Table Schema

t1_user_active_min.csv This table contains active minutes data logged after experiment started. Each row represents the total number of minutes spent on site for each user on a date. If a user never visited the site for a given date, there wouldn't be data for that uid on that date.

- uid: user ID
- dt: date when corresponding active minutes are registered
- active_mins: number of minutes spent on site for the date

t2_user_variant.csv This table contains usersÕ treatment assignment. Each row represents the assignment information for a unique user.

- uid: user ID
- variant_number: the experiment variant user is in. 0 for control, 1 for treatment
- dt: date when user entered the experiment, should be Ô2019-02-06Õ for all users
- signup_date: the date string that user signed up on

t3_user_active_min_pre.csv This table contains active minutes data before the experiment started. It has a similar format as t1, except the dt range can extend before the experiment start date.

- uid: user ID
- dt: date when corresponding active minutes are registered
- active_mins: number of minutes spent on site for the date

t4_user_attributes.csv This table contains data about some user attributes. Each row represents attributes of a unique user.

- uid: user ID
- user_type: segment that a user belongs to, measured by activity level of the user. Can be Ônew_userÕ, Ônon_readerÕ, ÔreaderÕ or ÔcontributorÕ
- gender: user gender. Can be ÔmaleÕ, ÔfemaleÕ or ÔunknownÕ

Code Journal

Start off by importing important libraries we will need and loading our given data tables.

In [252]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

t1 = pd.read_csv("t1_user_active_min.csv")
t2 = pd.read_csv("t2_user_variant.csv")
t3 = pd.read_csv("t3_user_active_min_pre.csv")
t4 = pd.read_csv("t4_user_attributes.csv")
```

Take a quick look at our data to get an idea of it.

In [802]:
```
t1
```

Out[802]:

	uid	dt	active_mins
0	0	2019-02-22	5.0
1	0	2019-03-11	5.0
2	0	2019-03-18	3.0
3	0	2019-03-22	4.0
4	0	2019-04-03	9.0
...	...	...	...
1066397	49999	2019-04-14	24.0
1066398	49999	2019-04-26	1.0
1066399	49999	2019-05-31	6.0
1066400	49999	2019-06-02	2.0
1066401	49999	2019-06-24	5.0

1066402 rows × 3 columns

In [801]:
```
t3
```

Out[801]:

	uid	dt	active_mins
0	0	2018-09-24	3.0
1	0	2018-11-08	4.0
2	0	2018-11-24	3.0
3	0	2018-11-28	6.0
4	0	2018-12-02	6.0
...	...	...	...
1190088	49999	2018-09-15	5.0
1190089	49999	2018-09-26	8.0
1190090	49999	2018-10-20	29.0
1190091	49999	2018-12-14	3.0
1190092	49999	2019-01-28	32.0

1190093 rows × 3 columns

Looks pretty normal so far. I looked at all the tabels in Excel as well and noted that (from t2) there were 50,000 unique users, of which the first 40,000 were in the control group and the last 10,000 were in the experimental group. Important to note is that both tables 1 and 3 maxed out their rows in Excel which gave the appearance of them having equal data observations, but here we can see that it looks like we have more entries for the data pre-experiment which makes logical sense. Next, I wanted to explore this discrepancy a little more since it caught my eye.

In [245]:
```
print(t3.uid.nunique())
print(t3.dt.nunique())
print(t1.uid.nunique())
print(t1.dt.nunique())
```

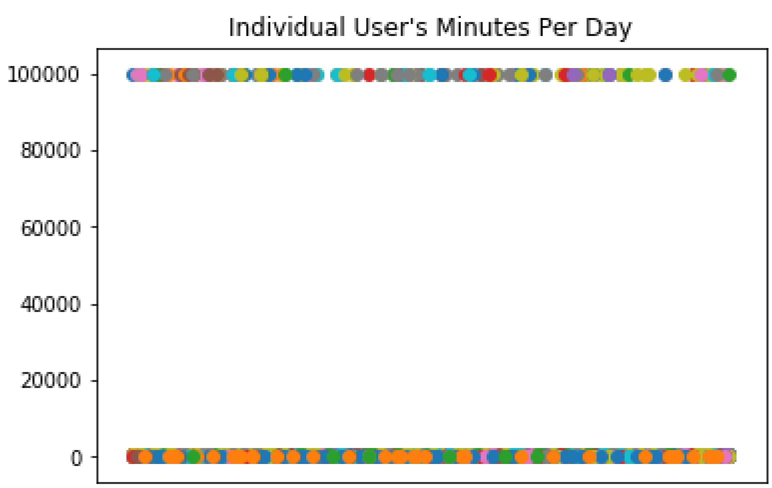
```
49697
180
46633
150
```

Now it is apparent to see that we have more users and more dates in the pre-experiment data, this is something to keep in mind as it could skew our calculations if we were to simply take total values of minutes spent on the Quora app. Even if one user spent x minutes everyday on the app pre-experiment and y minutes everyday post-experiment for some x < y, it could show as them spending more time pre-experiment, which would lead us to a false conclusion.

This graph was not strictly necessary but I got curious and wanted to see what it would look like if I plotted everyone's data.

In [77]:
```
plt.title("Individual User's Minutes Per Day")
cur_axes = plt.gca()
cur_axes.axes.get_xaxis().set_ticklabels([])
cur_axes.axes.get_xaxis().set_ticks([])
test1 = {}

for i in t1.uid.unique():
    test1[i] = t1[t1["uid"] == i]
    plt.scatter(test1[i]["dt"], test1[i]["active_mins"])
```



In [266]:
```
plt.close()
```

After doing so it became quite obvious that something was wrong with the data, note all the points up around 100,000 that obviously cannot be true. Naturally, I wanted to double check this graph's findings and see where these issues arose.

In [246]:
```
print(np.amax(t1))
print(np.amax(t3))
```

```
uid                    49999
dt                2019-07-05
active_mins            99999
dtype: object
uid                    49999
dt                2019-02-05
active_mins            99999
dtype: object
```

As seen, these errors are present in both tables and I double checked the table files I was given to make sure there was not some translation error when reading them in earlier in order to confirm. So, I wanted to get a list of all the locations where these 99999 numbers were showing up.

In [260]:
```
t1_errors = np.where(t1.active_mins == np.amax(t1.active_mins))
t3_errors = np.where(t3.active_mins == np.amax(t3.active_mins))
print(len(t1_errors[0]))
print(len(t3_errors[0]))
```

```
172
166
```

As of the time of doing this I wasn't aware this would be useful later on for question 2, this was all just a general inspection of the data that wasn't strictly necessary before I took on the questions themselves. I always feel it's helpful to get a general feel for what you're dealing with and deal with problems top-down rather than try to understand the data and the questions at the same time.

Now to create a new table without the erroneous rows of information, please note that if I were to implement this for real I would just edit the original data in order to make storage more efficient but for the sake of demonstration it was handy to make a copy.

In [495]:
```
t1_copy = t1
t1_copy = t1_copy.drop(t1_errors[0], axis=0)
t3_copy = t3
t3_copy = t3_copy.drop(t3_errors[0], axis=0)
```

In [262]:
```
t1.active_mins.describe()
```

Out[262]:
```
count    1.066402e+06
mean     3.616809e+01
std      1.270484e+03
min      1.000000e+00
25%      2.000000e+00
50%      5.000000e+00
75%      1.700000e+01
max      9.999900e+04
Name: active_mins, dtype: float64
```

In [263]:
```
t1_copy.active_mins.describe()
```

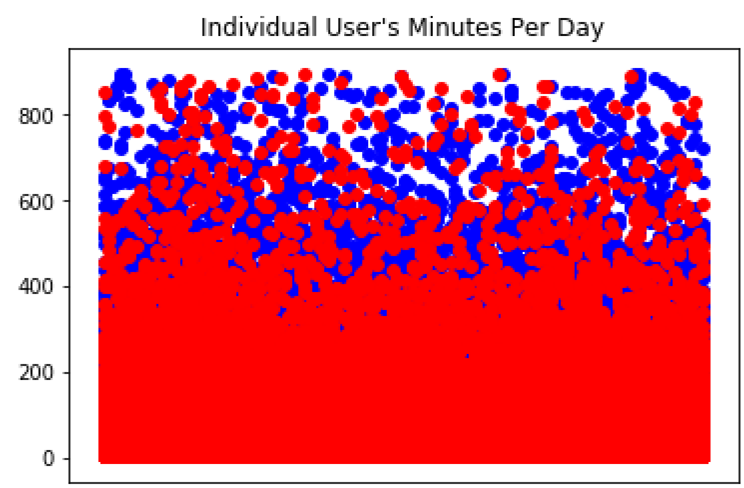
Out[263]:
```
count    1.066230e+06
mean     2.004248e+01
std      4.653763e+01
min      1.000000e+00
25%      2.000000e+00
50%      5.000000e+00
75%      1.700000e+01
max      8.970000e+02
Name: active_mins, dtype: float64
```

As seen above, our standard deviation is way down and our mean and max has also been reduced, signifying that our removal of error did indeed have the impact we wanted.

Again, not a necessary graph but when I have the free time to do so I like to get a good sense of our data and we can clearly see that we got rid of all our erroneous outliers.

In [287]:
```
plt.title("Individual User's Minutes Per Day")
cur_axes = plt.gca()
cur_axes.axes.get_xaxis().set_ticklabels([])
cur_axes.axes.get_xaxis().set_ticks([])
users = {}

for i in t1_copy.uid.unique():
    users[i] = t1_copy[t1_copy["uid"] == i]
    if t2.iloc[i].variant_number == 0:
        plt.scatter(users[i]["dt"], users[i]["active_mins"], c='blue')
    else:
        plt.scatter(users[i]["dt"], users[i]["active_mins"], c='red')
```

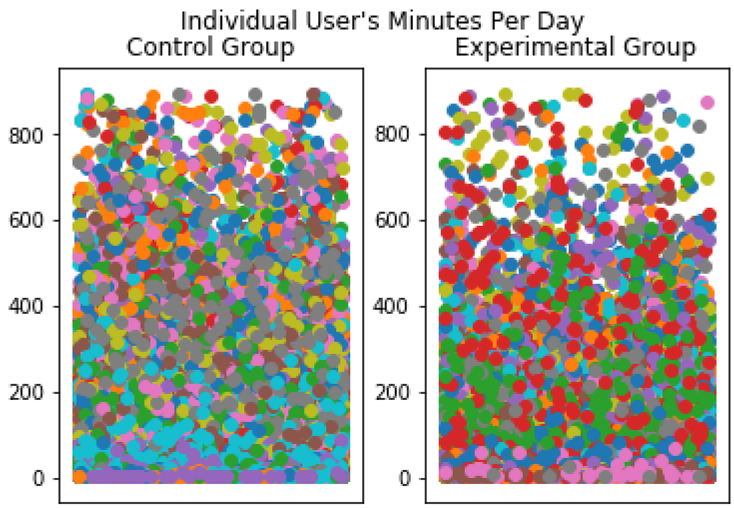


In [290]:
```
plt.close()
```

I tried to differentiate that graph by group to maybe take a peek at what our results might be, but unfortunately it didn't work out so well because of the volume of data points we have so I plotted the two side by side instead. Turns out that we get a nice Jackson Pollock painting but not a lot of information from them, so no luck here as far as eyeballing our results.

In [302]:
```
fig, (ax1, ax2) = plt.subplots(1,2)
fig.suptitle("Individual User's Minutes Per Day")
ax1.set_title("Control Group")
ax2.set_title("Experimental Group")
ax1.axes.get_xaxis().set_ticklabels([])
ax1.axes.get_xaxis().set_ticks([])
ax2.axes.get_xaxis().set_ticklabels([])
ax2.axes.get_xaxis().set_ticks([])

for i in t1_copy.uid.unique():
    users[i] = t1_copy[t1_copy["uid"] == i]
    if t2.iloc[i].variant_number == 0:
        ax1.scatter(users[i]["dt"], users[i]["active_mins"])
    else:
        ax2.scatter(users[i]["dt"], users[i]["active_mins"])
```



In [304]:
```
plt.close()
```

Question 2

Setting up two different DataFrames for the control and experimental group since we'll need that in the future. Again, not memory efficient but as this is more of a scratchbook/journal than an industry ready piece of code I'm not worrying about that. We already accounted for the errors before this so we don't need to do that again.

In [499]:
```
control = pd.DataFrame()
experimental = pd.DataFrame()

for i in t1_copy.uid.unique():
    if t2.iloc[i].variant_number == 0:
        control = control.append(t1_copy[t1_copy.uid == i])
    else:
        experimental = experimental.append(t1_copy[t1_copy.uid == i])
```

In [467]:
```
control.active_mins.describe()
```

Out[467]:
```
count    886815.000000
mean         19.337660
std          44.797631
```

