In [1]:

```
import pandas as pd
import numpy as np
from ggplot import *
```

In [2]:

```
# read input data - version 2
df2 = pd.read_csv("turnstile_weather_v2.csv", parse_dates=['datetim
e'])
# the original data
df = pd.read_csv('turnstile_data_master_with_weather.csv')
df2.head()
```

Out[2]:

	UNIT	DATEn	TIMEn	ENTRIESn	EXITSn	ENTRIESn_hourly	EXITSn_hourly	datet
0	R003	05-01- 11	00:00:00	4388333	2911002	0	0	2011 01 00:0
1	R003	05-01- 11	04:00:00	4388333	2911002	0	0	2011 01 04:0
2	R003	05-01- 11	12:00:00	4388333	2911002	0	0	2011 01 12:0
3	R003	05-01- 11	16:00:00	4388333	2911002	0	0	2011 01 16:0
4	R003	05-01- 11	20:00:00	4388333	2911002	0	0	2011 01 20:0

5 rows × 27 columns

In [3]:

```
df2.describe()
```

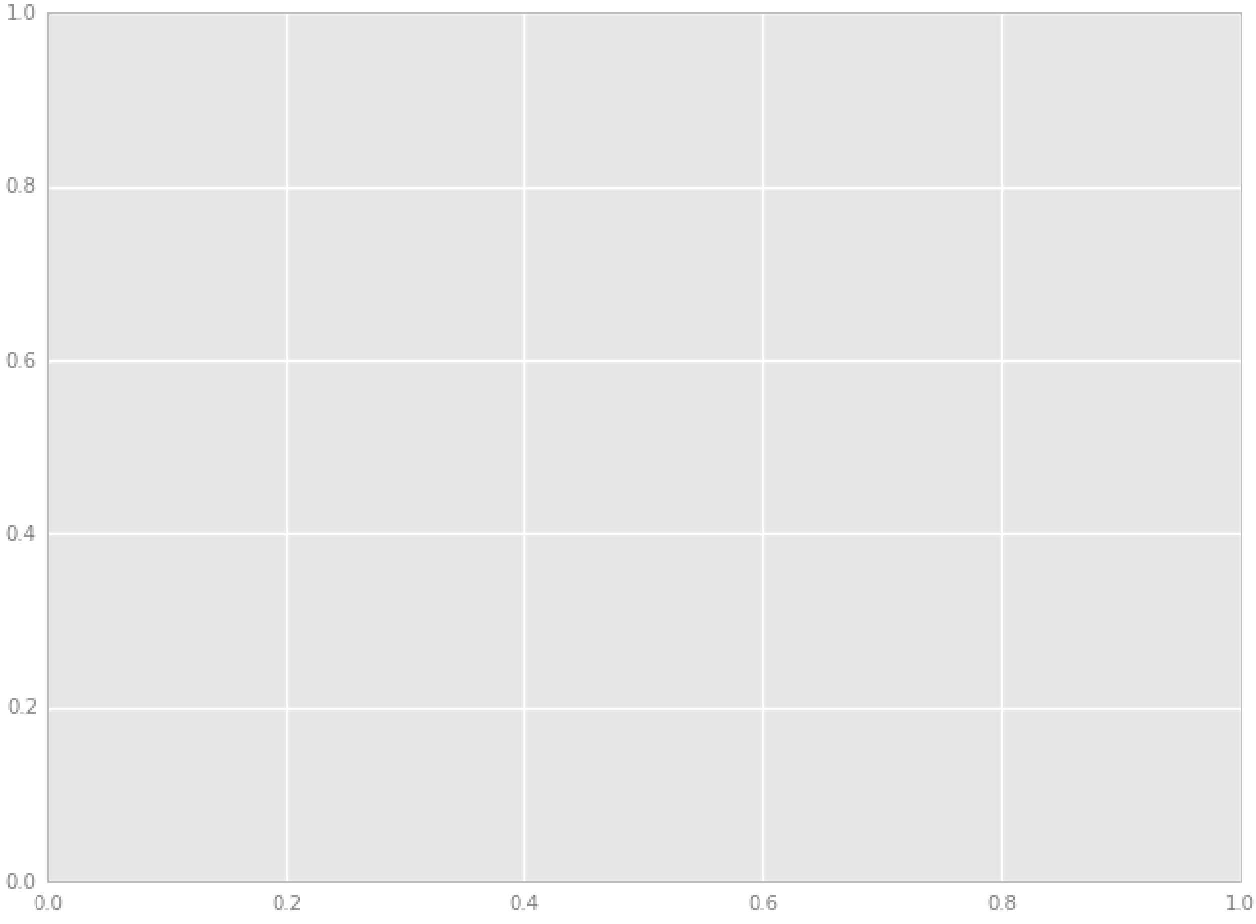
Out[3]:

