In [173]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

%matplotlib inline
```

In [174]:
```
time_df = pd.read_csv("time_data.csv")
```

In [206]:
```
len(np.unique(time_df['participant_number']))
```

Out[206]: 24

In [175]:
```
time_df.head()
```

Out[175]:

	participant_number	condition	menu_order	selection_time	correctly_predicted	error_
0	0	control	0	4073	True	False
1	0	control	0	2346	True	False
2	0	control	0	1072	True	False
3	0	control	0	1272	False	False
4	0	control	0	1904	True	False

In [176]:
```
np.sum(time_df['menu_order'])
```

Out[176]: 2520

In [177]:
```
time_df.dtypes
```

Out[177]:
```
participant_number      int64
condition              object
menu_order              int64
selection_time          int64
correctly_predicted      bool
error_in_trial           bool
dtype: object
```

In [178]:
```
time_df.describe()
```

Out[178]:

	participant_number	menu_order	selection_time
count	4320.000000	4320.000000	4320.000000
mean	12.000000	0.583333	1818.491204
std	7.360653	0.493064	1120.501622
min	0.000000	0.000000	526.000000
25%	5.750000	0.000000	1173.000000
50%	12.000000	1.000000	1528.000000
75%	18.250000	1.000000	2152.000000
max	24.000000	1.000000	19888.000000

(1) Median time per condition

In [212]:
```
np.median(time_df['selection_time'][time_df['condition'] == "control"])
```

Out[212]: 1594.0

In [213]:
```
np.median(time_df['selection_time'][time_df['condition'] != "control"])
```

Out[213]: 1478.0

(2) Median time per condition per Predicted vs Non-predicted trial

In [181]:
```
# average_speed = np.average(time_df['selection_time'], weights = (time_
df['condition'] == "control"))
# average_speed/1000 # to seconds
```

(2a) Within control condition: compare predicted vs non-predicted

In [214]:
```
np.median(time_df['selection_time'][time_df['condition'] == "control"][t
ime_df['correctly_predicted'] == True])
```

Out[214]: 1587.0

In [215]:
```
np.median(time_df['selection_time'][time_df['condition'] == "control"][t
ime_df['correctly_predicted'] == False])
```

Out[215]: 1611.5

(2b) Within ephemeral condition: compare predicted vs non-predicted trial

In [216]:
```
np.median(time_df['selection_time'][time_df['condition'] != "control"][time_df['correctly_predicted'] == True])
```

Out[216]: 1406.5

In [217]:
```
np.median(time_df['selection_time'][time_df['condition'] != "control"][time_df['correctly_predicted'] == False])
```

Out[217]: 1706.5

ANOVA 2 x 2 (menu x presentation order)

In [235]:
```
# from statsmodels.stats.anova import anova_lm
# from statsmodels.formula.api import ols
```

In [236]:
```
# time_df.dtypes
```

In [237]:
```
# time_df['condition'] = time_df['condition'].astype('str')#astype('int64')
```

In [238]:
```
# formula = 'selection_time ~ C(condition) + C(menu_order) + C(condition):C(menu_order)'
# model = OLS(formula, time_df).fit()
# aov_table = anova_lm(model, typ=2)
```

Output from R

> fit <- aov(selection_time~as.factor(condition) * as.factor(menu_order), data = time_df) > summary(fit) Df Sum Sq Mean Sq F value as.factor(condition) 1 4.349e+07 43493697 35.169 as.factor(menu_order) 1 1.060e+06 1060266 0.857 as.factor(condition):as.factor(menu_order) 1 4.041e+07 40414911 32.679 Residuals 4316 5.338e+09 1236710 Pr(>F) as.factor(condition) 3.26e-09 *** as.factor(menu_order) 0.355 as.factor(condition):as.factor(menu_order) 1.16e-08 *** Residuals --- Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

There is a main effect on condition, but no effect on the menu order.

ANOVA on menu * correct_predicted trials

Output from R

> fit2 <- aov(selection_time~as.factor(condition) * as.factor(correctly_predicted), data = time_df) > summary(fit2) Df Sum Sq Mean Sq F value Pr(>F) as.factor(condition) 1 4.349e+07 43493697 35.143 3.30e-09 *** as.factor(correctly_predicted) 1 2.932e+07 29323689 23.693 1.17e-06 ***

as.factor(condition):as.factor(correctly_predicted) 1 8.160e+06 8160432 6.594 0.0103 * Residuals 4316 5.342e+09 1237634 --- Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

There are main effects on condition and predicted trial. We also found interaction effect on condition and predicted trial

(3) Error rate

In [186]:
```
np.sum(time_df['error_in_trial'][time_df['condition'] == "control"])
# np.sum(time_df['condition'] == 'control')
```

Out[186]: 8

In [187]:
```
np.average(time_df['error_in_trial'][time_df['condition'] == "control"])
```

Out[187]: 0.003703703703703704

In [188]:
```
np.sum(time_df['error_in_trial'][time_df['condition'] != "control"])
# np.sum(time_df['condition'] != 'control')
```

Out[188]: 3

In [189]:
```
np.average(time_df['error_in_trial'][time_df['condition'] != "control"])
```

Out[189]: 0.001388888888888889

Qualitative Questions

In [190]:
```
qual_df = pd.read_csv('qualitative_data.csv')
qual_df.head()
```

Out[190]:

	participant_number	age	gender	menu_order	overall_preference	overall_easy	contro
0	0	26-30	Male	0	Second two	Second two	4
1	1	21-25	Female	0	Second two	Second two	5
2	2	21-25	Male	1	First two	First two	5
3	3	26-30	Female	1	Second two	Second two	7
4	4	21-25	Other	1	First two	First two	6

In [191]:
```
qual_df.columns
```

Out[191]:
```
Index(['participant_number', 'age', 'gender', 'menu_order',
       'overall_preference', 'overall_easy', 'control_satisfaction',
       'control_difficulty', 'control_frustration', 'control_efficienc
y',
       'ephemeral_satisfaction', 'ephemeral_difficulty',
       'ephemeral_frustration', 'ephemeral_efficiency'],
      dtype='object')
```

Gender

In [192]:
```
np.sum(qual_df['gender'] == "Male")
```

Out[192]: 14

In [193]:
```
len(qual_df['gender'])
```

Out[193]: 24

In [194]:
```
np.sum(qual_df['menu_order'])
```

Out[194]: 14

Overall Preference

In [195]:
```
qual_df['overall_pref'] = qual_df['overall_preference']
```

In [196]:
```
qual_df.loc[(qual_df['menu_order'] == 0) & (qual_df['overall_pref'] ==
'First two'), 'overall_pref'] = "Control"
qual_df.loc[(qual_df['menu_order'] == 0) & (qual_df['overall_pref'] ==
'Second two'), 'overall_pref'] = "Ephermal"
qual_df.loc[(qual_df['menu_order'] == 1) & (qual_df['overall_pref'] ==
'First two'), 'overall_pref'] = "Ephermal"
qual_df.loc[(qual_df['menu_order'] == 1) & (qual_df['overall_pref'] ==
'Second two'), 'overall_pref'] = "Control"
```

In [197]:
```
qual_df[['menu_order', 'overall_preference', 'overall_pref']].head()
```

Out[197]:

	menu_order	overall_preference	overall_pref
0	0	Second two	Ephermal
1	0	Second two	Ephermal
2	1	First two	Ephermal
3	1	Second two	Control
4	1	First two	Ephermal

In [208]:
```
print(str(np.sum(qual_df['overall_pref'] == "Ephermal")) + ' people pref
er the Ephermal condition')
print(str(np.sum(qual_df['overall_pref'] == "Ephermal")/len(qual_df) * 1
00) + ' % of people prefer the Ephermal condition')
```