```
min            1.000000
25%            2.000000
50%            5.000000
75%           16.000000
max          897.000000
Name: active_mins, dtype: float64
```

In [332]:
```
experimental.active_mins.describe()
```

Out[332]:
```
count    179415.000000
mean         23.526294
std          54.191356
min           1.000000
25%           3.000000
50%           7.000000
75%          19.000000
max         895.000000
Name: active_mins, dtype: float64
```

Create new dataframes for our two groups minutes per day, very easy to do from a dictionary. It might seem redundant to make a uid column too but it will come in handy if we need to access them (hint: we will later)

In [560]:
```
exp_per_user = {}
for i in experimental.uid.unique():
    exp_per_user[i] = [i, np.sum(experimental[experimental.uid == i].activ
e_mins)]

exp_df = pd.DataFrame.from_dict(exp_per_user, orient='index', columns=['ui
d','total_mins'])
exp_df
```

Out[560]:

	uid	total_mins
40000	40000	25.0
40001	40001	299.0
40002	40002	183.0
40004	40004	56.0
40005	40005	289.0
...	...	...
49995	49995	95.0
49996	49996	156.0
49997	49997	379.0
49998	49998	597.0
49999	49999	39.0

9208 rows × 2 columns

In [798]:
```
control_per_user = {}
```

```
for i in control.uid.unique():
    control_per_user[i] = [i, np.sum(control[control.uid == i].active_mins
)]

control_df = pd.DataFrame.from_dict(control_per_user, orient='index', colu
mns=['uid', 'total_mins'])
control_df.head()
```

Out[798]:

	uid	total_mins
0	0	43.0
1	1	15205.0
2	2	17.0
3	3	77.0
4	4	39.0

In [563]:

```
exp_df.total_mins.describe()
```

Out[563]:

```
count     9208.000000
mean       458.402476
std       1680.571091
min          1.000000
25%         23.000000
50%         71.000000
75%        227.000000
max      46742.000000
Name: total_mins, dtype: float64
```

In [564]:

```
control_df.total_mins.describe()
```

Out[564]:

```
count    37425.000000
mean       458.221162
std       1653.447132
min          1.000000
25%         16.000000
50%         52.000000
75%        191.000000
max      37191.000000
Name: total_mins, dtype: float64
```

Now we do calculations for our confidence interval in the difference of the means.

In [570]:

```
import math
def confidence_interval_means(A, B, z=1.96):
    A_mean = A.mean()
    B_mean = B.mean()
    A_std = A.std()
    B_std = B.std()
    pool_std = math.sqrt((A_std + B_std) / 2)
    temp = A_mean - B_mean
    temp2 = z*pool_std*math.sqrt(1/A.count() + 1/B.count())
    return [temp - temp2,temp + temp2]
```

In [571]:
```
confidence_interval_means(control_df.total_mins, exp_df.total_mins,)
```

Out[571]:
```
[-1.1122248128832295, 0.7495972467170047]
```

According to this, there is a 95% chance that the difference of the means of the two group's total minutes per user lies between -1.11 and .75. This is substantial enough evidence to state that there is no difference of mean total minutes per user spent.

Question 3

You decide to dive deeper into the data, so you gather a table of active minutes by user from before the experiment began. You should now use table 3 (t3_user_active_min_pre.csv) along with tables 1 and 2 for this question.

Using the statistical method of your choice and the pre-experiment data, update your 95% confidence interval of the overall average treatment effect.

I decided to go with another independent sample t-test this time and compare the total minute difference per user in experimental group (post-experiment minus pre-experiment) vs. the control group to see if there was a significant difference in means after the UI change went live.

In [652]:
```
pre_experimental = pd.DataFrame()
pre_control = pd.DataFrame()

for i in t3_copy.uid.unique():
    if t2.iloc[i].variant_number == 1:
        pre_experimental = pre_experimental.append(t3_copy[t3_copy.uid ==
i])
    else:
        pre_control = pre_control.append(t3_copy[t3_copy.uid == i])
```

In [653]:
```
pre_control_per_user = {}
for i in pre_control.uid.unique():
    pre_control_per_user[i] = [i, np.sum(pre_control[pre_control.uid == i]
.active_mins)]

pre_control_df = pd.DataFrame.from_dict(pre_control_per_user, orient='inde
x', columns=['uid','total_mins'])
```

In [797]:
```
pre_exp_per_user = {}
for i in pre_experimental.uid.unique():
    pre_exp_per_user[i] = [i, np.sum(pre_experimental[pre_experimental.uid
 == i].active_mins)]

pre_exp_df = pd.DataFrame.from_dict(pre_exp_per_user, orient='index', colu
mns=['uid','total_mins'])
pre_exp_df.head()
```

Out[797]:

	uid	total_mins

40001	40001	125.0
40002	40002	90.0
40003	40003	18.0
40004	40004	10.0
40005	40005	638.0

In [796]:

```
exp_df.head()
```

Out[796]:

	uid	total_mins
40000	40000	25.0
40001	40001	299.0
40002	40002	183.0
40004	40004	56.0
40005	40005	289.0

Well it will be hard to find differences with some missing uid's between the two groups. I'm going to adjust it so that for any user id in A that's not in B it's shown with a value of 0 in B and vice versa. This will preserve data integrity but also let us get a difference between them. First are the experimental groups.

In [574]:

```
# This could also be done by converting the lists into sets and finding th
e difference that way.
example = {}
for i in pre_experimental.uid.unique():
    if i not in experimental.uid.unique():
        example[i] = [i,0]
    else:
        continue
```

In [575]:

```
example2 = {}
for i in experimental.uid.unique():
    if i not in pre_experimental.uid.unique():
        example2[i] = [i,0]
    else:
        continue
```

In [576]:

```
temp = pd.DataFrame.from_dict(example, orient='index', columns=['uid','tot
al_mins'])
exp_df_copy = exp_df.append(temp)
exp_df_copy
```

Out[576]:

	uid	total_mins
40000	40000	25.0
40001	40001	299.0
40002	40002	183.0

40004	40004	56.0
40005	40005	289.0
...	...	...
49936	49936	0.0
49954	49954	0.0
49966	49966	0.0
49967	49967	0.0
49970	49970	0.0

9964 rows × 2 columns

In [577]:

```
temp2 = pd.DataFrame.from_dict(example2, orient='index', columns=['uid','t
otal_mins'])
pre_exp_df_copy = pre_exp_df.append(temp2)
pre_exp_df_copy
```

Out[577]:

	uid	total_mins
40001	40001	125.0
40002	40002	90.0
40003	40003	18.0
40004	40004	10.0
40005	40005	638.0
...	...	...
48858	48858	0.0
49255	49255	0.0
49373	49373	0.0
49703	49703	0.0
49810	49810	0.0

9964 rows × 2 columns

Now the control groups.

In [654]:

```
example = {}
for i in pre_control.uid.unique():
    if i not in control.uid.unique():
        example[i] = [i,0]
    else:
        continue
```

In [655]:

```
example2 = {}
for i in control.uid.unique():
```

```
    if i not in pre_control.uid.unique():
        example2[i] = [i,0]
    else:
        continue
```

In [795]:
```
temp = pd.DataFrame.from_dict(example, orient='index', columns=['uid','total_mins'])
control_df_copy = control_df.append(temp)
control_df_copy.head()
```

Out[795]:

	uid	total_mins
0	0	43.0
1	1	15205.0
2	2	17.0
3	3	77.0
4	4	39.0

In [794]:
```
temp2 = pd.DataFrame.from_dict(example2, orient='index', columns=['uid','total_mins'])
pre_control_df_copy = pre_control_df.append(temp2)
pre_control_df_copy.head()
```

Out[794]:

	uid	total_mins
0	0	70.0
1	1	19158.0
2	2	37.0
3	3	108.0
4	4	66.0

Looks like we've fixed our mismatch problem in the experimental groups. Just to make sure let's double check. And after we can do the control groups.

In [580]:
```
def Diff(li1, li2):
    return (list(set(li1) - set(li2)))
```

In [667]:
```
print(Diff(pre_exp_df_copy.uid, exp_df_copy.uid))
print(Diff(pre_control_df_copy.uid, control_df_copy.uid))
```

```
[]
[]
```

Voila, we've made sure we have the same uid's in both groups. Now let's find those differences.

In [664]:
```
temp = {}
#We can safely iterate through only one set of uid's because we made sure
```

```
that they are all the same for both groups.
for i in pre_exp_df_copy.uid:
    p = pre_exp_df_copy.loc[i].at['total_mins'] #pre-exp total mins for th
at uid
    e = exp_df_copy.loc[i].at['total_mins'] #post-exp total mins for that
uid
    temp[i] = [i, e-p] #e - p will give the difference in total minutes fo
r each uid post-exp

diff_exgroup = pd.DataFrame.from_dict(temp, orient='index', columns=['uid'
,'total_mins'])
```

In [793]:
```
diff_exgroup.head()
```

Out[793]:

	uid	total_mins
40001	40001	174.0
40002	40002	93.0
40003	40003	-18.0
40004	40004	46.0
40005	40005	-349.0

In [642]:
```
diff_exgroup.total_mins.describe()
```

Out[642]:
```
count      9964.000000
mean        150.086511
std         980.069940
min      -21776.000000
25%         -13.000000
50%           8.000000
75%          71.000000
max       24636.000000
Name: total_mins, dtype: float64
```

In [658]:
```
temp = {}
#We can safely iterate through only one set of uid's because we made sure
that they are all the same for both groups.
for i in pre_control_df_copy.uid:
    p = pre_control_df_copy.loc[i].at['total_mins'] #pre-exp total mins fo
r that uid
    e = control_df_copy.loc[i].at['total_mins'] #post-exp total mins for t
hat uid
    temp[i] = [i, e-p] #e - p will give the difference in total minutes fo
r each uid post-exp

diff_cgroup = pd.DataFrame.from_dict(temp, orient='index', columns=['uid',
'total_mins'])
```

In [962]:
```
diff_cgroup.head()
```

Out[962]:

uid	total_mins

0	0	-27.0
1	1	-3953.0
2	2	-20.0
3	3	-31.0
4	4	-27.0

In [661]:
```
diff_cgroup.total_mins.describe()
```

Out[661]:
```
count    39888.000000
mean       -46.391772
std        938.919155
min     -24053.000000
25%        -43.000000
50%         -6.000000
75%         18.000000
max      20942.000000
Name: total_mins, dtype: float64
```

Now we have a dataframe of all our user's increase/decrease in minutes post-experiment and also our standard deviations for the two are similar which is always nice.

In [662]:
```
confidence_interval_means(diff_cgroup.total_mins, diff_exgroup.total_mins)
```

Out[662]:
```
[-197.15823912040804, -195.79832768495282]
```

Well that's a stark difference from what our previous test showed. It appears that we had some confounding variables that the last test didn't account for. Just to double check I wanted to do the test again but this time just ignoring uid's that aren't in both instead of putting in placeholder 0's. This should only reinforce our previous conclusion.

In [669]:
```
def intersection(li1, li2):
    return list(set(li1) & set(li2))
```

In [679]:
```
shared_cuids = intersection(pre_control.uid, control.uid)
shared_euids = intersection(pre_experimental.uid, experimental.uid)
```

In [696]:
```
# After playing around with different function implementations, it turns o
ut that this is actually a lot more
# efficient than what I was doing before, but I already ran that part so I
 didn't feel it was necessary to go back
# and change it just for the sake of appearances. Learning and applying ne
w knowledge on the fly is also important!
pre_cdiction = {}
cdiction = {}
for i in shared_cuids:
    pre_cdiction[i] = [i, pre_control_df.loc[i].at['total_mins']]
    cdiction[i] = [i, control_df.loc[i].at['total_mins']]

shared_pre_cgroup = pd.DataFrame.from_dict(pre_cdiction, orient='index', c
olumns=['uid','total_mins'])
```