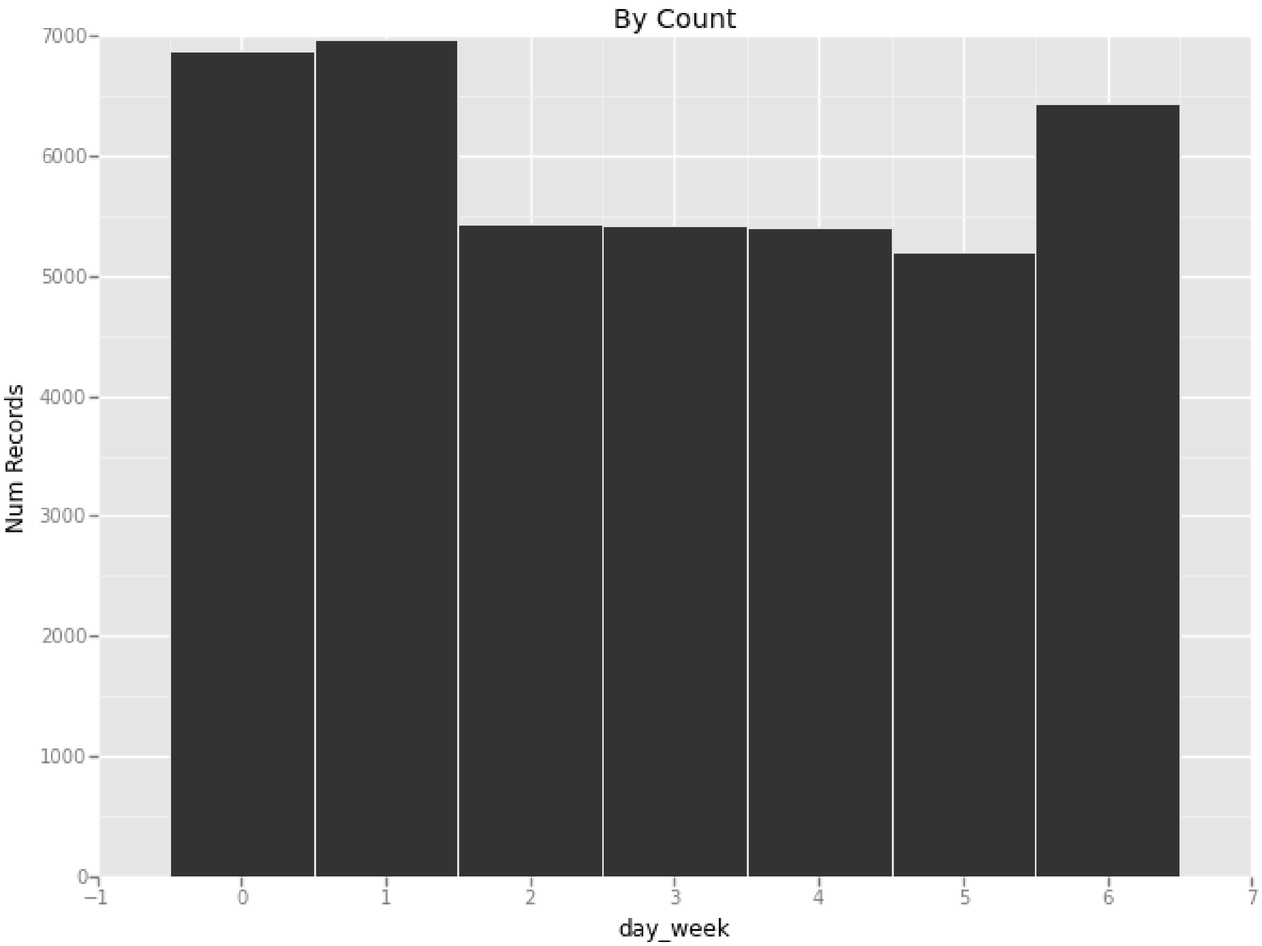
	ENTRIESn	EXITSn	ENTRIESn_hourly	EXITSn_hourly	hour	d
count	4.264900e+04	4.264900e+04	42649.000000	42649.000000	42649.000000	4
mean	2.812486e+07	1.986993e+07	1886.589955	1361.487866	10.046754	2.
std	3.043607e+07	2.028986e+07	2952.385585	2183.845409	6.938928	2.
min	0.000000e+00	0.000000e+00	0.000000	0.000000	0.000000	0.
25%	1.039762e+07	7.613712e+06	274.000000	237.000000	4.000000	1.
50%	1.818389e+07	1.331609e+07	905.000000	664.000000	12.000000	3.
75%	3.263049e+07	2.393771e+07	2255.000000	1537.000000	16.000000	5.
max	2.357746e+08	1.493782e+08	32814.000000	34828.000000	20.000000	6.