```
15 people prefer the Ephermal condition
62.5 % of people prefer the Ephermal condition
```

Overall Easiness

In [209]:
```
qual_df['overall_easiness'] = qual_df['overall_easy']
qual_df.loc[(qual_df['menu_order'] == 0) & (qual_df['overall_easy'] ==
'First two'), 'overall_easiness'] = "Control"
qual_df.loc[(qual_df['menu_order'] == 0) & (qual_df['overall_easy'] ==
'Second two'), 'overall_easiness'] = "Ephermal"
qual_df.loc[(qual_df['menu_order'] == 1) & (qual_df['overall_easy'] ==
'First two'), 'overall_easiness'] = "Ephermal"
qual_df.loc[(qual_df['menu_order'] == 1) & (qual_df['overall_easy'] ==
'Second two'), 'overall_easiness'] = "Control"
```

In [210]:
```
qual_df[['menu_order', 'overall_easy', 'overall_easiness']].head()
```

Out[210]:

	menu_order	overall_easy	overall_easiness
0	0	Second two	Ephermal
1	0	Second two	Ephermal
2	1	First two	Ephermal
3	1	Second two	Control
4	1	First two	Ephermal

In [211]:
```
print(str(np.sum(qual_df['overall_easiness'] == "Ephermal")) + ' people
 found the Ephermal condition easier to use')
print(str(np.sum(qual_df['overall_easiness'] == "Ephermal")/len(qual_df)
 * 100) + ' % of people found the Ephermal condition easier to use')
```