```
shared_cgroup = pd.DataFrame.from_dict(cdiction, orient='index', columns=[
'uid','total_mins'])
```

In [706]:
```
pre_ediction = {}
ediction = {}
for i in shared_euids:
    pre_ediction[i] = [i, pre_exp_df.loc[i].at['total_mins']]
    ediction[i] = [i, exp_df.loc[i].at['total_mins']]

shared_pre_egroup = pd.DataFrame.from_dict(pre_ediction, orient='index', c
olumns=['uid','total_mins'])
shared_egroup = pd.DataFrame.from_dict(ediction, orient='index', columns=[
'uid','total_mins'])
```

In [709]:
```
temp = {}
for i in shared_euids:
    p = shared_pre_egroup.loc[i].at['total_mins'] #pre-exp total mins for
that uid
    e = shared_egroup.loc[i].at['total_mins'] #post-exp total mins for tha
t uid
    temp[i] = [i, e-p] #e - p will give the difference in total minutes fo
r each uid post-exp

shared_diff_exgroup = pd.DataFrame.from_dict(temp, orient='index', columns
=['uid','total_mins'])
```

In [715]:
```
shared_diff_exgroup.total_mins.describe()
```

Out[715]:
```
count     9165.000000
mean       164.654119
std       1020.378387
min     -21776.000000
25%        -13.000000
50%         12.000000
75%         82.000000
max      24636.000000
Name: total_mins, dtype: float64
```

In [711]:
```
temp = {}
for i in shared_cuids:
    p = shared_pre_cgroup.loc[i].at['total_mins'] #pre-exp total mins for
that uid
    e = shared_cgroup.loc[i].at['total_mins'] #post-exp total mins for tha
t uid
    temp[i] = [i, e-p] #e - p will give the difference in total minutes fo
r each uid post-exp

shared_diff_cgroup = pd.DataFrame.from_dict(temp, orient='index', columns=
['uid','total_mins'])
```

In [960]:
```
shared_diff_cgroup.total_mins.describe()
```

Out[960]:
```
count    37313.000000
mean       -47.295447
std        968.271500
```

```
min      -24053.000000
25%         -47.000000
50%          -6.000000
75%          22.000000
max       20942.000000
Name: total_mins, dtype: float64
```

In [713]:
```
confidence_interval_means(shared_diff_cgroup.total_mins, shared_diff_exgro
up.total_mins)
```

Out[713]:
```
[-212.6700882045657, -211.22904291122538]
```

That seems to confirm what we saw before, it's even more pronounced if we choose this method of accounting for the uid discrepancy. I chose to enter the first confidence interval as my answer but either or would suffice to prove the point.

Question 4

In real life, experiment results can be nuanced. We provide you now additionally table 4 (t4_user_attributes.csv), which might help you analyze the results better. You should think about the context of the experiment and hypothesize why the analysis above could be insufficient. Explore the data and answer the following questions.

I felt it would be helpful for our engineers if we could visualize the different user's minutes spent in terms of a distribution so we're going to use seaborn to check that out further now that we have access to the user's attributes. This will also let us look at treatment effect for different covariates.

In [755]:
```
import seaborn as sns
# We get some nonharmful warnings, just wanted to clear up the page a litt
le.
import warnings
warnings.filterwarnings('ignore')
def dist_plot(dataframe1,dataframe2,column1,column2,xaxistitle,data_label1
,data_label2):
    sns.distplot(dataframe1[column1], hist=False, label = data_label1, axl
abel = xaxistitle)
    sns.distplot(dataframe2[column2], hist=False, label = data_label2, axl
abel = xaxistitle)
    plt.show()
```

In [1038]:
```
t4.head()
```

Out[1038]:

	uid	gender	user_type
0	0	male	non_reader
1	1	male	reader
2	2	male	non_reader
3	3	male	non_reader
4	4	male	non_reader

By Gender & User Type

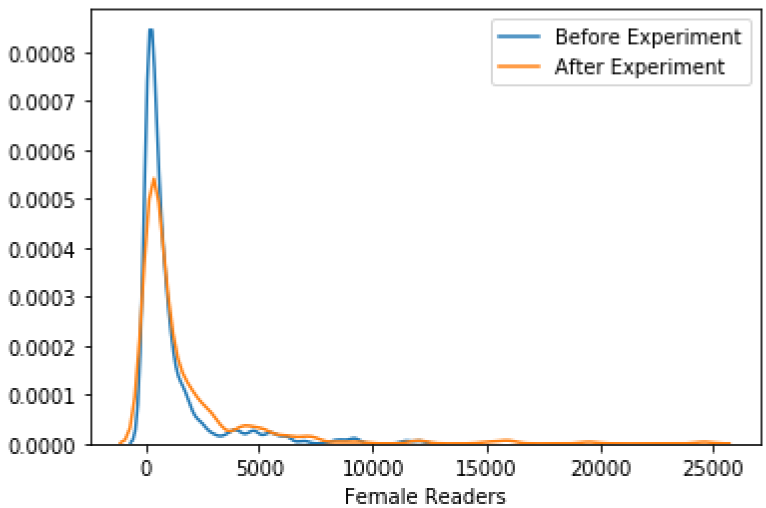
Note that each co-variate group's graphs will be shown in the form:

```
- Experimental group before and after the experiment
- Control vs Experimental groups after the experiment
```

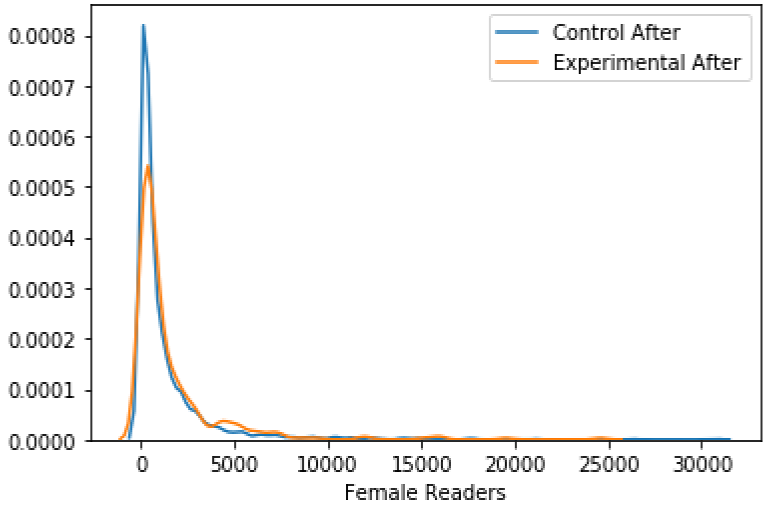
and that the title on the x-axis is the title of the graph, it's actually plotted as the number of minutes on the x-axis and the inverse of the count on the y-axis (it's a histogram just without the bars). It's just showing probability distributions so sometimes they will go negative too even though it's not possible to have negative minutes spent on the app.

Readers

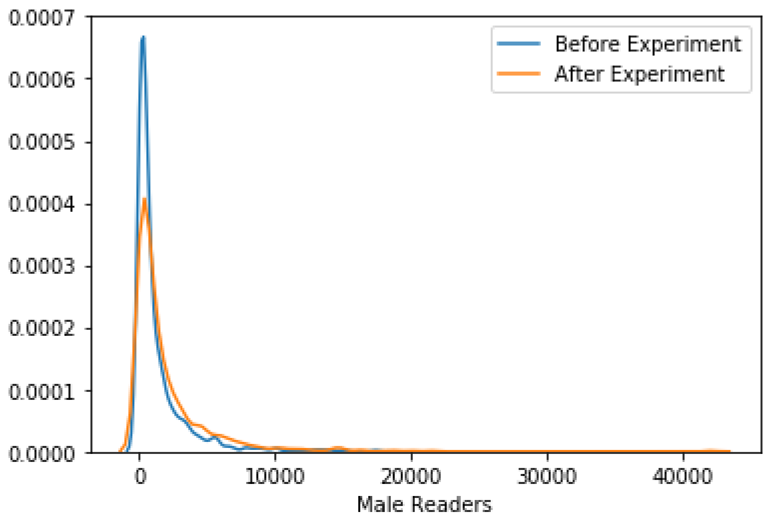
In [856]:
```
dist_plot(pre_exp_df[(t4["gender"] == "female")&(t4["user_type"] == "reade
r")],
           exp_df[(t4["gender"] == "female")&(t4["user_type"] == "reader")
],
          "total_mins","total_mins","Female Readers", "Before Experiment",
"After Experiment")
```



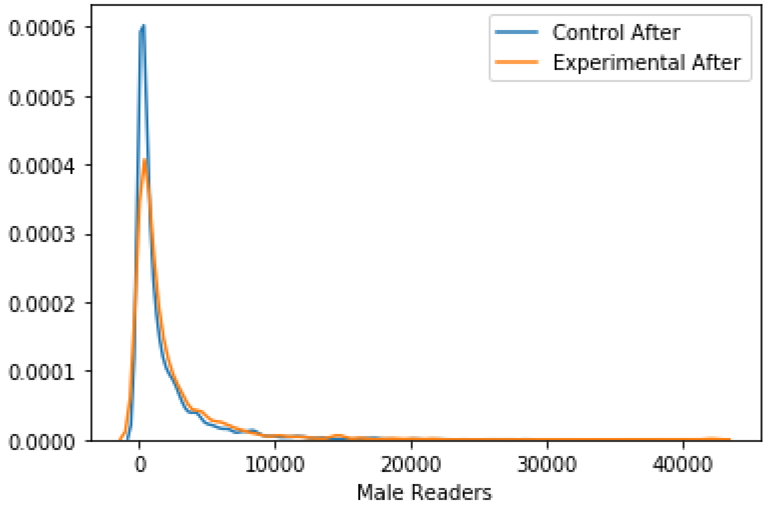
In [1061]:
```
dist_plot(control_df[(t4["gender"] == "female")&(t4["user_type"] == "read
er")],
           exp_df[(t4["gender"] == "female")&(t4["user_type"] == "reader"
)],
          "total_mins","total_mins","Female Readers", "Control After","Ex
perimental After")
```



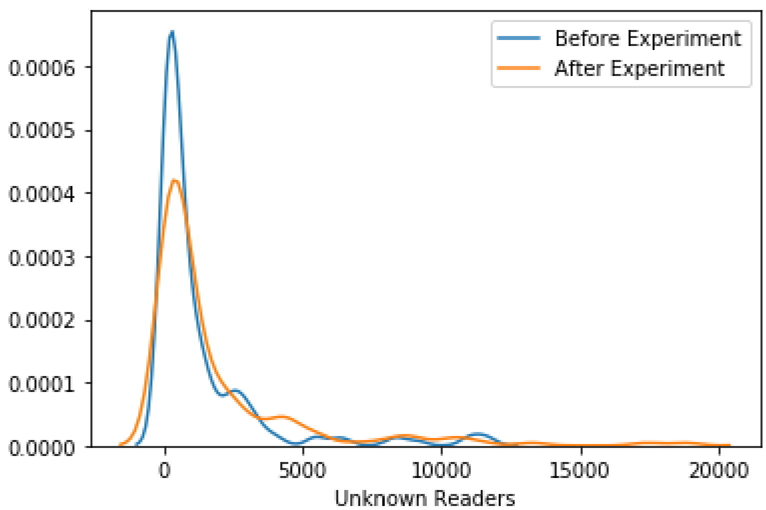
In [857]:
```
dist_plot(pre_exp_df[(t4["gender"] == "male")&(t4["user_type"] == "reader"
)],
             exp_df[(t4["gender"] == "male")&(t4["user_type"] == "reader")],
            "total_mins","total_mins","Male Readers", "Before Experiment","A
fter Experiment")
```



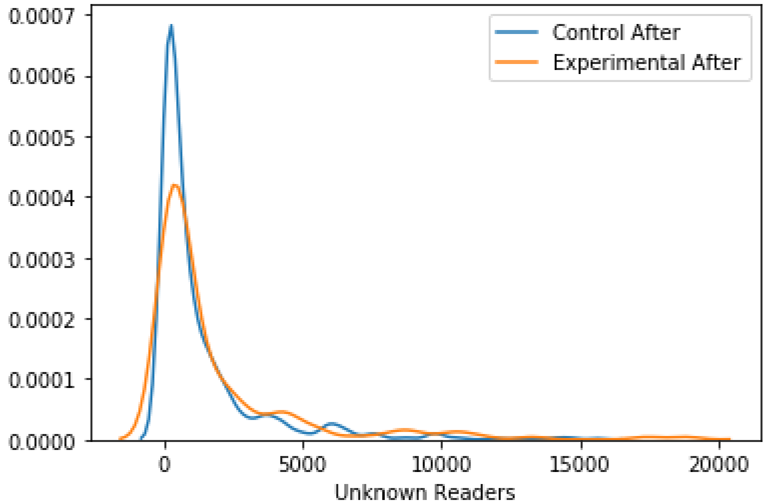
In [1060]:
```
dist_plot(control_df[(t4["gender"] == "male")&(t4["user_type"] == "reader
")],
             exp_df[(t4["gender"] == "male")&(t4["user_type"] == "reader")]
,
            "total_mins","total_mins","Male Readers", "Control After","Expe
rimental After")
```



In [858]:
```
dist_plot(pre_exp_df[(t4["gender"] == "unknown")&(t4["user_type"] == "read
er")],
             exp_df[(t4["gender"] == "unknown")&(t4["user_type"] == "reader"
)],
            "total_mins","total_mins","Unknown Readers", "Before Experiment"
,"After Experiment")
```

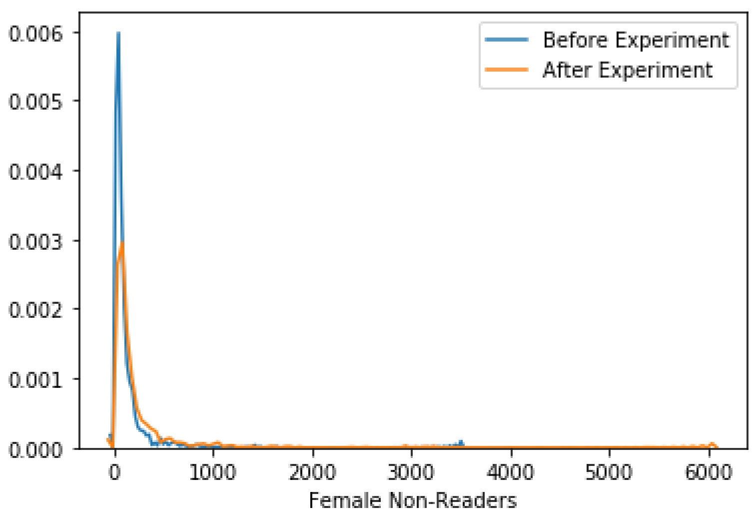


In [1059]:
```
dist_plot(control_df[(t4["gender"] == "unknown")&(t4["user_type"] == "rea
der")],
             exp_df[(t4["gender"] == "unknown")&(t4["user_type"] == "reader
")],
            "total_mins","total_mins","Unknown Readers", "Control After","E
xperimental After")
```

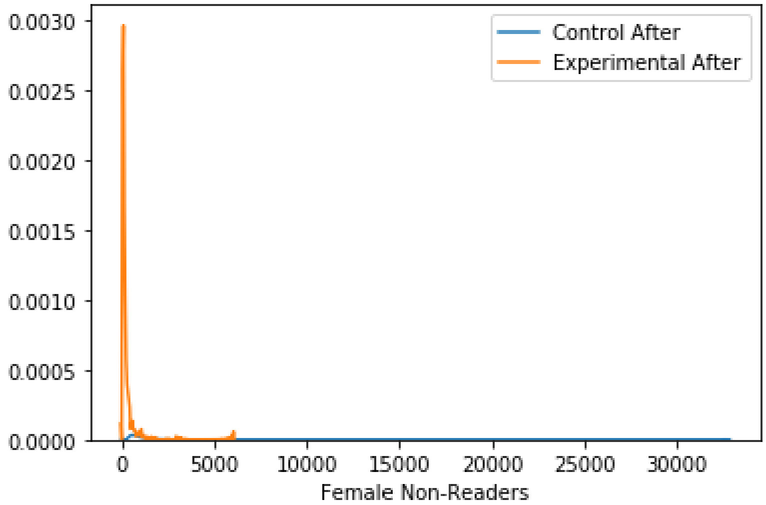


Non-Readers

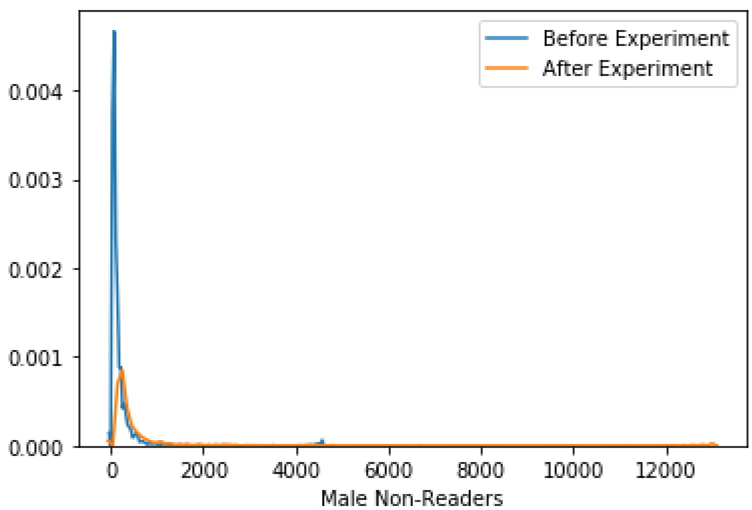
In [859]:
```
dist_plot(pre_exp_df[(t4["gender"] == "female")&(t4["user_type"] == "non_reader")],
           exp_df[(t4["gender"] == "female")&(t4["user_type"] == "non_reader")],
          "total_mins","total_mins","Female Non-Readers", "Before Experiment","After Experiment")
```



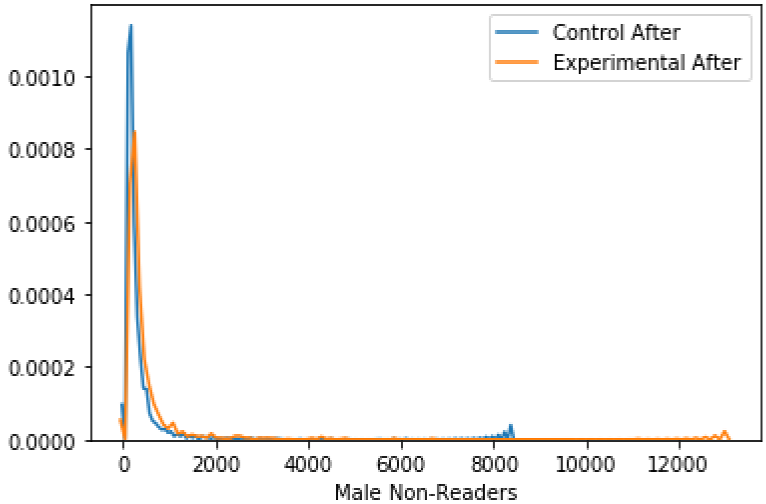
In [1058]:
```
dist_plot(control_df[(t4["gender"] == "female")&(t4["user_type"] == "non_reader")],
           exp_df[(t4["gender"] == "female")&(t4["user_type"] == "non_reader")],