8 rows × 21 columns

In [6]:

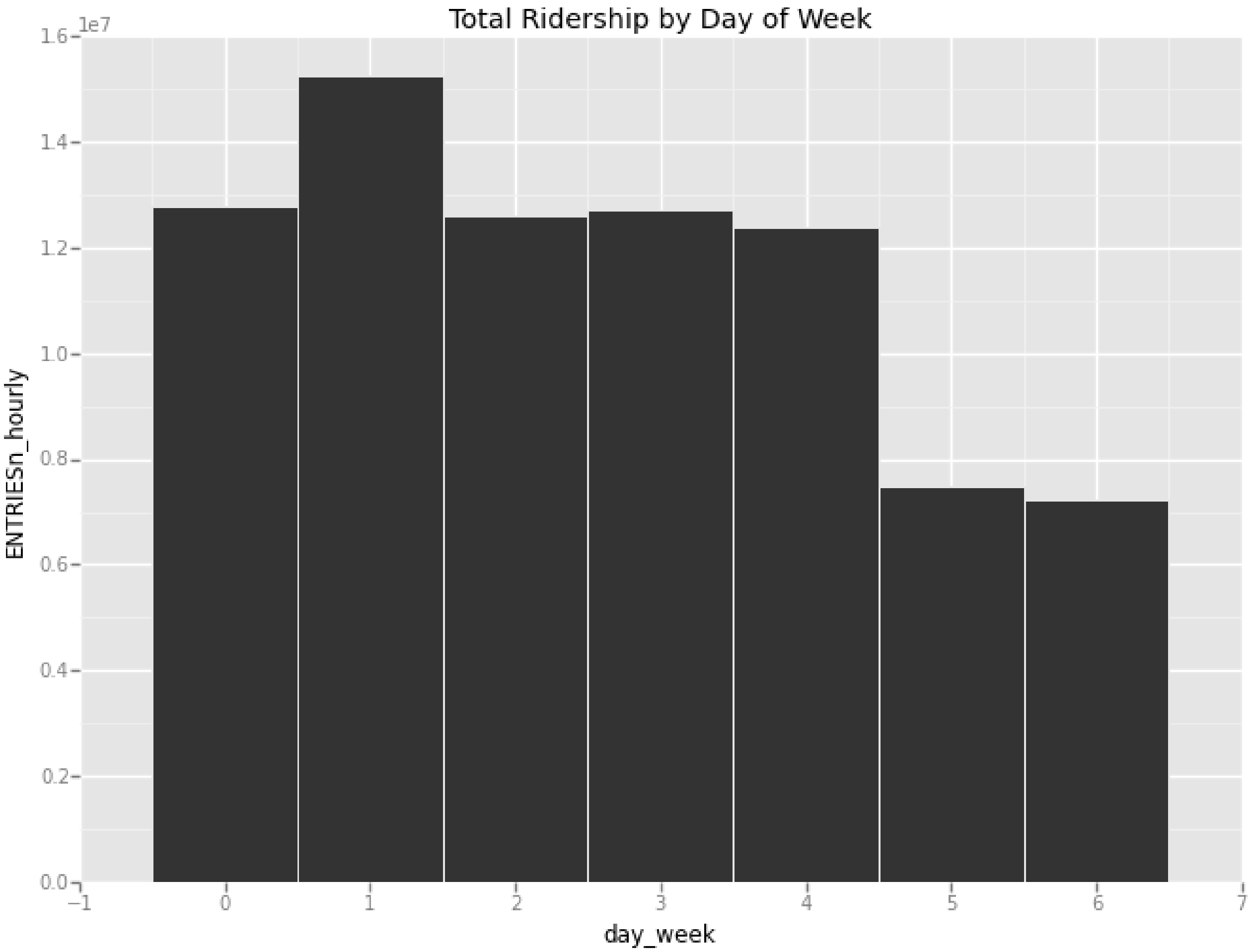
```
# distribution of data by day of week
#ggplot(aes(x='day_week'), data=df2) + geom_histogram(binwidth=1) # note this is lumping last bin in with second to last
day_count = df2[['day_week', 'ENTRIESn_hourly']].groupby('day_week', as_index=False).aggregate(np.count_nonzero)
ggplot(aes(x='day_week', y='ENTRIESn_hourly'), data=day_count) + geom_bar(stat='identity') + labs(title='By Count') + ylab('Num Records')
```





