)
5	[40, 50)
6	[50, 60)
7	[40, 50)
8	[18, 30)
9	[18, 30)

```
[10 rows x 23 columns]
```

In [18]:
```
pd.DataFrame(
    df.groupby(
        by='age_group'
    )['conversion'].sum()
).head(5)
```

Out[18]:
```
            conversion
age_group
70+                221
[18, 30)           922
[30, 40)          1715
[40, 50)           834
[50, 60)           697
```

In [19]:
```
pd.DataFrame(
    df.groupby(
        by='age_group'
    )['conversion'].count()
).head(5)
```

Out[19]:
```
            conversion
age_group
70+                469
[18, 30)          5669
[30, 40)         16938
[40, 50)         10526
[50, 60)          6862
```

In [20]:
```
# Calculate conversion rate
conversions_by_age_group = df.groupby(
    by='age_group'
)['conversion'].sum() / df.groupby(
    by='age_group'
)['conversion'].count() * 100.0
```

In [21]:
```
pd.DataFrame(conversions_by_age_group)
```

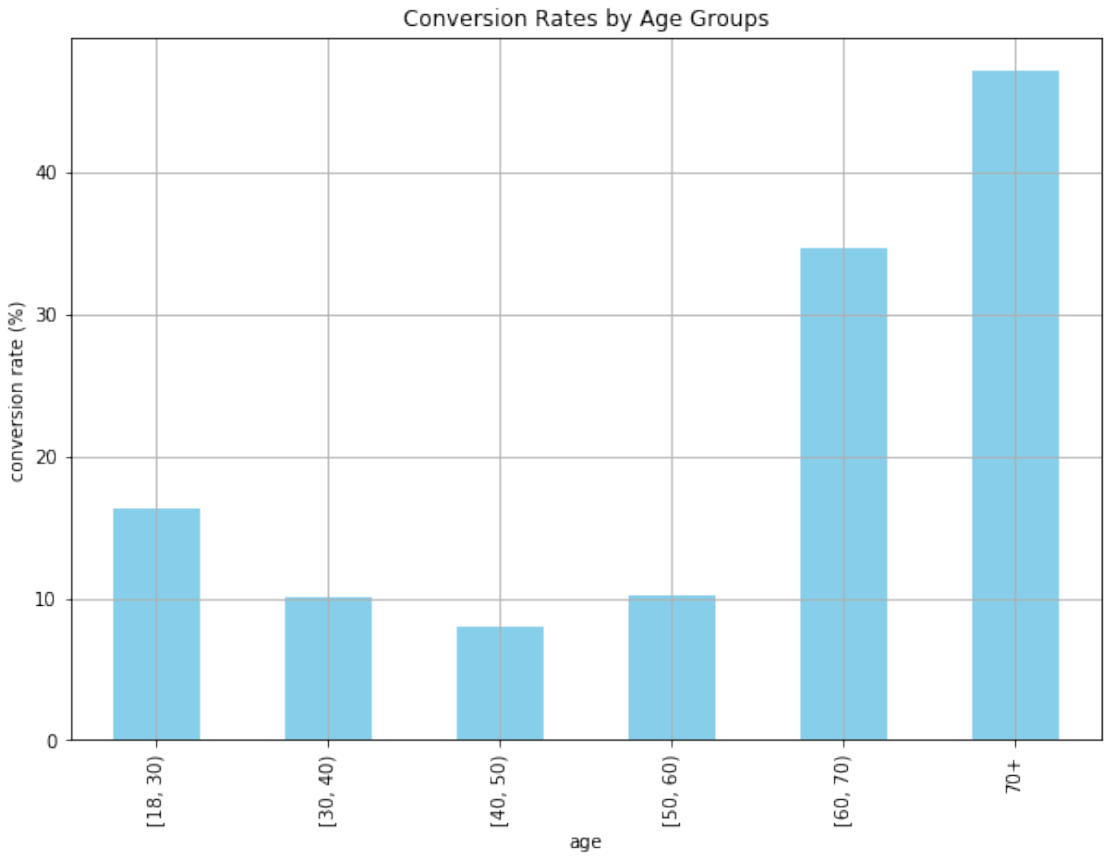
Out[21]:
```
            conversion
age_group
70+          47.121535
[18, 30)     16.263891
[30, 40)     10.125162
[40, 50)      7.923238
[50, 60)     10.157389
[60, 70)     34.668508
```