          "total_mins","total_mins","Female Non-Readers", "Control After","Experimental After")
```



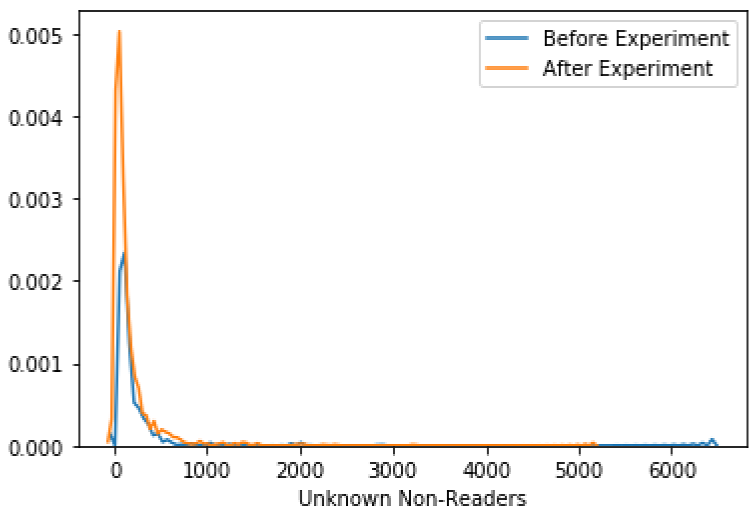
In [860]:
```
dist_plot(pre_exp_df[(t4["gender"] == "male")&(t4["user_type"] == "non_rea
der")],
            exp_df[(t4["gender"] == "male")&(t4["user_type"] == "non_reader
")],
           "total_mins","total_mins","Male Non-Readers", "Before Experiment
","After Experiment")
```



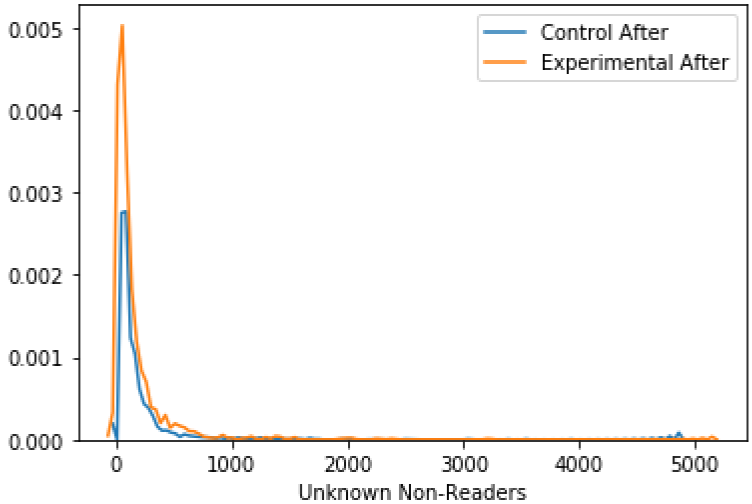
In [1057]:
```
dist_plot(control_df[(t4["gender"] == "male")&(t4["user_type"] == "non_re
ader")],
            exp_df[(t4["gender"] == "male")&(t4["user_type"] == "non_reade
r")],
           "total_mins","total_mins","Male Non-Readers", "Control After","
Experimental After")
```



In [861]:
```
dist_plot(pre_exp_df[(t4["gender"] == "unknown")&(t4["user_type"] == "non_
reader")],
          exp_df[(t4["gender"] == "unknown")&(t4["user_type"] == "non_rea
der")],
         "total_mins","total_mins","Unknown Non-Readers", "Before Experim
ent","After Experiment")
```

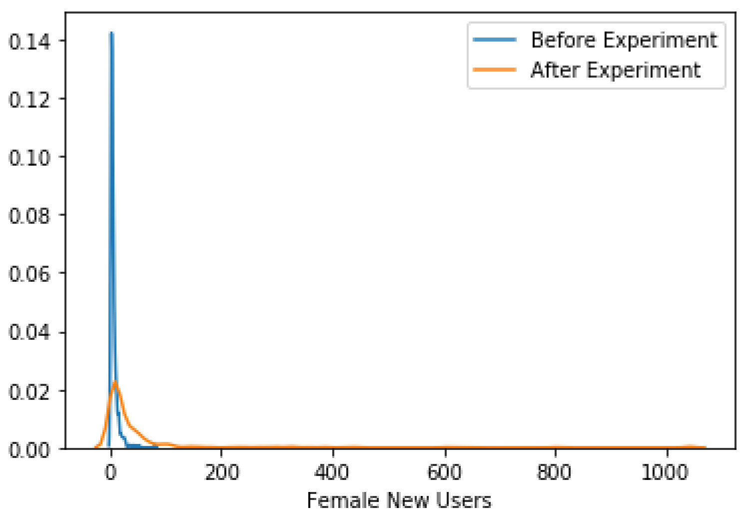


In [1056]:
```
dist_plot(control_df[(t4["gender"] == "unknown")&(t4["user_type"] == "non
_reader")],
          exp_df[(t4["gender"] == "unknown")&(t4["user_type"] == "non_re
ader")],
         "total_mins","total_mins","Unknown Non-Readers", "Control After
","Experimental After")
```

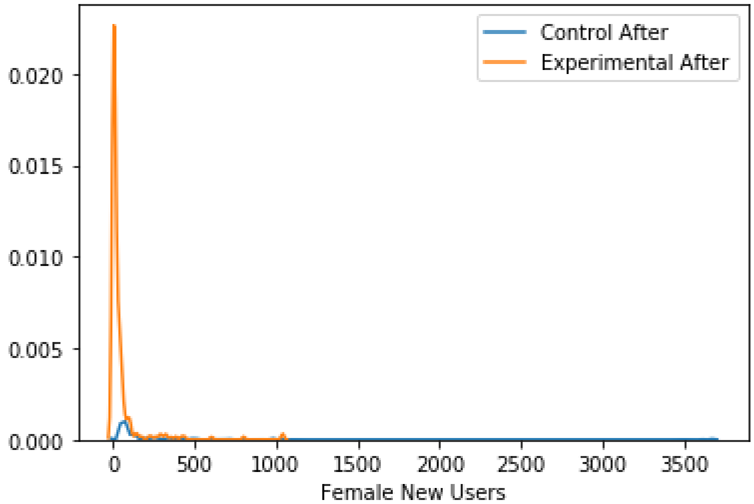


New Users

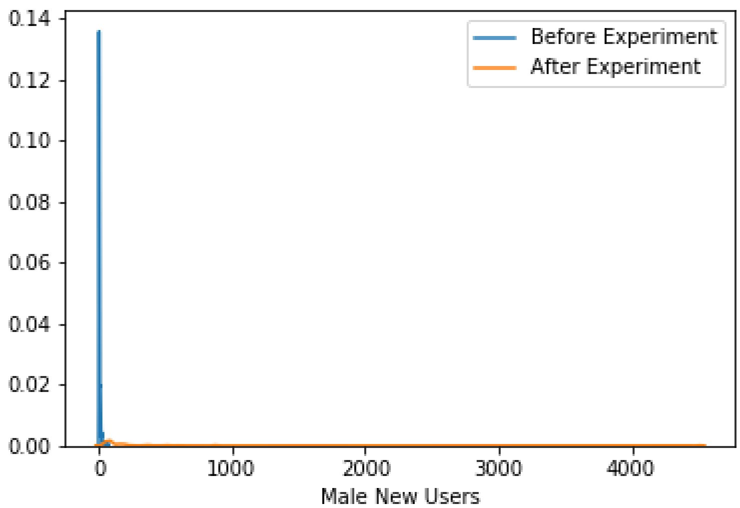
In [862]:
```
dist_plot(pre_exp_df[(t4["gender"] == "female")&(t4["user_type"] == "new_user")],
           exp_df[(t4["gender"] == "female")&(t4["user_type"] == "new_user")],
          "total_mins","total_mins","Female New Users", "Before Experiment","After Experiment")
```



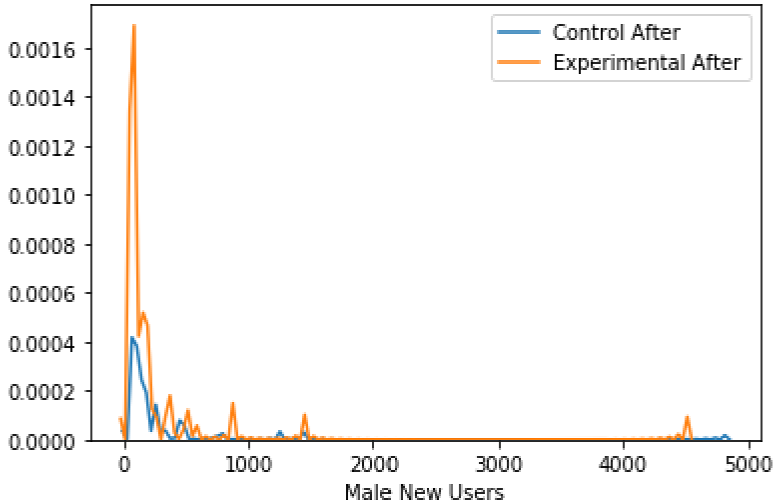
In [1055]:
```
dist_plot(control_df[(t4["gender"] == "female")&(t4["user_type"] == "new_user")],
           exp_df[(t4["gender"] == "female")&(t4["user_type"] == "new_user")],
          "total_mins","total_mins","Female New Users", "Control After","Experimental After")
```



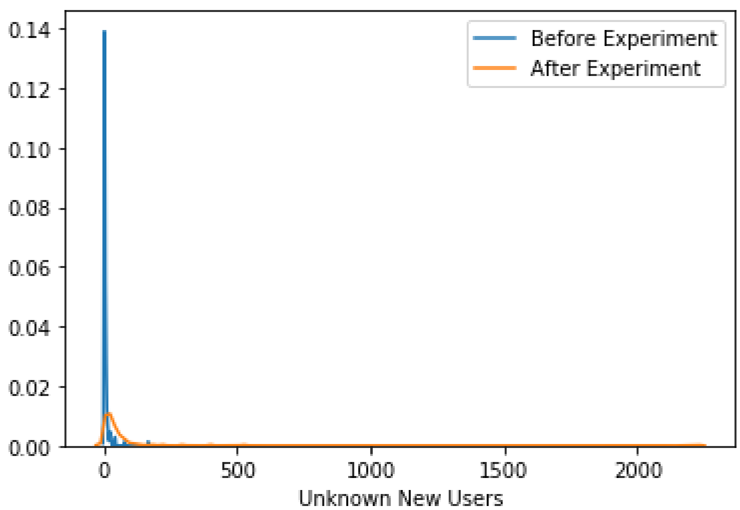
In [863]:
```
dist_plot(pre_exp_df[(t4["gender"] == "male")&(t4["user_type"] == "new_use
r")],
              exp_df[(t4["gender"] == "male")&(t4["user_type"] == "new_user")
],
             "total_mins","total_mins","Male New Users", "Before Experiment",
"After Experiment")
```



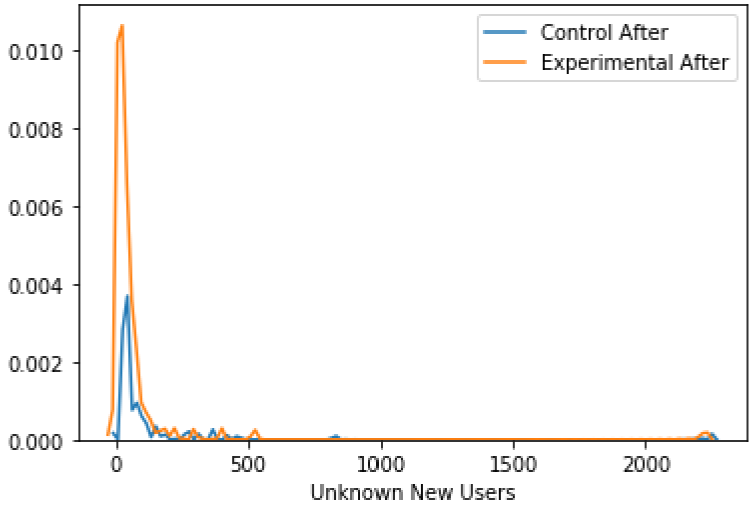
In [1054]:
```
dist_plot(control_df[(t4["gender"] == "male")&(t4["user_type"] == "new_us
er")],
              exp_df[(t4["gender"] == "male")&(t4["user_type"] == "new_user"
)],
             "total_mins","total_mins","Male New Users", "Control After","Ex
perimental After")
```



In [864]:
```
dist_plot(pre_exp_df[(t4["gender"] == "unknown")&(t4["user_type"] == "new_user")],
          exp_df[(t4["gender"] == "unknown")&(t4["user_type"] == "new_user")],
         "total_mins","total_mins","Unknown New Users", "Before Experiment","After Experiment")
```

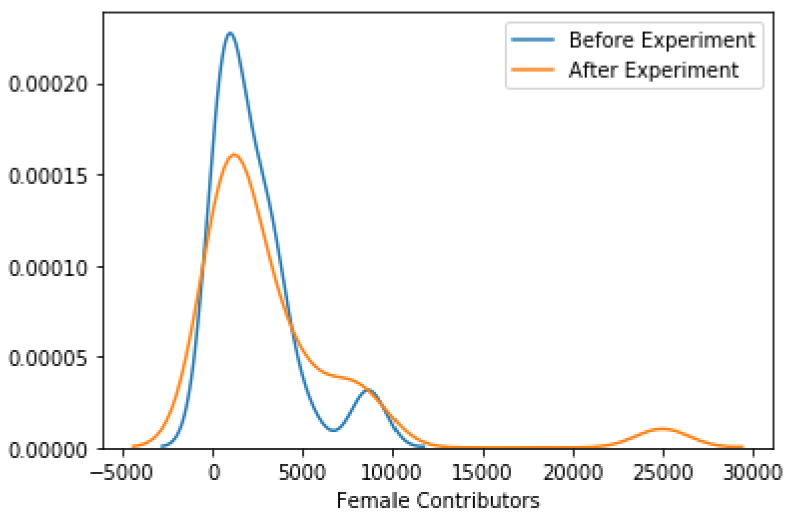


In [1053]:
```
dist_plot(control_df[(t4["gender"] == "unknown")&(t4["user_type"] == "new_user")],
          exp_df[(t4["gender"] == "unknown")&(t4["user_type"] == "new_user")],
         "total_mins","total_mins","Unknown New Users", "Control After", "Experimental After")
```

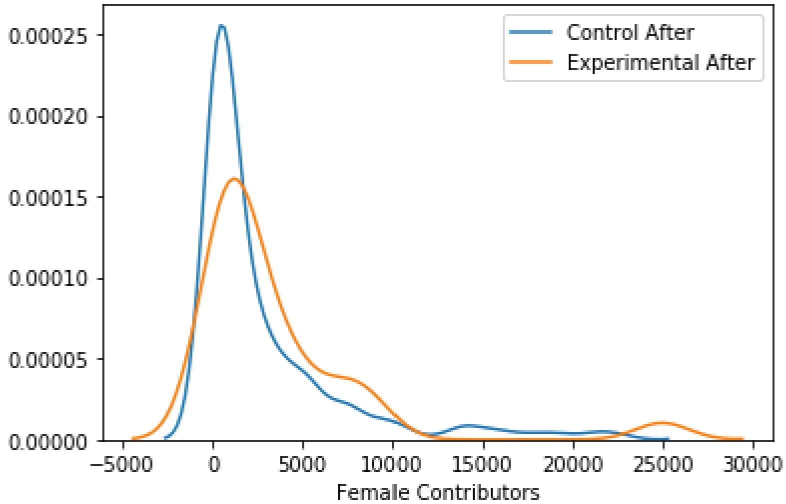


Contributors

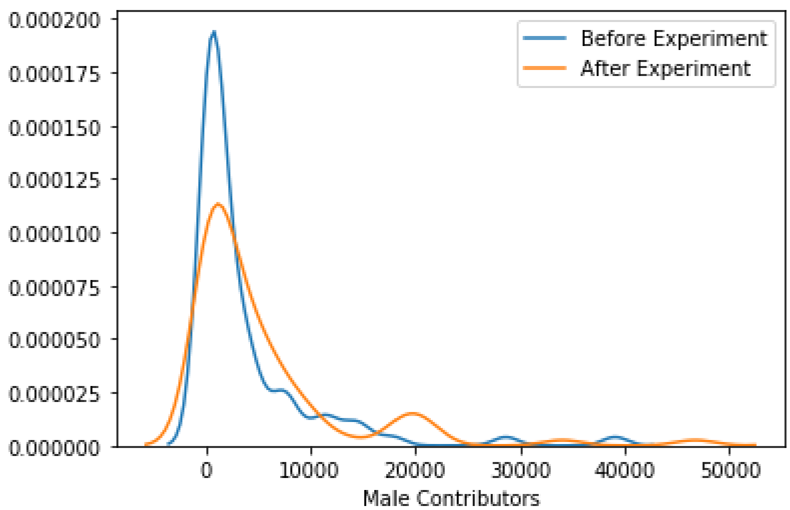
In [865]:
```
dist_plot(pre_exp_df[(t4["gender"] == "female")&(t4["user_type"] == "contr
ibutor")],
          exp_df[(t4["gender"] == "female")&(t4["user_type"] == "contribu
tor")],
         "total_mins","total_mins","Female Contributors", "Before Experim
ent","After Experiment")
```



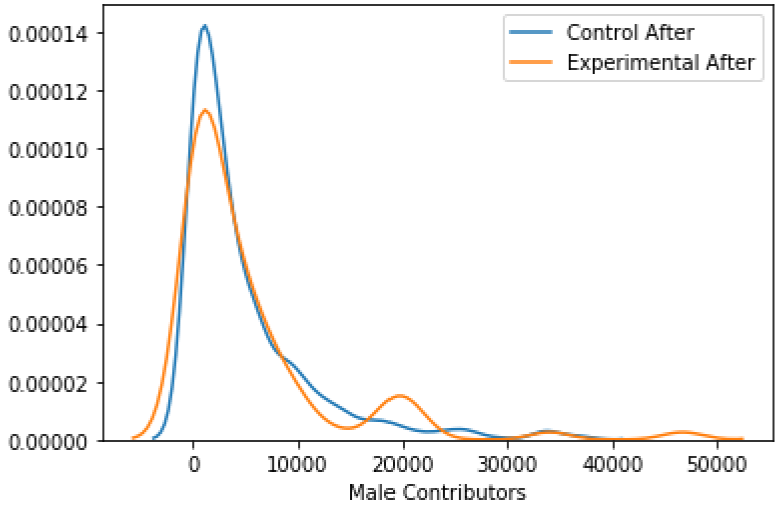
In [1052]:
```
dist_plot(control_df[(t4["gender"] == "female")&(t4["user_type"] == "cont
ributor")],
          exp_df[(t4["gender"] == "female")&(t4["user_type"] == "contrib
utor")],
         "total_mins","total_mins","Female Contributors", "Control After
","Experimental After")
```



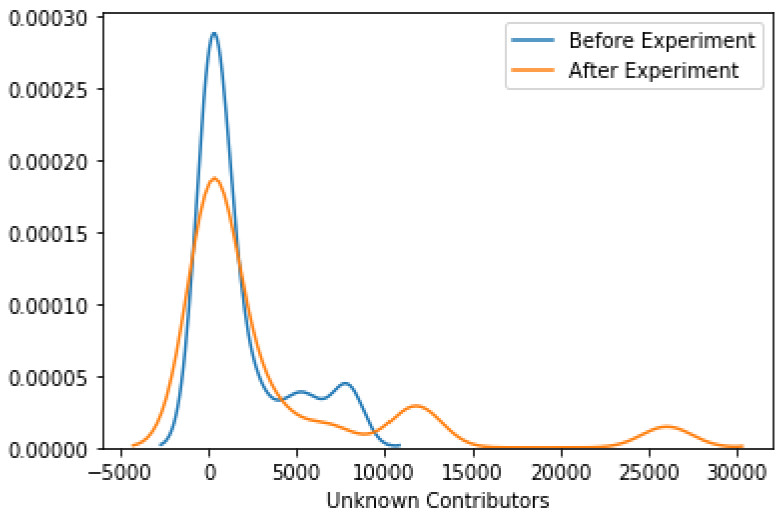
In [866]:
```
dist_plot(pre_exp_df[(t4["gender"] == "male")&(t4["user_type"] == "contrib
utor")],
           exp_df[(t4["gender"] == "male")&(t4["user_type"] == "contributo
r")],
          "total_mins","total_mins","Male Contributors", "Before Experimen
t","After Experiment")
```



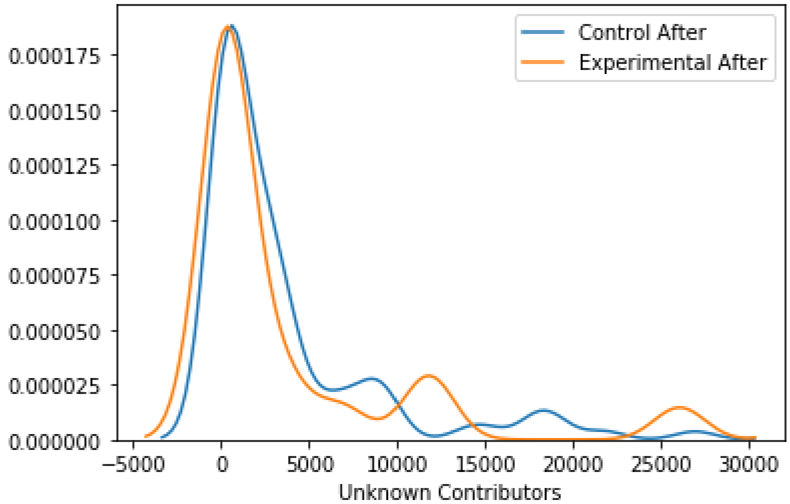
In [1051]:
```
dist_plot(control_df[(t4["gender"] == "male")&(t4["user_type"] == "contri
butor")],
           exp_df[(t4["gender"] == "male")&(t4["user_type"] == "contribut
or")],
          "total_mins","total_mins","Male Contributors", "Control After",
"Experimental After")
```



In [1050]:
```
dist_plot(pre_exp_df[(t4["gender"] == "unknown")&(t4["user_type"] == "contributor")],
             exp_df[(t4["gender"] == "unknown")&(t4["user_type"] == "contributor")],
            "total_mins","total_mins","Unknown Contributors", "Before Experiment","After Experiment")
```

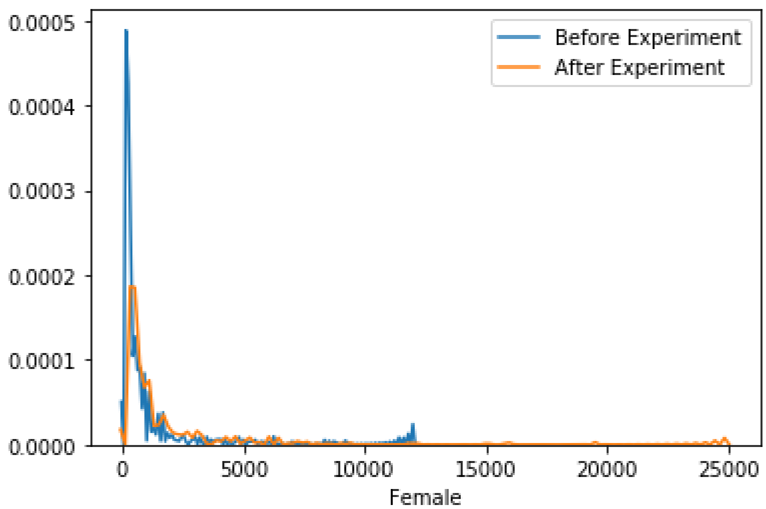


In [1049]:
```
dist_plot(control_df[(t4["gender"] == "unknown")&(t4["user_type"] == "contributor")],
             exp_df[(t4["gender"] == "unknown")&(t4["user_type"] == "contributor")],
            "total_mins","total_mins","Unknown Contributors", "Control After","Experimental After")
```

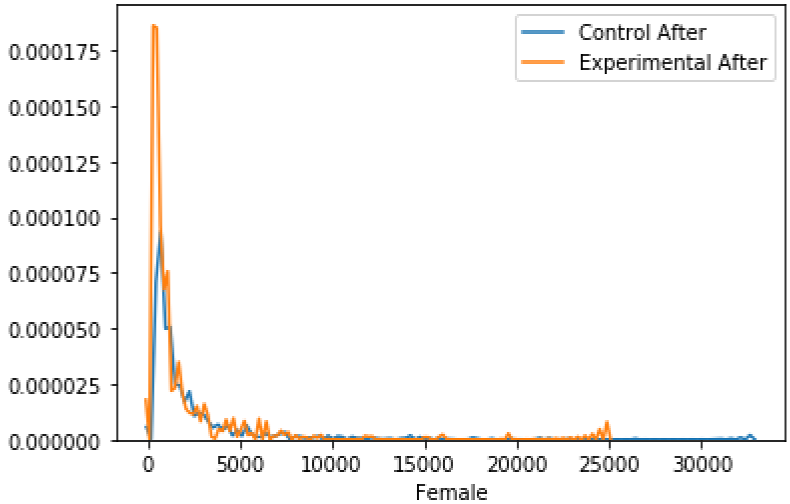


By Gender

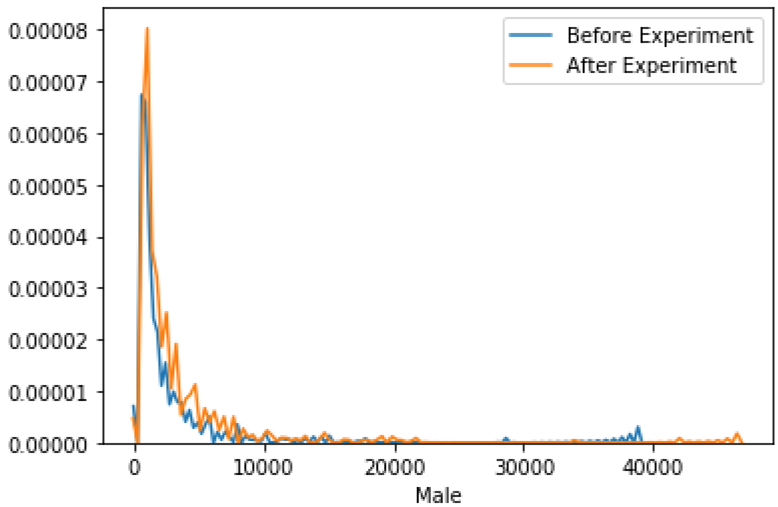
In [868]:
```
dist_plot(pre_exp_df[(t4["gender"] == "female")],
          exp_df[(t4["gender"] == "female")],
         "total_mins","total_mins","Female", "Before Experiment","After E
xperiment")
```



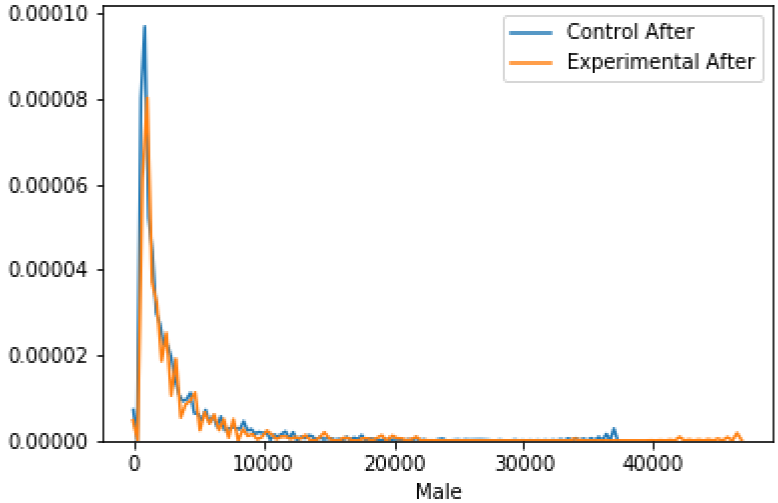
In [1041]:
```
dist_plot(control_df[(t4["gender"] == "female")],
          exp_df[(t4["gender"] == "female")],
         "total_mins","total_mins","Female", "Control After","Experiment
al After")
```



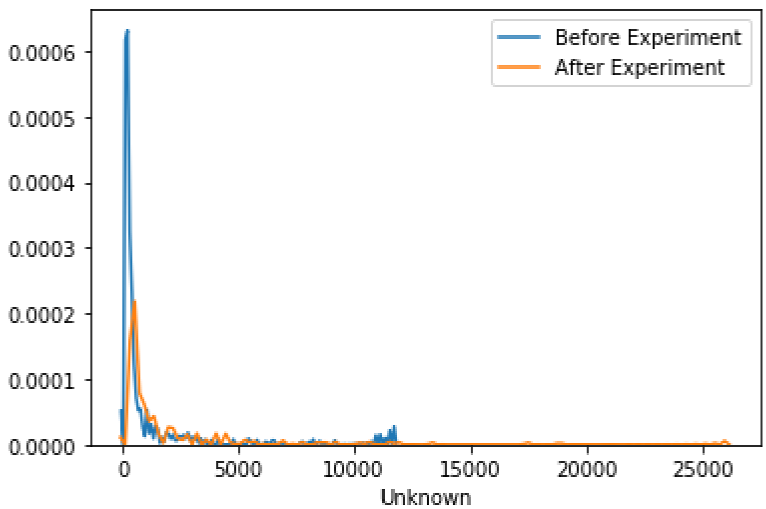
In [869]:
```
dist_plot(pre_exp_df[(t4["gender"] == "male")],
          exp_df[(t4["gender"] == "male")],
         "total_mins","total_mins","Male", "Before Experiment","After Exp
eriment")
```



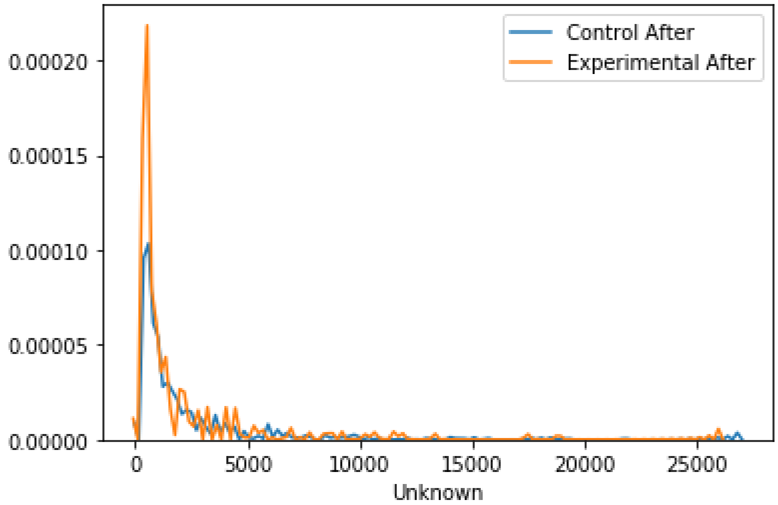
In [1040]:
```
dist_plot(control_df[(t4["gender"] == "male")],
          exp_df[(t4["gender"] == "male")],
         "total_mins","total_mins","Male", "Control After","Experimental
 After")
```



In [870]:
```
dist_plot(pre_exp_df[(t4["gender"] == "unknown")],
             exp_df[(t4["gender"] == "unknown")],
            "total_mins","total_mins","Unknown", "Before Experiment","After
Experiment")
```

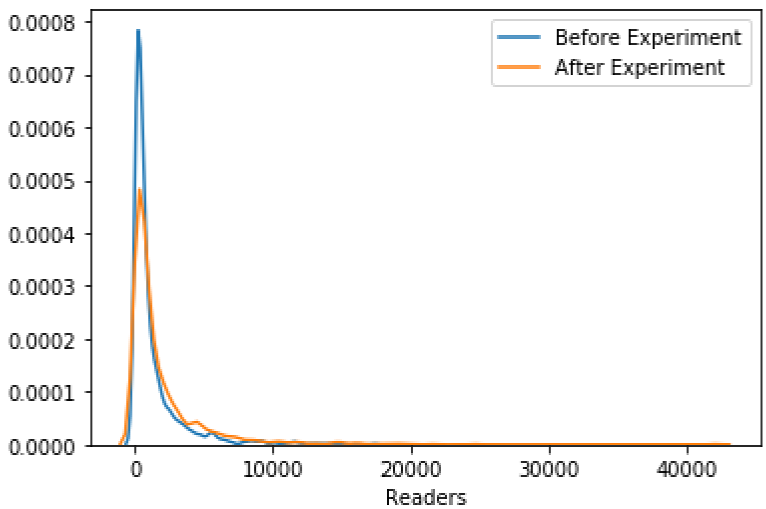


In [1047]:
```
dist_plot(control_df[(t4["gender"] == "unknown")],
             exp_df[(t4["gender"] == "unknown")],
            "total_mins","total_mins","Unknown", "Control After","Experimen
tal After")
```

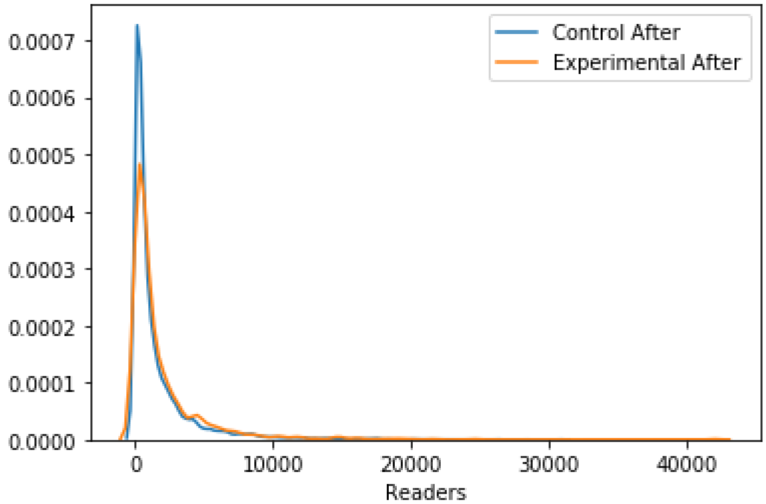


By User Type

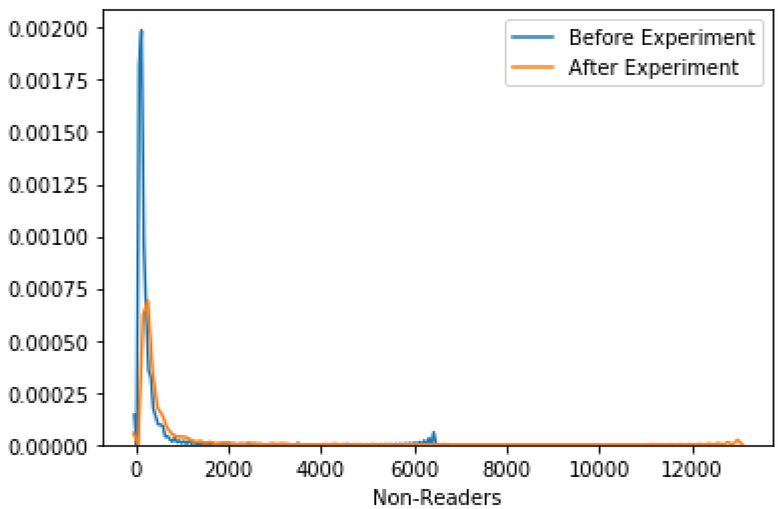
In [871]:
```
dist_plot(pre_exp_df[(t4["user_type"] == "reader")],
          exp_df[(t4["user_type"] == "reader")],
         "total_mins","total_mins","Readers", "Before Experiment","After
Experiment")
```



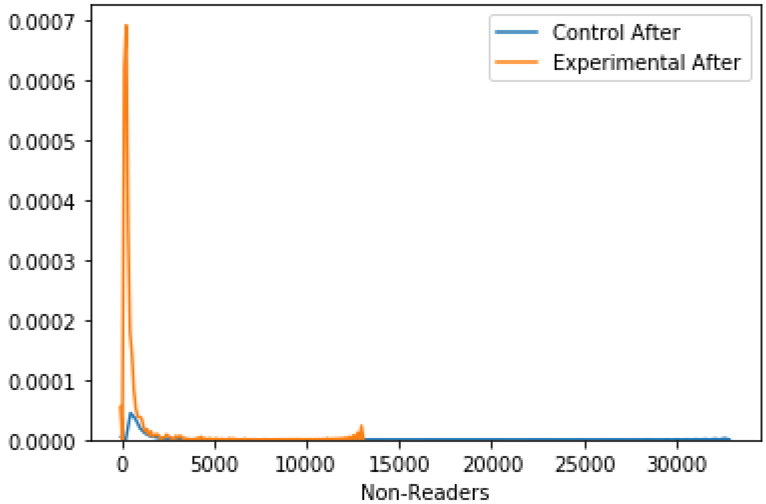
In [1043]:
```
dist_plot(control_df[(t4["user_type"] == "reader")],
          exp_df[(t4["user_type"] == "reader")],
         "total_mins","total_mins","Readers", "Control After","Experimen
tal After")
```



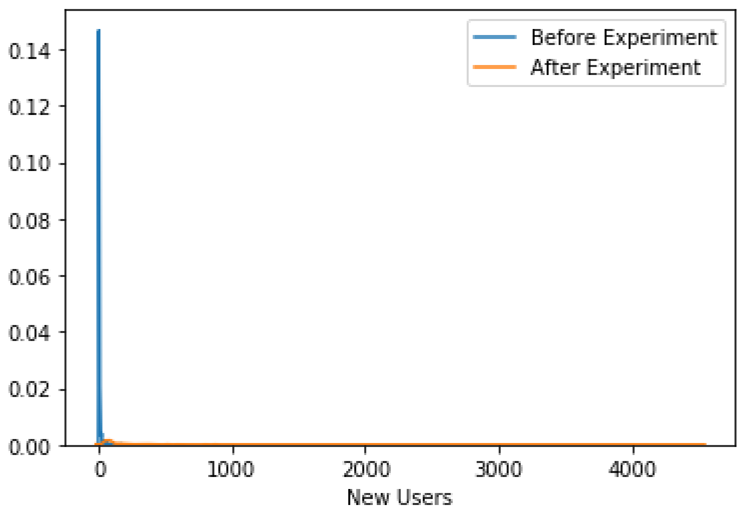
In [872]:
```
dist_plot(pre_exp_df[(t4["user_type"] == "non_reader")],
           exp_df[(t4["user_type"] == "non_reader")],
          "total_mins","total_mins","Non-Readers", "Before Experiment","Af
ter Experiment")
```



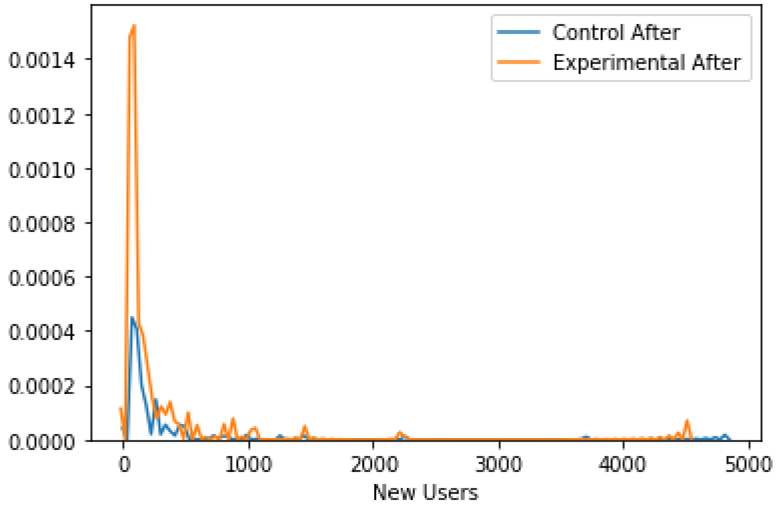
In [1044]:
```
dist_plot(control_df[(t4["user_type"] == "non_reader")],
           exp_df[(t4["user_type"] == "non_reader")],
          "total_mins","total_mins","Non-Readers", "Control After","Exper
imental After")
```



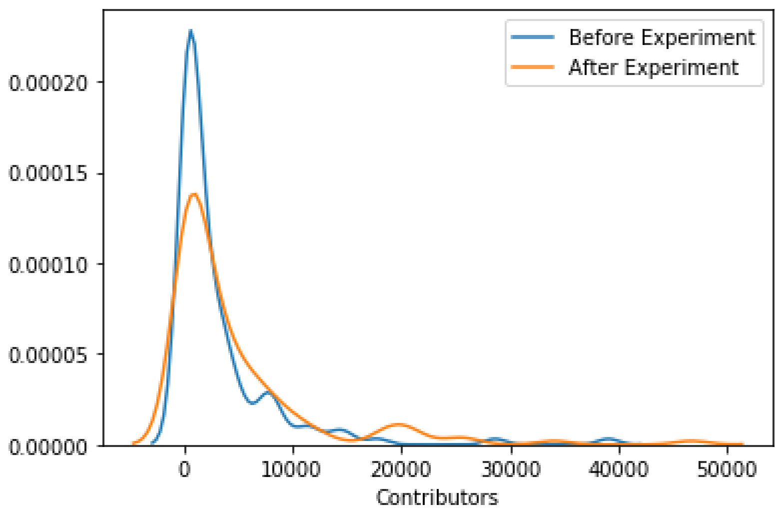
In [873]:
```
dist_plot(pre_exp_df[(t4["user_type"] == "new_user")],
           exp_df[(t4["user_type"] == "new_user")],
          "total_mins","total_mins","New Users", "Before Experiment","Afte
r Experiment")
```



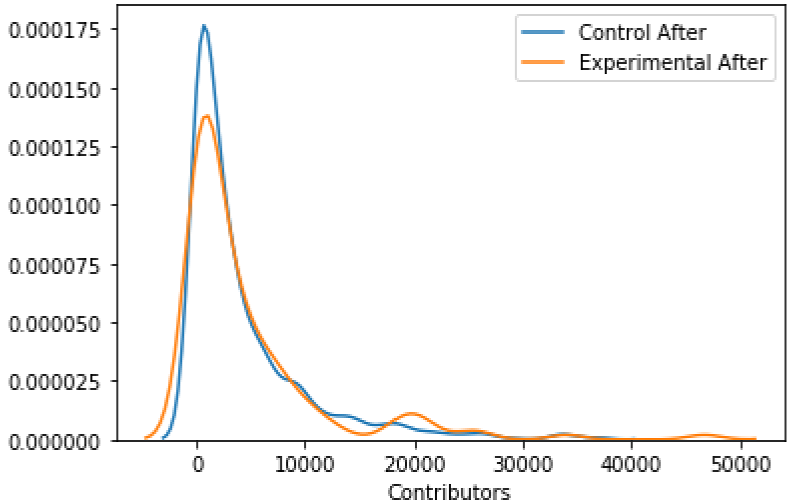
In [1045]:
```
dist_plot(control_df[(t4["user_type"] == "new_user")],
           exp_df[(t4["user_type"] == "new_user")],
          "total_mins","total_mins","New Users", "Control After","Experim
ental After")
```



In [874]:
```
dist_plot(pre_exp_df[(t4["user_type"] == "contributor")],
           exp_df[(t4["user_type"] == "contributor")],
          "total_mins","total_mins","Contributors", "Before Experiment","A
fter Experiment")
```



In [1046]:
```
dist_plot(control_df[(t4["user_type"] == "contributor")],
           exp_df[(t4["user_type"] == "contributor")],
          "total_mins","total_mins","Contributors", "Control After","Expe
rimental After")
```



Now it's time to take a look at the design of the experiment and see if anything stands out as a possible issue. Let's begin by just getting a table to breakdown our user types. I did some of this in Excel, because it was easier to set up, which is where the hard-coded numbers are coming from.

Total Users

In [1026]:

```
cols = ['Reader', 'Non-Reader', 'New User', 'Contributor', 'Total']
index = ['Female', 'Male', 'Unknown', 'Total']
ua_matrix = np.matrix('2157, 10480, 1591, 249, 14477; 4880, 19877, 2321,
679, 27757; 965, 5709, 976, 116, 7766;'
                     + '8002, 36066, 4888, 1044, 50000')
ua_df = pd.DataFrame(ua_matrix,index=index, columns=cols)
ua_df.style
```

Out[1026]:

	Reader	Non-Reader	New User	Contributor	Total
Female	2157	10480	1591	249	14477
Male	4880	19877	2321	679	27757
Unknown	965	5709	976	116	7766
Total	8002	36066	4888	1044	50000

Now we have our table and we can see our user's breakdowns, if we want to make it prettier and even throw in a little heatmap we can do so as well. I'm removing the "Total" row as it would mess up the gradient.

In [1027]:

```
cols2 = ['Reader', 'Non-Reader', 'New User', 'Contributor']
index2 = ['Female', 'Male', 'Unknown']
ua_matrix2 = np.matrix('2157, 10480, 1591, 249; 4880, 19877, 2321, 679; 9
65, 5709, 976, 116')
ua_df2 = pd.DataFrame(ua_matrix2,index=index2, columns=cols2)
ua_df2.style.background_gradient(cmap='Blues')
```

Out[1027]:

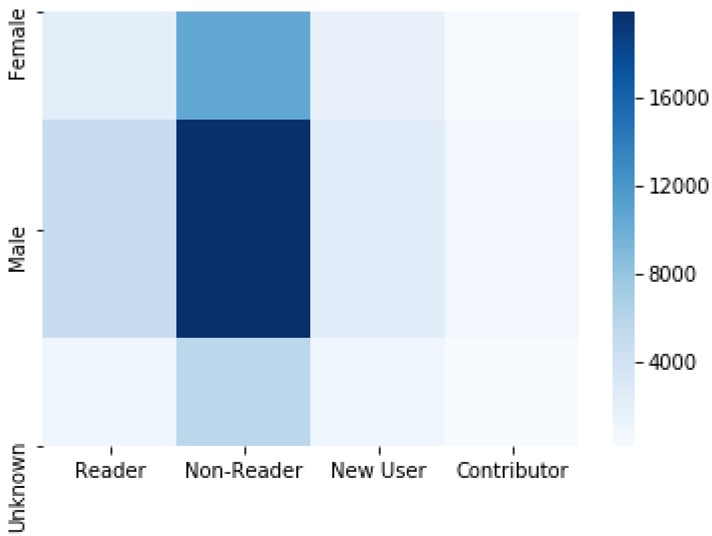
	Reader	Non-Reader	New User	Contributor

Female	2157	10480	1591	249
Male	4880	19877	2321	679
Unknown	965	5709	976	116

Unfortunately, the current version of matplotlib broke heatmaps and it's not really worth downgrading for so the y-axis labels are a little skewed but it serves its purpose.

In [1028]:
```
sns.heatmap(ua_df2, cmap="Blues")
```

Out[1028]: <matplotlib.axes._subplots.AxesSubplot at 0x1aadca6510>



Control Users

In [1029]:
```
# Probably should have made a function at this point and I would if I were going to use this again but I'm already
# pretty much done
cols = ['Reader', 'Non-Reader', 'New User', 'Contributor', 'Total']
index = ['Female', 'Male', 'Unknown', 'Total']
ua_control_matrix = np.matrix('1821,8387,1176,223,11607; 4126,15768,1747,596,22237; '
                               + '786,4544,730,96,6156; 6733,28699,3653,915,40000')
ua_control_df = pd.DataFrame(ua_control_matrix,index=index, columns=cols)
ua_control_df.style
```

Out[1029]:

	Reader	Non-Reader	New User	Contributor	Total
Female	1821	8387	1176	223	11607
Male	4126	15768	1747	596	22237
Unknown	786	4544	730	96	6156
Total	6733	28699	3653	915	40000

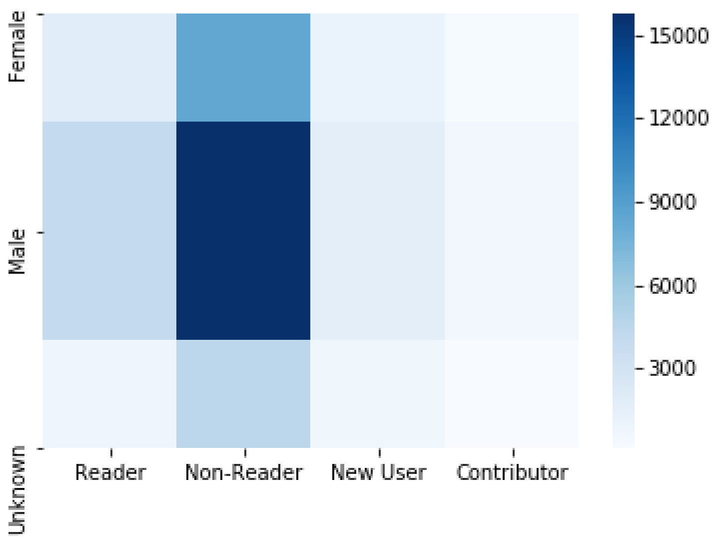
In [1030]:
```
cols2 = ['Reader', 'Non-Reader', 'New User', 'Contributor']
index2 = ['Female', 'Male', 'Unknown']
ua_control_matrix2 = np.matrix('1821,8387,1176,223; 4126,15768,1747,596;
'
                               + '786,4544,730,96')
ua_control_df2 = pd.DataFrame(ua_control_matrix2,index=index2, columns=co
ls2)
ua_control_df2.style.background_gradient(cmap='Blues')
```

Out[1030]:

	Reader	Non-Reader	New User	Contributor
Female	1821	8387	1176	223
Male	4126	15768	1747	596
Unknown	786	4544	730	96

In [1031]:
```
sns.heatmap(ua_control_df2, cmap="Blues")
```

Out[1031]: <matplotlib.axes._subplots.AxesSubplot at 0x1aadfd8890>



Experimental Users

In [1032]:
```
cols = ['Reader', 'Non-Reader', 'New User', 'Contributor', 'Total']
index = ['Female', 'Male', 'Unknown', 'Total']
ua_experimental_matrix = np.matrix('336,2093,415,26,2870; 754,4109,574,83
,5520; '
                                   + '179,1165,246,20,1610; 1269,7367,1235
,129,10000')
ua_experimental_df = pd.DataFrame(ua_experimental_matrix,index=index, col
umns=cols)
ua_experimental_df.style
```

Out[1032]:

	Reader	Non-Reader	New User	Contributor	Total
Female	336	2093	415	26	2870

Male	754	4109	574	83	5520
Unknown	179	1165	246	20	1610
Total	1269	7367	1235	129	10000

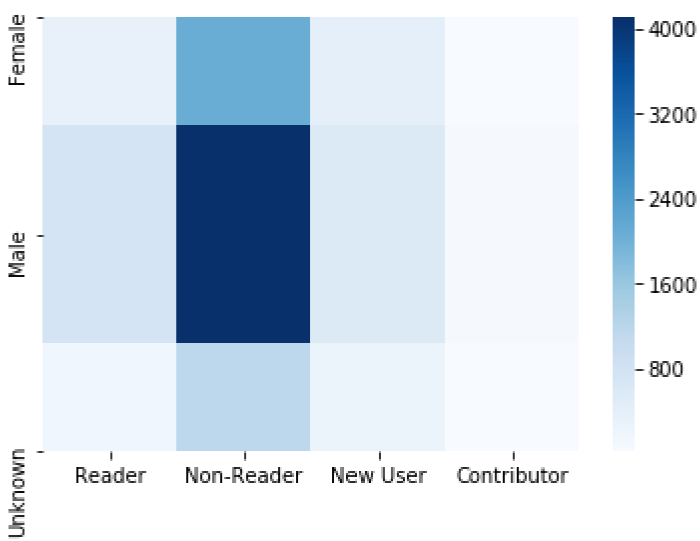
In [1033]:
```
cols = ['Reader', 'Non-Reader', 'New User', 'Contributor']
index = ['Female', 'Male', 'Unknown']
ua_experimental_matrix2 = np.matrix('336,2093,415,26; 754,4109,574,83; '
                                    + '179,1165,246,20')
ua_experimental_df2 = pd.DataFrame(ua_experimental_matrix2,index=index, c
olumns=cols)
ua_experimental_df2.style.background_gradient(cmap='Blues')
```

Out[1033]:

	Reader	Non-Reader	New User	Contributor
Female	336	2093	415	26
Male	754	4109	574	83
Unknown	179	1165	246	20

In [1034]:
```
sns.heatmap(ua_experimental_df2, cmap="Blues")
```

Out[1034]: <matplotlib.axes._subplots.AxesSubplot at 0x1aae105cd0>



It looks like the two groups were broken down in a way that is representative of the sample (and hopefully population but there were some over/undersamplings that I discussed more in my answer to this question).