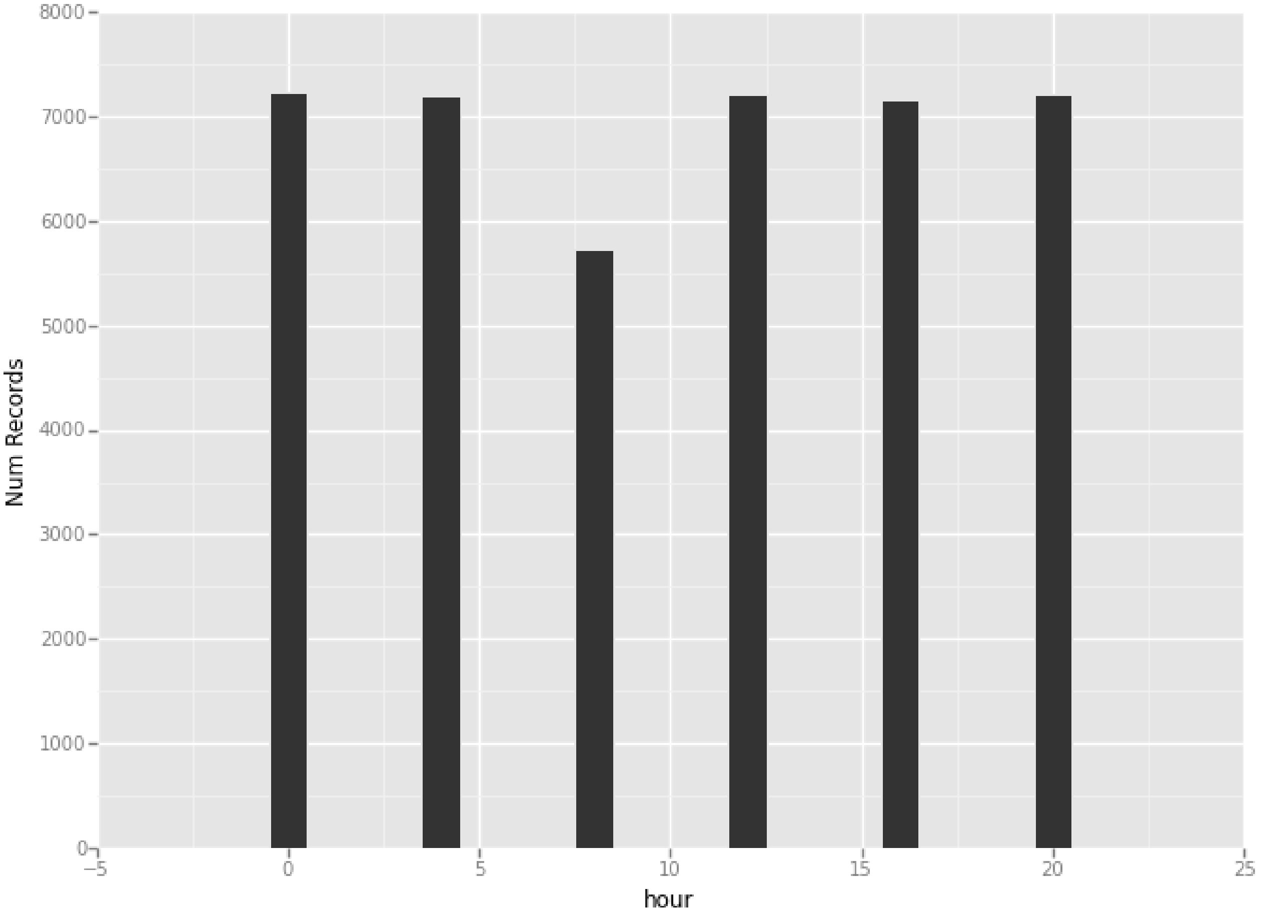
```
Out[6]: <ggplot: (8738211923377)>
```

```
In [7]: day_sum = df2[['day_week', 'ENTRIESn_hourly']].groupby('day_week', a
        s_index=False).aggregate(np.sum)
        ggplot(aes(x='day_week', y='ENTRIESn_hourly'), data=day_sum) + geom_ba
        r(stat='identity') + labs(title='Total Ridership by Day of Week')
```


Total Ridership by Day of Week
1e7
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
ENTRIESn_hourly
−1
0
1
2
3
4
5
6
7
day_week