```
15 people found the Ephermal condition easier to use
62.5 % of people found the Ephermal condition easier to use
```

Note that: if people prefer the condition, they also say they found the condition easy to use.

Qualitative Questions Analysis + Plot

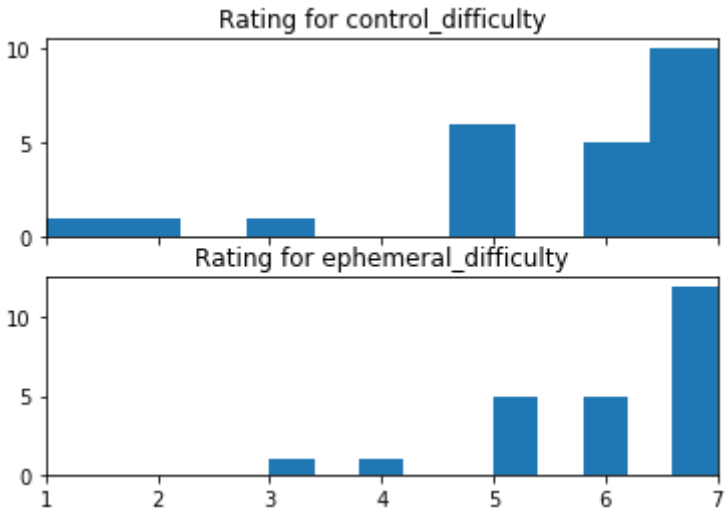
In [199]:
```
from scipy.stats import wilcoxon
```

In [200]:
```
def plot_qual_function(x, y):
    fig, ax = plt.subplots(2, sharex=True)
    ax[0].hist(qual_df[x])
    ax[0].set_xlim([1,7])
    ax[0].set_title('Rating for ' + x)
    print('The average for ' + x +'is: ' + str(np.average(qual_df[x])))

    ax[1].hist(qual_df[y])
    ax[1].set_xlim([1,7])
    ax[1].set_title('Rating for ' + y)
    print('The average for ' + y +'is: '+ str(np.average(qual_df[y])))

    plt.show()
    print(wilcoxon(qual_df[x], qual_df[y]))
```

Difficulty

In [201]:
```
plot_qual_function('control_difficulty', 'ephemeral_difficulty')
```

```
The average for control_difficultyis: 5.666666666666667
The average for ephemeral_difficultyis: 6.083333333333333
```



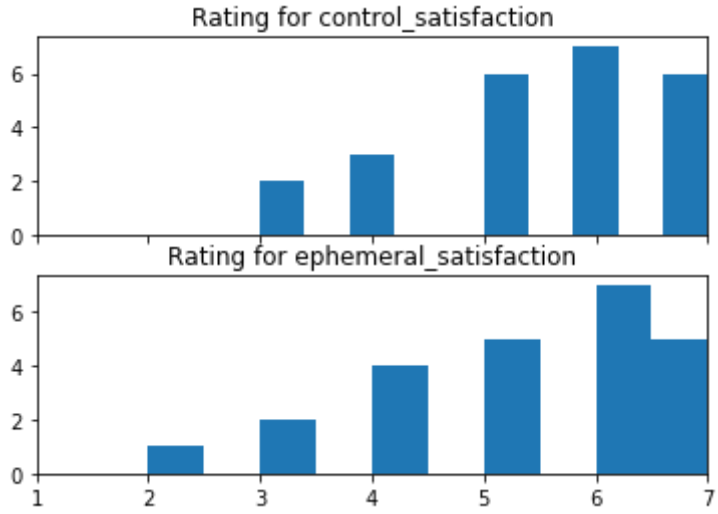

```
WilcoxonResult(statistic=7.5, pvalue=0.26319907816125776)