In [22]:
```
ax = conversions_by_age_group.loc[
    ['[18, 30)', '[30, 40)', '[40, 50)', '[50, 60)', '[60, 70)', '70+']
```

```
].plot(
    kind='bar',
    color='skyblue',
    grid=True,
    figsize=(10, 7),
    title='Conversion Rates by Age Groups'
)

ax.set_xlabel('age')
ax.set_ylabel('conversion rate (%)')

plt.show()
```



As you can see the variations by each age group are much smaller than before when we calculate the conversion rate by age across all clients.

0.3 3. Conversion Rates by Number of Contacts

```
In [23]: pd.DataFrame(
             df.groupby(
                 by='campaign'
```

```
        )['conversion'].sum()
    ).head(5)
```

Out[23]:

campaign	conversion
1	2300
2	1211
3	574
4	249
5	120

```
In [24]: pd.DataFrame(
             df.groupby(
                 by='campaign'
             )['conversion'].count()
         ).head(5)
```

Out[24]:

campaign	conversion
1	17642
2	10570
3	5341
4	2651
5	1599

```
In [25]: # Calculate conversion rate by contacts
         conversions_by_contacts = df.groupby(
             by='campaign'
         )['conversion'].sum() / df.groupby(
             by='campaign'
         )['conversion'].count() * 100.0

In [26]: pd.DataFrame(conversions_by_contacts).head(5)
```

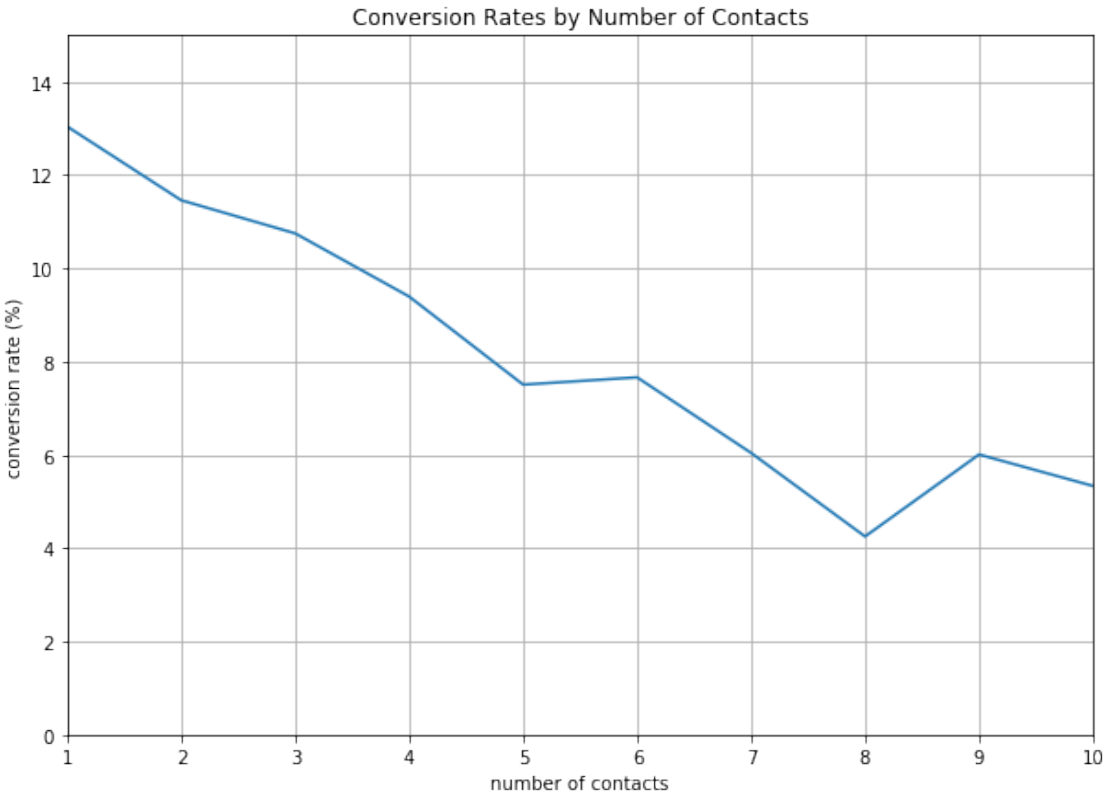
Out[26]:

campaign	conversion
1	13.037071
2	11.456954
3	10.747051
4	9.392682
5	7.504690

```
In [27]: ax = conversions_by_contacts[:10].plot(
             grid=True,
             figsize=(10, 7),
             xticks=conversions_by_contacts.index[:10],
             title='Conversion Rates by Number of Contacts'
         )
```

```
ax.set_ylim([0, 15])
ax.set_xlabel('number of contacts')
ax.set_ylabel('conversion rate (%)')

plt.show()
```



0.4 4. Conversions vs. Non-Conversions

Now we look at demographic differences between the converted clients and non-converted clients. This type of analysis can hep differentiates converted groups from non-converted groups in our marketing campaigns and hep understand the target clients better and what types of customers respond better.

0.4.1 4.1 Marital Status

```
In [28]: conversions_by_marital_status_df = pd.pivot_table(df, values='y', index='marital',
                                                           columns='conversion', aggfunc=len)
         conversions_by_marital_status_df
```

```
Out[28]: conversion      0     1
         marital
         divorced     4136   476
```

```
married      22396  2532
single        9948  1620
unknown         68    12
```

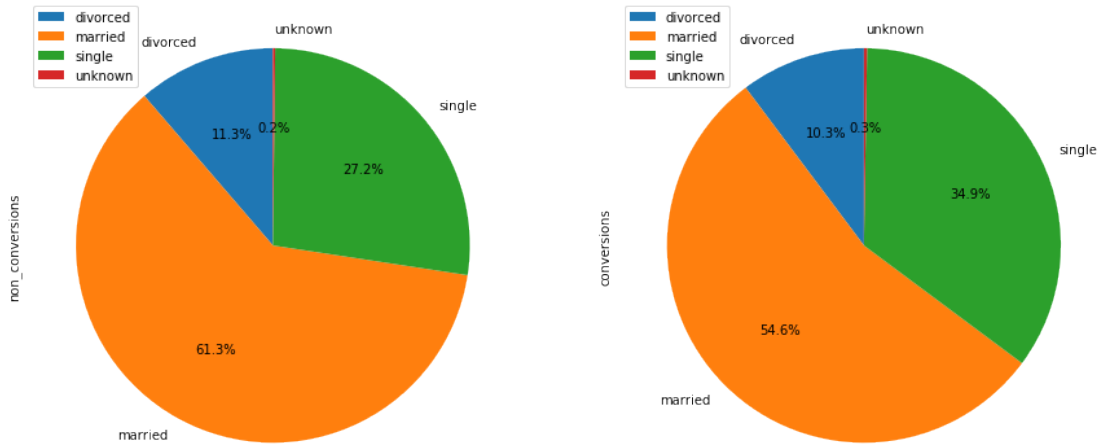
aggfunc() allows us to supply the type of aggregation we want to perform. We use len function to simply count the number of clients for each group.

```
In [29]: conversions_by_marital_status_df.columns = ['non_conversions', 'conversions']
         conversions_by_marital_status_df
```

```
Out[29]:           non_conversions  conversions
         marital
         divorced              4136          476
         married              22396         2532
         single                9948         1620
         unknown                 68           12
```

```
In [30]: # Use pie chart to represent this data
         conversions_by_marital_status_df.plot(
             kind='pie',
             figsize=(15, 7),
             startangle=90,
             subplots=True,
             autopct=lambda x: '%0.1f%%' % x
         )

         plt.show()
```



Using pie charts, we can easily visualize the similarities and differences between two groups. In this example, we can easily see that married group takes up the largest proportions in both conversions and non-conversions groups, while the single group comes as the second.

0.4.2 4.2 Education

```
In [31]: conversions_by_education_df = pd.pivot_table(df, values='y', index='education', columns
         conversions_by_education_df
```

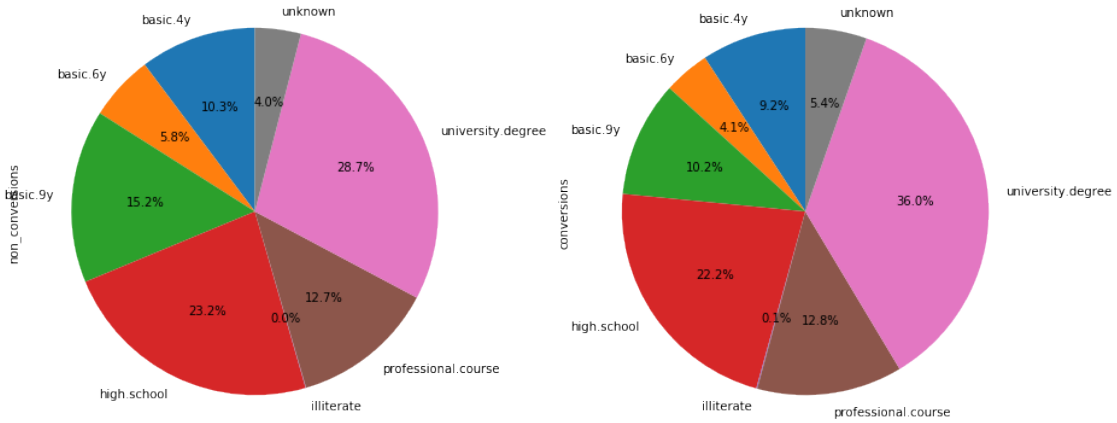
```
Out[31]: conversion               0     1
         education
         basic.4y              3748   428
         basic.6y              2104   188
         basic.9y              5572   473
         high.school           8484  1031
         illiterate              14     4
         professional.course   4648   595
         university.degree    10498  1670
         unknown               1480   251
```

```
In [32]: conversions_by_education_df.columns = ['non_conversions', 'conversions']
         conversions_by_education_df
```

```
Out[32]:                      non_conversions  conversions
         education
         basic.4y                        3748          428
         basic.6y                        2104          188
         basic.9y                        5572          473
         high.school                     8484         1031
         illiterate                        14            4
         professional.course             4648          595
         university.degree              10498         1670
         unknown                         1480          251
```

```
In [33]: conversions_by_education_df.plot(
             kind='pie',
             figsize=(15, 7),
             startangle=90,
             subplots=True,
             autopct=lambda x: '%0.1f%%' % x,
             legend=False
         )

         plt.show()
```



0.4.3 4.3 Last Contact Duration

```
In [34]: df.groupby('conversion')['duration'].describe()
```

```
Out[34]:              count        mean         std   min    25%    50%     75%  \
         conversion
         0           36548.0  220.844807  207.096293   0.0   95.0  163.5  279.00
         1            4640.0  553.191164  401.171871  37.0  253.0  449.0  741.25

                       max
         conversion
         0          4918.0
         1          4199.0
```

```
In [35]: duration_df = pd.concat([
             df.loc[df['conversion'] == 1, 'duration'].reset_index(drop=True),
             df.loc[df['conversion'] == 0, 'duration'].reset_index(drop=True)
         ], axis=1)

         duration_df.columns = ['conversions', 'non_conversions']

         duration_df = duration_df / (60*60)
```

```
In [36]: duration_df.head(10)
```

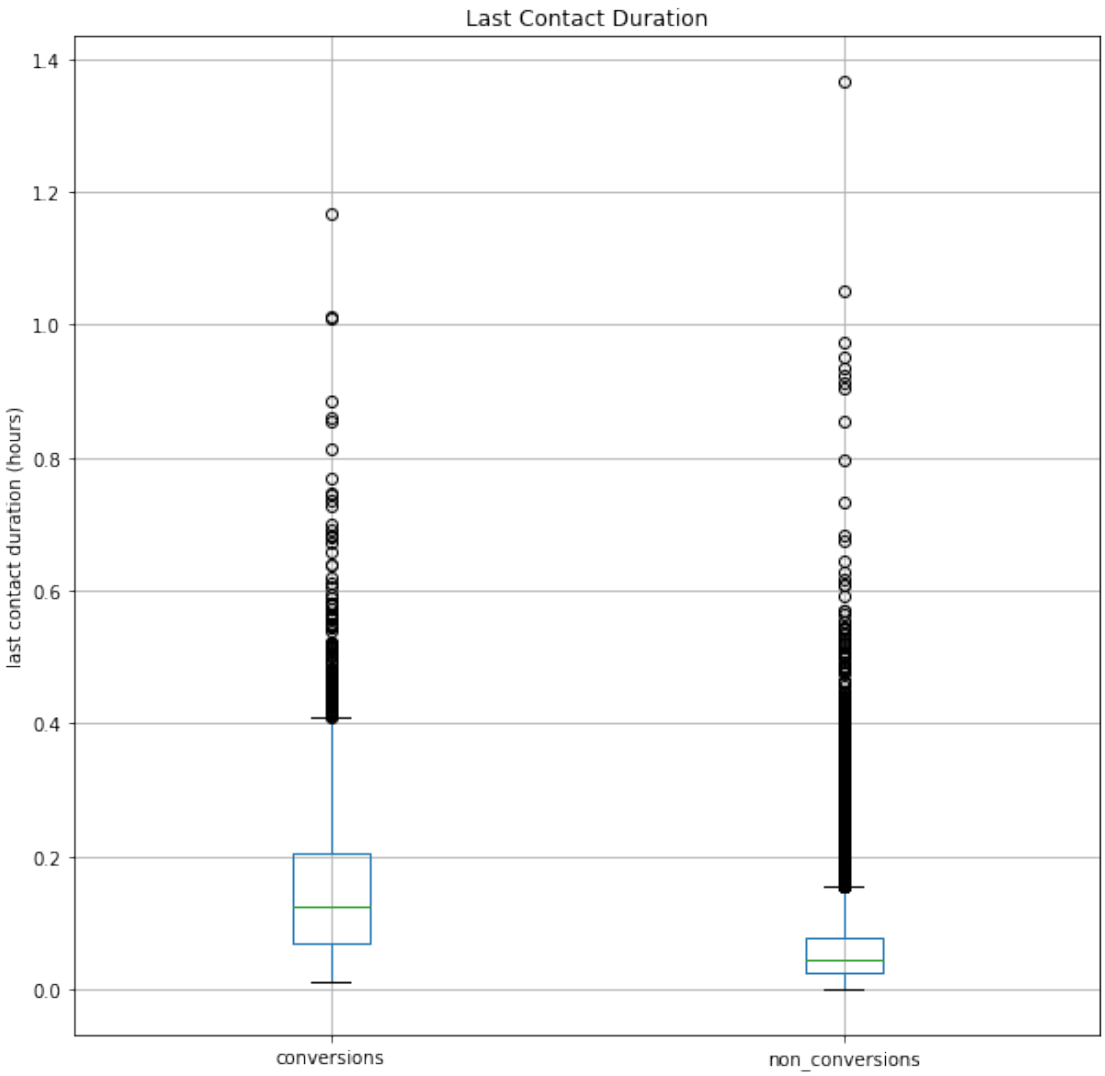
```
Out[36]:    conversions  non_conversions
         0     0.437500         0.072500
         1     0.289444         0.041389
         2     0.407500         0.062778
         3     0.160833         0.041944
         4     0.128056         0.085278
```

```
5      0.186944          0.055000
6      0.259722          0.038611
7      0.333611          0.060278
8      0.286111          0.105556
9      0.450833          0.013889
```

In [37]:
```
ax = duration_df.plot(
    kind='box',
    grid=True,
    figsize=(10, 10),
)

ax.set_ylabel('last contact duration (hours)')
ax.set_title('Last Contact Duration')

plt.show()
```



0.5 5. Conversion by more than one criterion

0.5.1 Conversions by Age Groups and Marital Status

In [38]:
```
# Grouping the data by two features (columns)
# and summing the number of conversions

age_marital_df = df.groupby(['age_group', 'marital'])['conversion'].sum().unstack('ma
```

In [39]:
```
# Divide the figures of the previous dataframe by the total number of clients in each

age_marital_df = age_marital_df.divide(
    df.groupby(
        by='age_group'
    )['conversion'].count(),
    axis=0
)
```

In [40]:
```
age_marital_df
```

Out[40]:

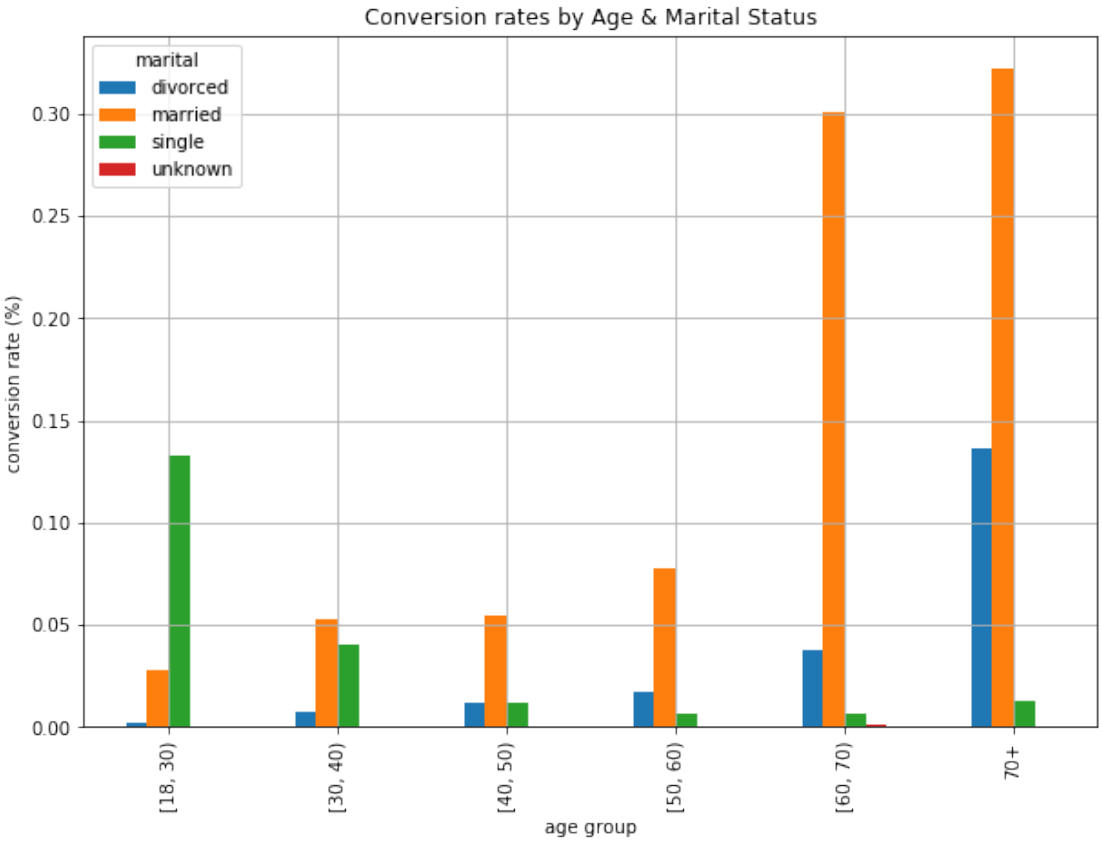
marital	divorced	married	single	unknown
age_group				
70+	0.136461	0.321962	0.012793	0.000000
[18, 30)	0.002117	0.027871	0.132475	0.000176
[30, 40)	0.007557	0.052958	0.040383	0.000354
[40, 50)	0.011970	0.054627	0.012350	0.000285
[50, 60)	0.017342	0.077674	0.006412	0.000146
[60, 70)	0.037293	0.301105	0.006906	0.001381

In [41]:
```
# Another way to visualize the previous output with a bar plot

ax = age_marital_df.loc[
    ['[18, 30)', '[30, 40)', '[40, 50)', '[50, 60)', '[60, 70)', '70+']
].plot(
    kind='bar',
    grid=True,
    figsize=(10,7)
)

ax.set_title('Conversion rates by Age & Marital Status')
ax.set_xlabel('age group')
ax.set_ylabel('conversion rate (%)')

plt.show()
```

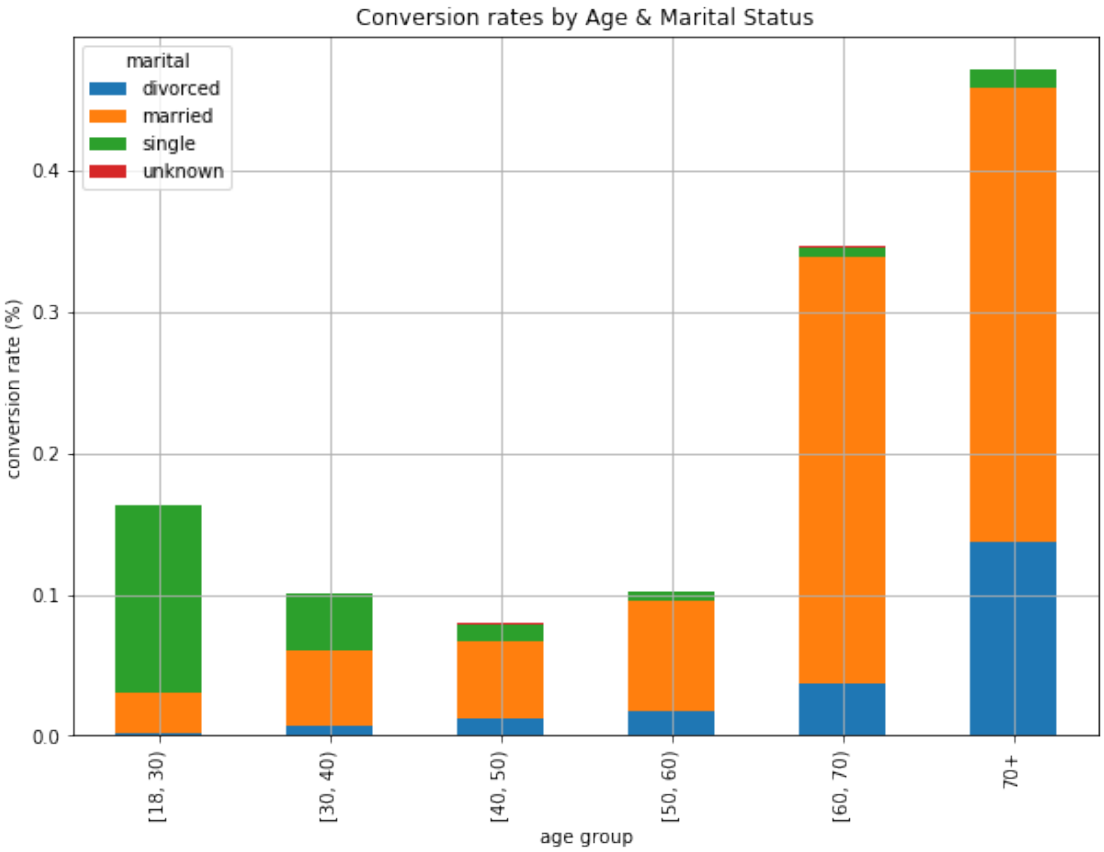


In [42]:
```
#  if you want to stack those previous 4 bars for each of the marital statuses, use a

ax = age_marital_df.loc[
    ['[18, 30)', '[30, 40)', '[40, 50)', '[50, 60)', '[60, 70)', '70+']
].plot(
    kind='bar',
    stacked=True,
    grid=True,
    figsize=(10,7)
)

ax.set_title('Conversion rates by Age & Marital Status')
ax.set_xlabel('age group')
ax.set_ylabel('conversion rate (%)')

plt.show()
```


Conversion rates by Age & Marital Status
marital
divorced
married
single
unknown
conversion rate (%)
0.0
0.1
0.2
0.3
0.4
[18, 30)
[30, 40)
[40, 50)
[50, 60)
[60, 70)
70+
age group