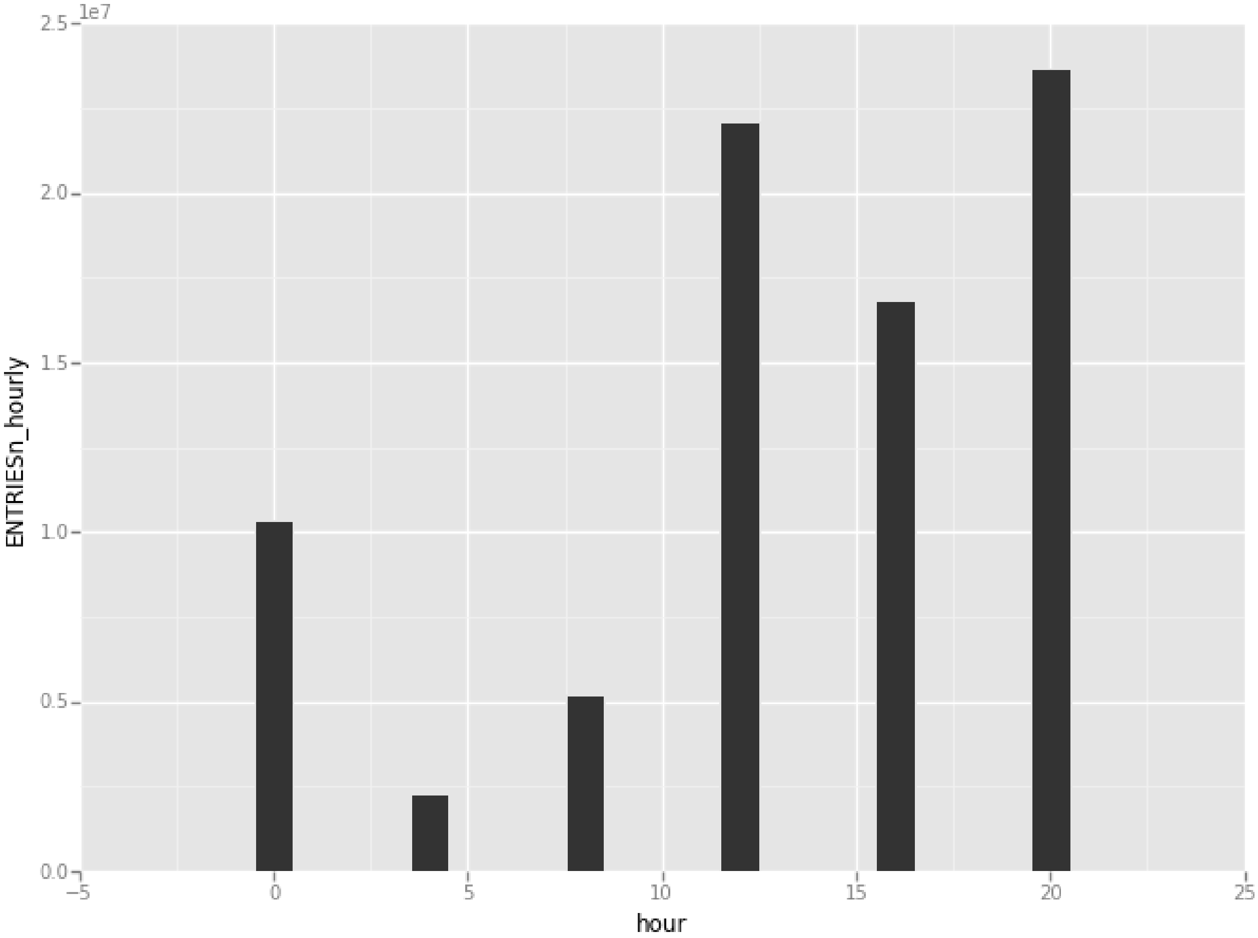
```
Out[7]: <ggplot: (8738211773285)>
```

```
In [25]: # distribution of data by hour
hour_count = df2[['hour', 'ENTRIESn_hourly']].groupby('hour', as_inde
x=False).aggregate(np.count_nonzero)
#ggplot(aes(x='hour'), data=df2) + geom_histogram(binwidth=1) # last b
in is being included with next to last bin
ggplot(aes(x='hour', y='ENTRIESn_hourly'), data=hour_count) + geom_ba
r(stat='identity') + ylab('Num Records')
```