/Users/pwong/anaconda3/lib/python3.6/site-packages/scipy/stats/morestat
s.py:2388: UserWarning: Warning: sample size too small for normal appro
ximation.
  warnings.warn("Warning: sample size too small for normal approximatio
n.")
```

No significant difference on difficulty between the two conditions

Satisfaction

In [202]:
```
plot_qual_function('control_satisfaction', 'ephemeral_satisfaction')
```

```
The average for control_satisfactionis: 5.5
The average for ephemeral_satisfactionis: 5.25
```




```
WilcoxonResult(statistic=14.0, pvalue=0.1445852799373539)
```

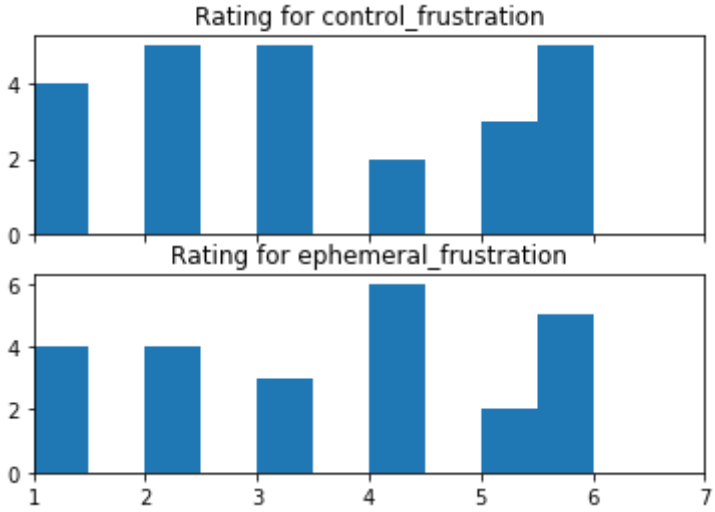
Significant difference between control and ephermal on satisfaction that people found Control condition more satisfying to use.

The average satisfaction rating for Control was 5.71 (out of 7), while the average rating for Ephermeral was 5.38 (out of 7). Control was more satisfying to use than Ephermeral (z=4.0, p = .034)

Frustration

In [203]:
```
plot_qual_function('control_frustration', 'ephemeral_frustration')
```

```
The average for control_frustrationis: 3.4166666666666665
The average for ephemeral_frustrationis: 3.5416666666666665
```



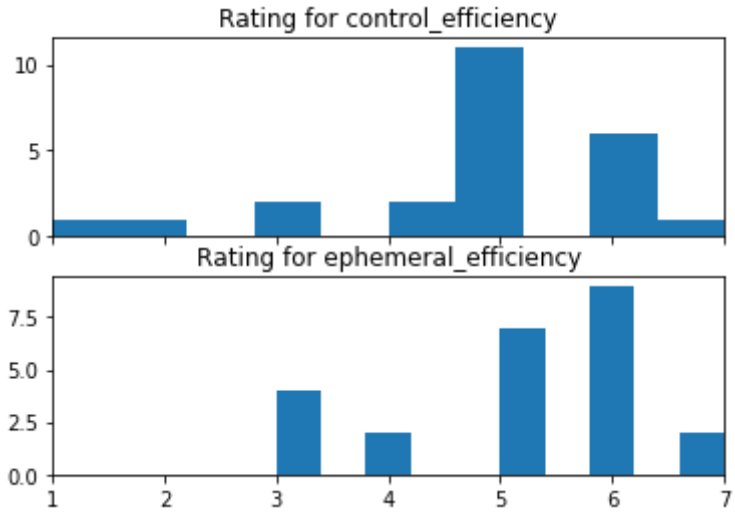

```
WilcoxonResult(statistic=26.0, pvalue=0.8755611072198808)
```

No significant difference between control and ephermal on frustration

Efficiency

In [204]:
```
plot_qual_function('control_efficiency', 'ephemeral_efficiency')
```

```
The average for control_efficiencyis: 4.791666666666667
The average for ephemeral_efficiencyis: 5.125
```




```
WilcoxonResult(statistic=19.5, pvalue=0.4026458293627345)
```

No significant difference between control and ephermal on efficiency

In [141]:
```
qual_df.columns
```

Out[141]:
```
Index(['participant_number', 'age', 'gender', 'menu_order',
       'overall_preference', 'overall_easy', 'control_satisfaction',
       'control_difficulty', 'control_frustration', 'control_efficienc
y',
       'ephemeral_satisfaction', 'ephemeral_difficulty',
       'ephemeral_frustration', 'ephemeral_efficiency'],
      dtype='object')
```