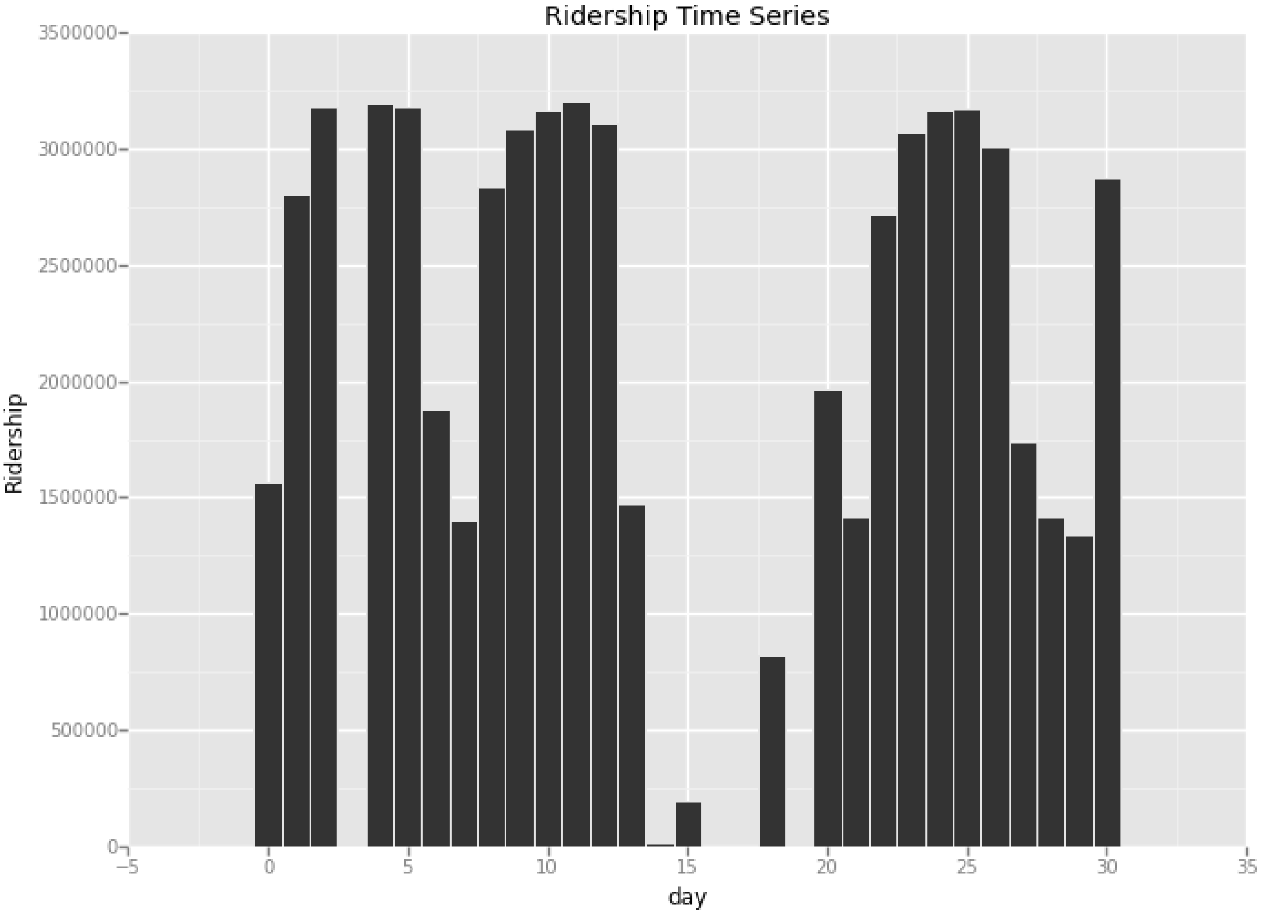
```
Out[25]: <ggplot: (8786094427405)>
```

```
In [8]: hour_sum = df2[['hour', 'ENTRIESn_hourly']].groupby('hour', as_index=F
        alse).aggregate(np.sum)
        ggplot(aes(x='hour', y='ENTRIESn_hourly'), data=hour_sum) + geom_bar(s
        tat='identity')
```



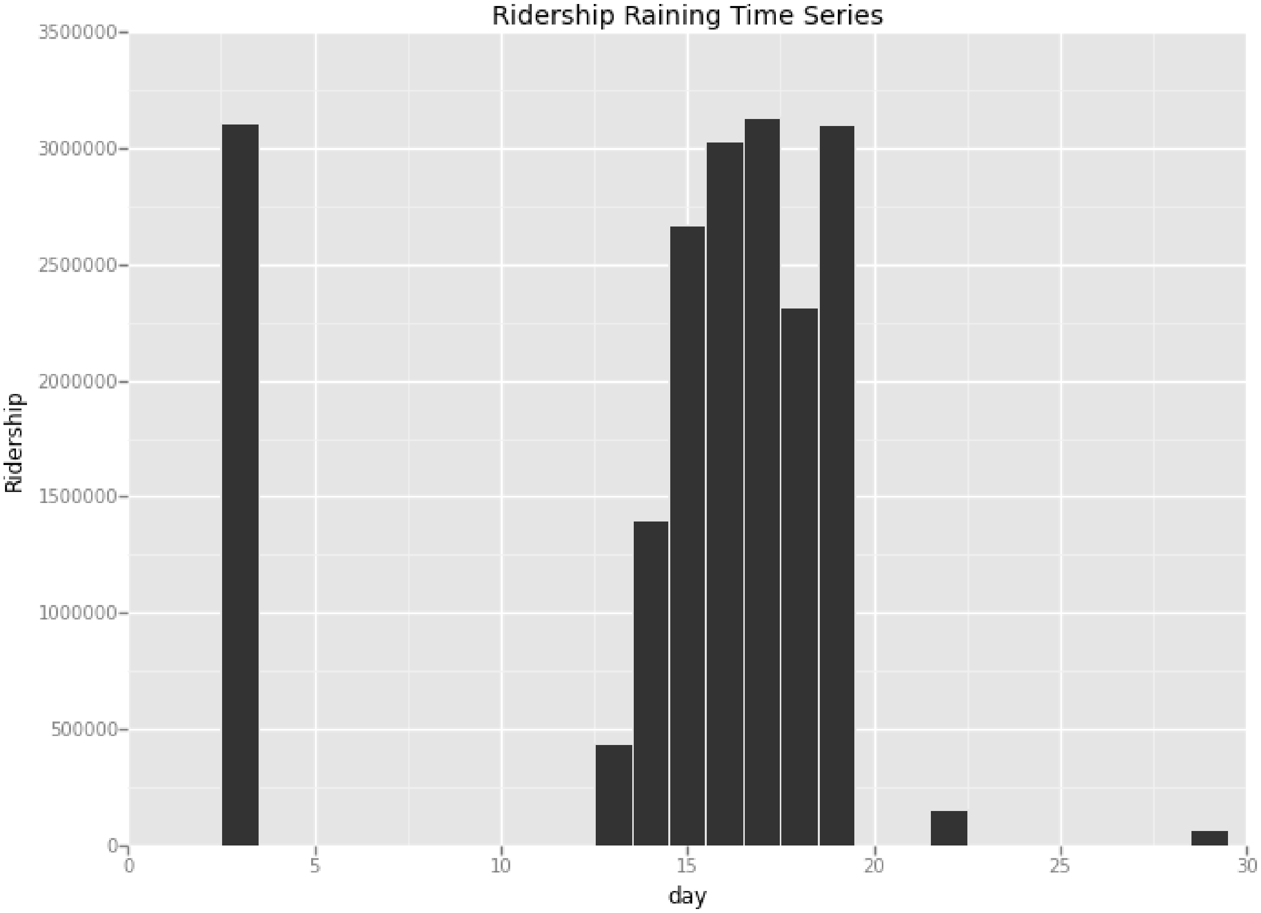
```
Out[8]: <ggplot: (8736946058553)>
```

```
In [19]: # let's plot as time series
         # plotting the time series allows to spot gaps and outliers too
         from datetime import datetime
         norain = df2[df2['rain'] == 0]
         epoch = datetime(2011, 5, 1)
         # add seconds since some known time as column
         norain['day'] = norain.apply(lambda x: (x['datetime'] - epoch).days, a
         xis=1)
         norain = norain.set_index(['day'], drop=False)
         time_series = norain[['day', 'ENTRIESn_hourly']].groupby('day', as_ind
         ex=False).aggregate(np.sum)
         ggplot(aes(x='day', y='ENTRIESn_hourly'), data=time_series) + geom_ba
         r(stat="identity") + ylab('Ridership') + labs(title='Ridership Time Se
         ries')
```



```
Out[19]: <ggplot: (8736945200025)>
```

In [27]:
```
rain = df2[df2['rain'] == 1]
rain['day'] = rain.apply(lambda x: (x['datetime'] - epoch).days, axi
s=1)
rain = rain.set_index(['day'], drop=False)
time_series_rain = rain[['day', 'ENTRIESn_hourly']].groupby('day', a
s_index=False).aggregate(np.sum)
ggplot(aes(x='day', y='ENTRIESn_hourly'), data=time_series_rain) + geo
m_bar(stat="identity") + ylab('Ridership') + labs(title='Ridership Rai
ning Time Series')
```

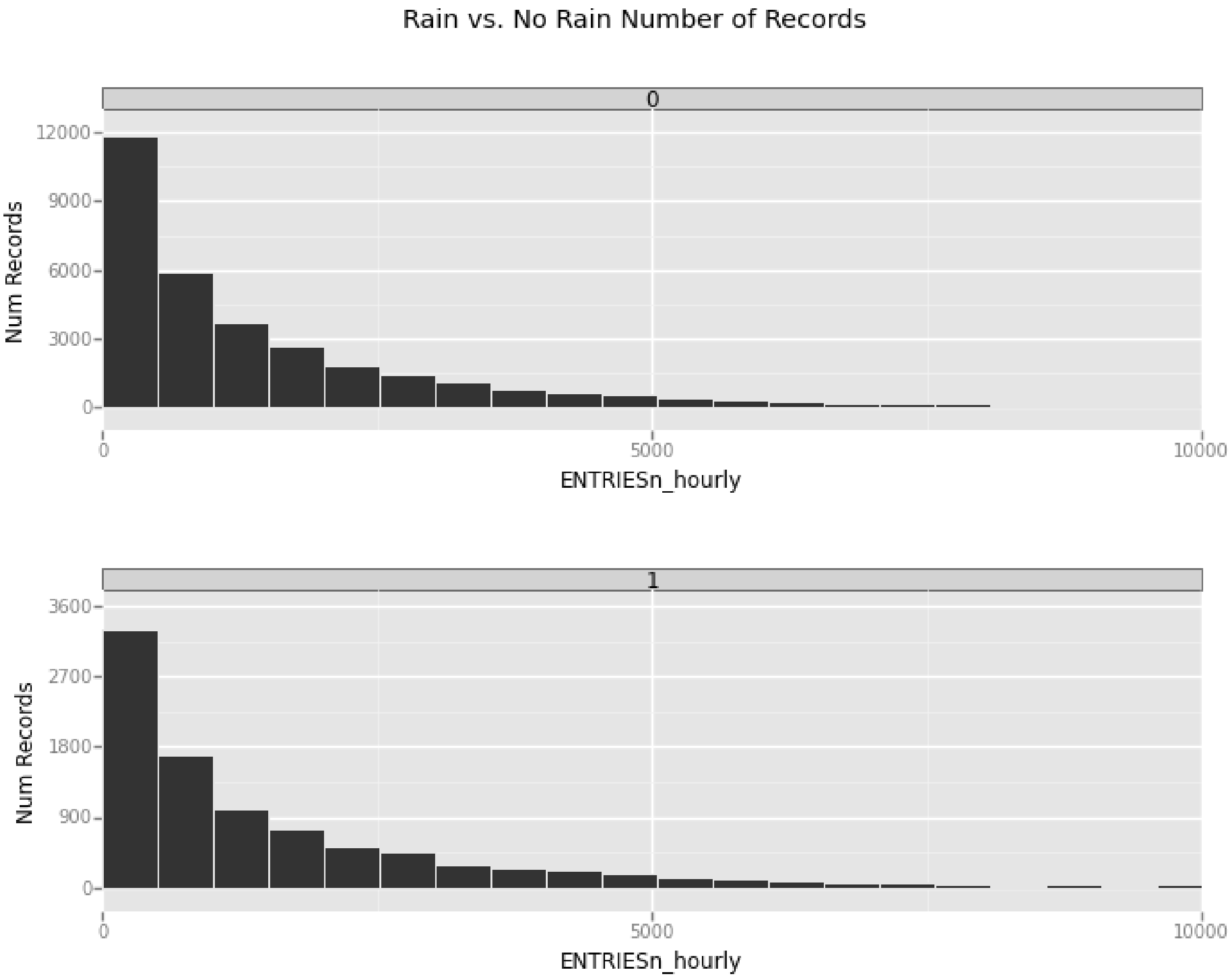


```
Out[27]: <ggplot: (8736944950769)>
```

In [28]:
```
ggplot(aes(x='ENTRIESn_hourly'), data=df2) + geom_histogram(binwidth=5
00) + scale_x_continuous(limits=(0, 10000)) +\
facet_wrap('rain') + ylab('Num Records') + labs(title='Rain vs. No Rai
n Number of Records')
```

```
/home/jay/anaconda/lib/python2.7/site-packages/ggplot/ggplot.py:200: R
untimeWarning: Facetting is currently not supported with geom_bar. See
                    https://github.com/yhat/ggplot/issues/196 for more
information
  warnings.warn(msg, RuntimeWarning)
/home/jay/anaconda/lib/python2.7/site-packages/pandas/util/decorator
s.py:81: FutureWarning: the 'rows' keyword is deprecated, use 'index'
instead
  warnings.warn(msg, FutureWarning)
/home/jay/anaconda/lib/python2.7/site-packages/ggplot/geoms/geom_bar.p
y:47: FutureWarning: comparison to `None` will result in an elementwis
e object comparison in the future.

  _reset = self.bottom == None or (self.ax != None and self.ax != ax)
```



```
Out[28]: <ggplot: (8736944872637)>
```

In [10]:

```
sum_rain = df2[['rain', 'ENTRIESn_hourly']].groupby('rain', as_index=F
alse).aggregate(np.sum)
#ggplot(aes(x='ENTRIESn_hourly'), data=sum_rain) + geom_histogram(binw
idth=500) + scale_x_continuous(limits=(0, 10000)) +\
#facet_wrap('rain') + ylab('Ridership') + labs(title='Rain vs. No Rain
Ridership')
sum_rain
```

Out[10]:

	rain	ENTRIESn_hourly
0	0	61020916
1	1	19440259

In [41]:

```
rain = df2[df2['rain'] == 1]['ENTRIESn_hourly']
rain.describe()
```

Out[41]:

```
count     9585.000000
mean      2028.196035
std       3189.433373
min          0.000000
25%        295.000000
50%        939.000000
75%       2424.000000
max      32289.000000
Name: ENTRIESn_hourly, dtype: float64
```

In [42]:

```
norain = df2[df2['rain'] == 0]['ENTRIESn_hourly']
norain.describe()
```

Out[42]:

```
count    33064.000000
mean      1845.539439
std       2878.770848
min          0.000000
25%        269.000000
50%        893.000000
75%       2197.000000
max      32814.000000
Name: ENTRIESn_hourly, dtype: float64
```

In [43]:

```
print rain.mean() - norain.mean()
print rain.median() - norain.median()
```

```
182.656596808
46.0
```

In [ ]: