# Analyzing the NYC Subway Dataset

Questions

## Overview

This project consists of two parts. In Part 1 of the project, you should have completed the questions in Problem Sets 2, 3, and 4 in the Introduction to Data Science course. This document addresses part 2 of the project. Please use this document as a template and answer the following questions to explain your reasoning and conclusion behind your work in the problem sets. You will attach a document with your answers to these questions as part of your final project submission.

### Section 0. References

https://en.wikipedia.org/wiki/Nonparametric_statistics (https://en.wikipedia.org/wiki/Nonparametric_statistics) https://en.wikipedia.org/wiki/Mann–Whitney_U_test (https://en.wikipedia.org/wiki/Mann–Whitney_U_test) http://dss.princeton.edu/online_help/analysis/interpreting_regression.htm (http://dss.princeton.edu/online_help/analysis/interpreting_regression.htm) https://en.wikipedia.org/wiki/Polynomial_regression (https://en.wikipedia.org/wiki/Polynomial_regression)

### Section 1. Statistical Test

***1.1 Which statistical test did you use to analyze the NYC subway data? Did you use a one-tail or a two-tail P value? What is the null hypothesis? What is your p-critical value?***

***ANS :***

```
We choose Mann—Whitney U test two-tail P value.

H0 is "the distribution of Hourly_entries for the two groups are equal"

We define α = 0.05
```

***1.2 Why is this statistical test applicable to the dataset? In particular, consider the assumptions that the test is making about the distribution of ridership in the two samples.***

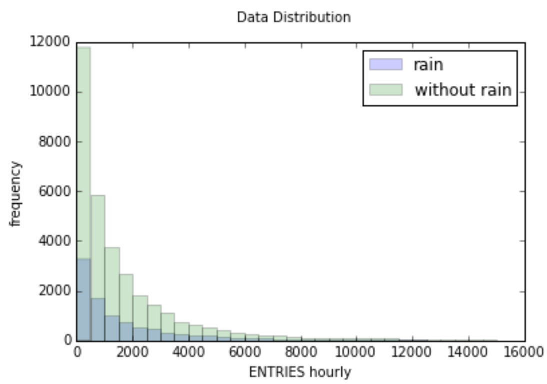
***ANS :***

In [2]:

```
import pandas as pd
import os
import scipy
import scipy.stats
import numpy as np
import statsmodels.api as sm
alpha = 0.05
filename = r'~/WorkSpace/Udemy-Datascience/IntroductionToDataScience/P2-Analyzing-NYC/turnstile_weather_v2.csv'
turnstile_weather = pd.read_csv(filename)

H0 = "\"the distribution of ENTRIESn_hourly for the two groups are equal\""
not_H0 = "\"the distribution of ENTRIESn_hourly for the two groups are not equal\""
```

In [3]:
```
%pylab inline
import matplotlib.pyplot as plt

rain_df = turnstile_weather[turnstile_weather.rain == 1]['ENTRIESn_hourly']
norain_df = turnstile_weather[turnstile_weather.rain == 0]['ENTRIESn_hourly']
df_compare = pd.DataFrame({'rain': rain_df,
               'without rain': norain_df})
df_compare.plot(kind='hist', alpha=0.2,bins = range(0, 15000 + 500, 500) ) ##,bins = range(0, 37 + 3, 3)

# plt.set_title('Data Distribution')
plt.suptitle('Data Distribution')
plt.xlabel("ENTRIES hourly")
plt.ylabel("frequency")
plt.show()
print "As you can see, the distribution is not normal."
```

```
Populating the interactive namespace from numpy and matplotlib
```




```
As you can see, the distribution is not normal.
```

In [4]:
```
print "The number of rain data is %s ." % len(rain_df)
print "The number of without rain data is %s ." % len(norain_df)
```

```
The number of rain data is 9585 .
The number of without rain data is 33064 .
```

In [6]:
```
import scipy
w,p = scipy.stats.shapiro(norain_df)
print "The p-value for the hypothesis that the without rain data was drawn from a normal distribution is %.10f ." % p

w,p = scipy.stats.shapiro(rain_df)
print "The p-value for the hypothesis that the with rain data was drawn from a normal distribution is %.10f ." % p
```

```
The p-value for the hypothesis that the without rain data was drawn from a normal distribution is 0.0000000000 .
The p-value for the hypothesis that the with rain data was drawn from a normal distribution is 0.0000000000 .
```

### Conclusion :

Part of the assumption of parametric tests like the T-test is that the distributions are normal.

As you can see the distribution is not normal so we use the Mann-Whitney over the T-test.

Ref: http://vassarstats.net/textbook/parametric.html (http://vassarstats.net/textbook/parametric.html) Ref: https://en.wikipedia.org/wiki/Mann%E2%80%93Whitney_U_test (https://en.wikipedia.org/wiki/Mann%E2%80%93Whitney_U_test) Ref: https://en.wikipedia.org/wiki/Shapiro%E2%80%93Wilk_test (https://en.wikipedia.org/wiki/Shapiro%E2%80%93Wilk_test)

***1.3 What results did you get from this statistical test? These should include the following numerical values: p-values, as well as the means for each of the two samples under test.***

**ANS:**

In [7]:
```
def test_has_effect(df1,df2 , name_df1, name_df2 ,show_graph= False):
    U,p = scipy.stats.mannwhitneyu(df1,df2)
    df1_mean = np.mean(df1)
    df2_mean = np.mean(df2)
    #agg = turnstile_weather.groupby(['is_precipi'], as_index=False).mean()
    if show_graph:
        df_compare = pd.DataFrame({name_df1: df1,
                    name_df2: df2})
        df_compare.plot(kind='hist', alpha=0.2,bins = range(0, 15000 + 500, 500) ) ##,bins = range(0, 37 + 3, 3)

        # plt.set_title('Data Distribution')
        plt.suptitle('Data Distribution')
        plt.xlabel("ENTRIES hourly")
        plt.ylabel("frequency")
        plt.show()
        print "As you can see, the distribution is not normal."
    p = 2*p
    print " After we perform mannwhitney t-test the conclusion is:"
    print " p-values for two-tailed is %.9f."%p
    if p < alpha :
        print " We reject the null hypothesis. \n %s" % (not_H0)
    else :
        print " We accept the null hypothesis. \n %s" % (H0)
    print " %s mean is %.10f"% (name_df1,df1_mean )
    print " %s mean is %.10f"% (name_df2,df2_mean )

df1 = turnstile_weather[turnstile_weather.rain == 1]['ENTRIESn_hourly']
df2 = turnstile_weather[turnstile_weather.rain == 0]['ENTRIESn_hourly']
test_has_effect(df1,df2 , 'with rain', 'without rain' )
```

```
 After we perform mannwhitney t-test the conclusion is:
 p-values for two-tailed is 0.000005482.
 We reject the null hypothesis.
 "the distribution of ENTRIESn_hourly for the two groups are not equal"
 with rain mean is 2028.1960354721
 without rain mean is 1845.5394386644
```

***1.4 What is the significance and interpretation of these results?***

**ANS:**

The distribution of "ENTRIESn_hourly" of 2 samples are statistically significance different.

## Section 2. Linear Regression

***2.1 What approach did you use to compute the coefficients theta and produce prediction for ENTRIESn_hourly in your regression model:***

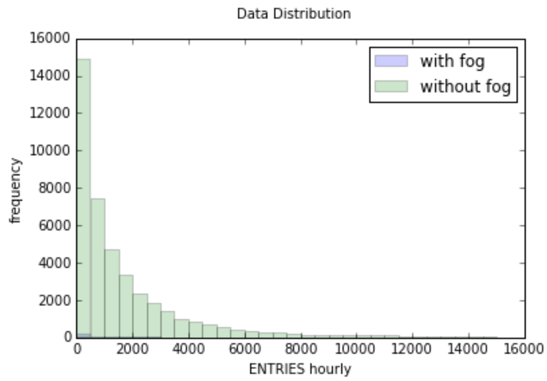
OLS using Statsmodels or Scikit Learn Gradient descent using Scikit Learn Or something different?

ANS: We decide to go for OLS.

***2.2 What features (input variables) did you use in your model? Did you use any dummy variables as part of your features?***

***ANS :***

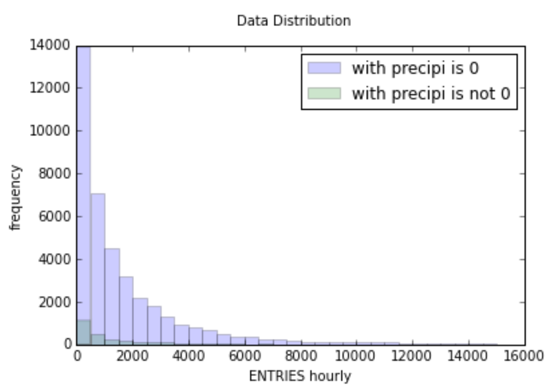
In [8]:
```
df1 = turnstile_weather[turnstile_weather.fog == 1]['ENTRIESn_hourly']
df2 = turnstile_weather[turnstile_weather.fog == 0]['ENTRIESn_hourly']
test_has_effect(df1,df2 , 'with fog', 'without fog',show_graph=True )
```




```
As you can see, the distribution is not normal.
 After we perform mannwhitney t-test the conclusion is:
 p-values for two-tailed is 0.006688382.
 We reject the null hypothesis.
 "the distribution of ENTRIESn_hourly for the two groups are not equal"
 with fog mean is 1631.9809069212
 without fog mean is 1889.1161496566
```

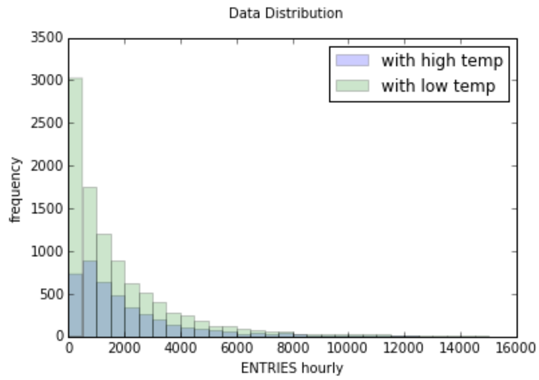
In [9]:
```
resolution = 0.2
turnstile_weather['is_precipi'] =  turnstile_weather['precipi'].apply(lambda x: 0 if x == 0 else 1 )


df1 = turnstile_weather[turnstile_weather.is_precipi == 0]['ENTRIESn_hourly']
df2 = turnstile_weather[turnstile_weather.is_precipi == 1]['ENTRIESn_hourly']
print "is_precipi == 0 data size is %s "%len(df1)
print "is_precipi == 1 data size is %s "%len(df2)
test_has_effect(df1,df2 , 'with precipi is 0', 'with precipi is not 0' ,show_graph=True)
```

```
is_precipi == 0 data size is 39827
is_precipi == 1 data size is 2822
```



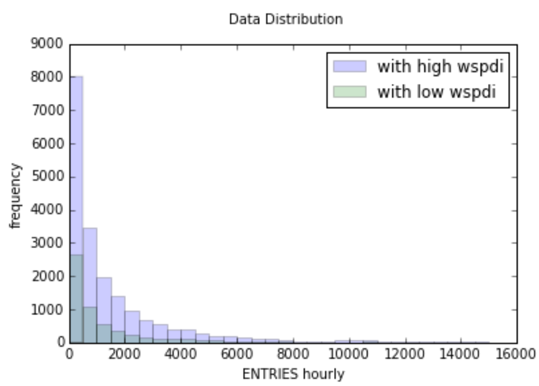

```
As you can see, the distribution is not normal.
 After we perform mannwhitney t-test the conclusion is:
 p-values for two-tailed is 0.000191212.
 We reject the null hypothesis.
 "the distribution of ENTRIESn_hourly for the two groups are not equal"
 with precipi is 0 mean is 1896.7423104929
 with precipi is not 0 mean is 1743.3093550673
```

In [10]:
```
resolution = 10
turnstile_weather['tempi_group'] =  turnstile_weather['tempi'].apply(lambda x: np.round(x/resolution)*resolution )
agg = turnstile_weather.groupby(['tempi_group'], as_index=False).mean()
df1 = turnstile_weather[turnstile_weather.tempi_group == 70]['ENTRIESn_hourly']
df2 = turnstile_weather[turnstile_weather.tempi_group == 80]['ENTRIESn_hourly']

test_has_effect(df1,df2 , 'with low temp', 'with high temp' ,show_graph=True)
```




```
As you can see, the distribution is not normal.
 After we perform mannwhitney t-test the conclusion is:
 p-values for two-tailed is 0.000000000.
 We reject the null hypothesis.
 "the distribution of ENTRIESn_hourly for the two groups are not equal"
 with low temp mean is 2021.6490019960
 with high temp mean is 2291.7881670534
```

In [12]:
```
resolution = 5
turnstile_weather['wspdi_group'] =  turnstile_weather['wspdi'].apply(lambda x: np.round(x/resolution)*resolution )
agg = turnstile_weather.groupby(['wspdi_group'], as_index=False).mean()

df1 = turnstile_weather[turnstile_weather.wspdi_group == 0]['ENTRIESn_hourly']
df2 = turnstile_weather[turnstile_weather.wspdi_group == 5]['ENTRIESn_hourly']
test_has_effect(df1,df2 , 'with low wspdi', 'with high wspdi' ,show_graph=True)
```




```
As you can see, the distribution is not normal.
 After we perform mannwhitney t-test the conclusion is:
 p-values for two-tailed is 0.000000000.
 We reject the null hypothesis.
 "the distribution of ENTRIESn_hourly for the two groups are not equal"
 with low wspdi mean is 1549.0320479863
 with high wspdi mean is 1715.1301188904
```

***Conclusion :***

We use rain, fog, tempi as features in our model and add UNIT, hour, wspdi and day_week to features using dummy varible.

In [13]:
```
features = turnstile_weather[['rain','fog','tempi']]
dummy_units = pd.get_dummies(turnstile_weather['UNIT'], prefix='unit')
features = features.join(dummy_units)
dummy_hour = pd.get_dummies(turnstile_weather['hour'], prefix='hour')
features = features.join(dummy_hour)
dummy_day_week = pd.get_dummies(turnstile_weather['day_week'], prefix='day_week')
features = features.join(dummy_day_week)
# dummy_tempi = pd.get_dummies(turnstile_weather['tempi_group'], prefix='tempi_group')
# features = features.join(dummy_tempi)
dummy_wspdi = pd.get_dummies(turnstile_weather['wspdi_group'], prefix='wspdi_group')
features = features.join(dummy_wspdi)
```

#### *2.3 Why did you select these features in your model? We are looking for specific reasons that lead you to believe that*

the selected features will contribute to the predictive power of your model.

Your reasons might be based on intuition. For example, response for fog might be: “I decided to use fog because I thought that when it is very foggy outside people might decide to use the subway more often.”

Your reasons might also be based on data exploration and experimentation, for example: “I used feature X because as soon as I included it in my model, it drastically improved my R2 value.”

**ANS:**

UNIT,Hour,day of week has obviously effect the ENTRIESn_hourly.

We decide to do not use station because it corresponding to the remote unit.

"rain,fog,tempi and precipi" has an effect by the t-test conclusion.

#### *2.4 What are the parameters (also known as "coefficients" or "weights") of the non-dummy features in your linear regression model?*

In [15]:
```
X = features
y = np.array(turnstile_weather['ENTRIESn_hourly'])
X = sm.add_constant(X)
olsmod = sm.OLS(y, X)
olsres = olsmod.fit()
ynewpred = olsres.predict(X)
intercept , params = olsres.params[0],olsres.params[1:]
```

**ANS:**

In [159]:
```
print "rain coefficients is %.10f"%params[0]
print "fog coefficients is %.10f"%params[1]
```

```
rain coefficients is -32.4709952114
fog coefficients is -182.0574321738
```

#### *2.5 What is your model’s R2 (coefficients of determination) value?*

In [160]:
```
print olsres.rsquared
```

```
0.545517676888
```

In [161]:
```
def plot_residuals(turnstile_weather, predictions):
    name_df1 = "Prediction by our models"
    df1 = (turnstile_weather['ENTRIESn_hourly'] - predictions)

    name_df2 = "Predict by mean value"
    df2 = (turnstile_weather['ENTRIESn_hourly'] - turnstile_weather['ENTRIESn_hourly'].mean())
    df_compare = pd.DataFrame({name_df1: df1,
               name_df2: df2})


    df_compare = pd.DataFrame({name_df1: df1,
               name_df2: df2})
    df_compare.plot(kind='hist', alpha=0.2,bins = range(-6000, 6000 , 300))




    plt.suptitle('Predict ENTRIESn_hourly Error compare between Our model and Mean')
    plt.xlabel("ENTRIES hourly Error")
    plt.ylabel("frequency")
    fig = matplotlib.pyplot.gcf()
    fig.set_size_inches(18.5, 10.5, forward=True)
    plt.show()
    return
```

**ANS:**

In statistics, the coefficient of determination, denoted R2 or r2 and pronounced R squared, is a number that indicates how well data fit a statistical model.

Our model's R2 is 0.54629540704.

The result indicates that the model explains 54.63% of the variability of the response data around its mean.

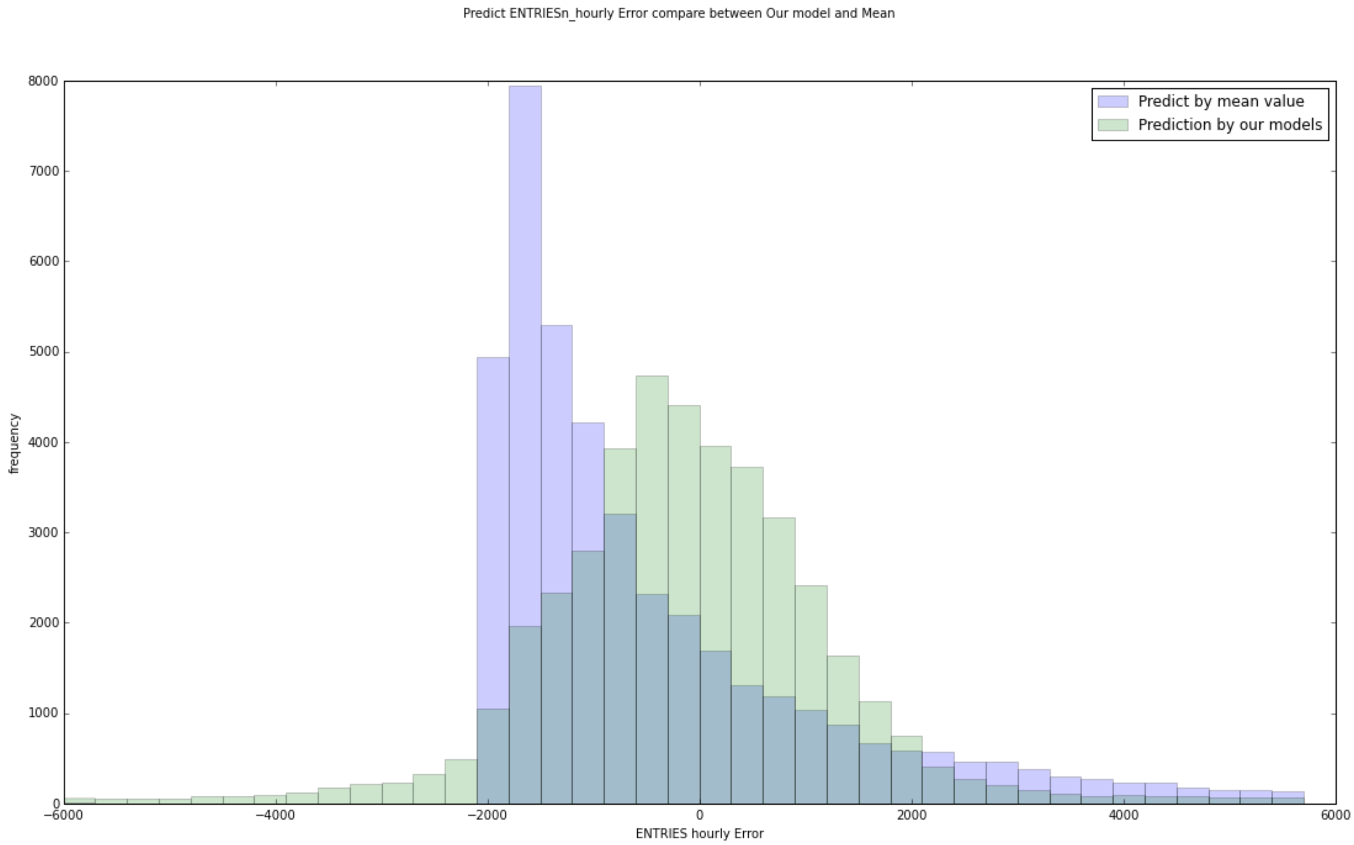
***2.6 What does this R2 value mean for the goodness of fit for your regression model? Do you think this linear model to predict ridership is appropriate for this dataset, given this R2 value?***

In [21]:
```
predictions = ynewpred
```

**ANS :**

```
Our R2(0.546) indicates that the regression line fits the data more than any other models that R2 less than 0.546 and still ha
s the chance to improve the model for make it closer to 1.
```

In [163]:
```
plot_residuals(turnstile_weather, predictions)
```




As you can see our model's prediction is obviously better than predict by mean.

# Section 3. Visualization

***3.1 One visualization should contain two histograms: one of ENTRIESn_hourly for rainy days and one of ENTRIESn_hourly for non-rainy days.***

You can combine the two histograms in a single plot or you can use two separate plots.

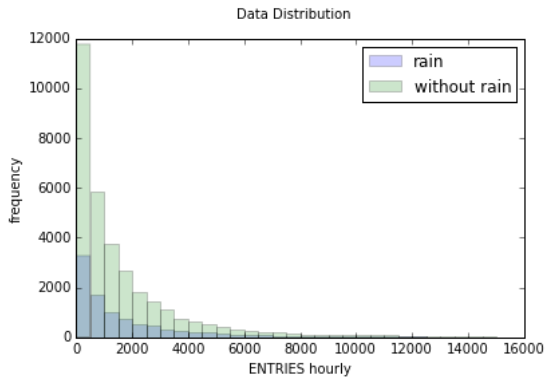
If you decide to use to two separate plots for the two histograms, please ensure that the x-axis limits for both of the plots are identical. It is much easier to compare the two in that case.

For the histograms, you should have intervals representing the volume of ridership (value of ENTRIESn_hourly) on the x-axis and the frequency of occurrence on the y-axis. For example, each interval (along the x-axis), the height of the bar for this interval will represent the number of records (rows in our data) that have ENTRIESn_hourly that falls in this interval.

Remember to increase the number of bins in the histogram (by having larger number of bars). The default bin width is not sufficient to capture the variability in the two samples.

**ANS:**

In [164]:
```
rain_df = turnstile_weather[turnstile_weather.rain == 1]['ENTRIESn_hourly']
norain_df = turnstile_weather[turnstile_weather.rain == 0]['ENTRIESn_hourly']
df_compare = pd.DataFrame({'rain': rain_df,
               'without rain': norain_df})
df_compare.plot(kind='hist', alpha=0.2,bins = range(0, 15000 + 500, 500) ) ##,bins = range(0, 37 + 3, 3)

plt.xlabel("ENTRIES hourly")
plt.suptitle('Data Distribution')
plt.ylabel("frequency")
plt.show()

print "As you can see, the distribution is not normal."
```




```
As you can see, the distribution is not normal.
```

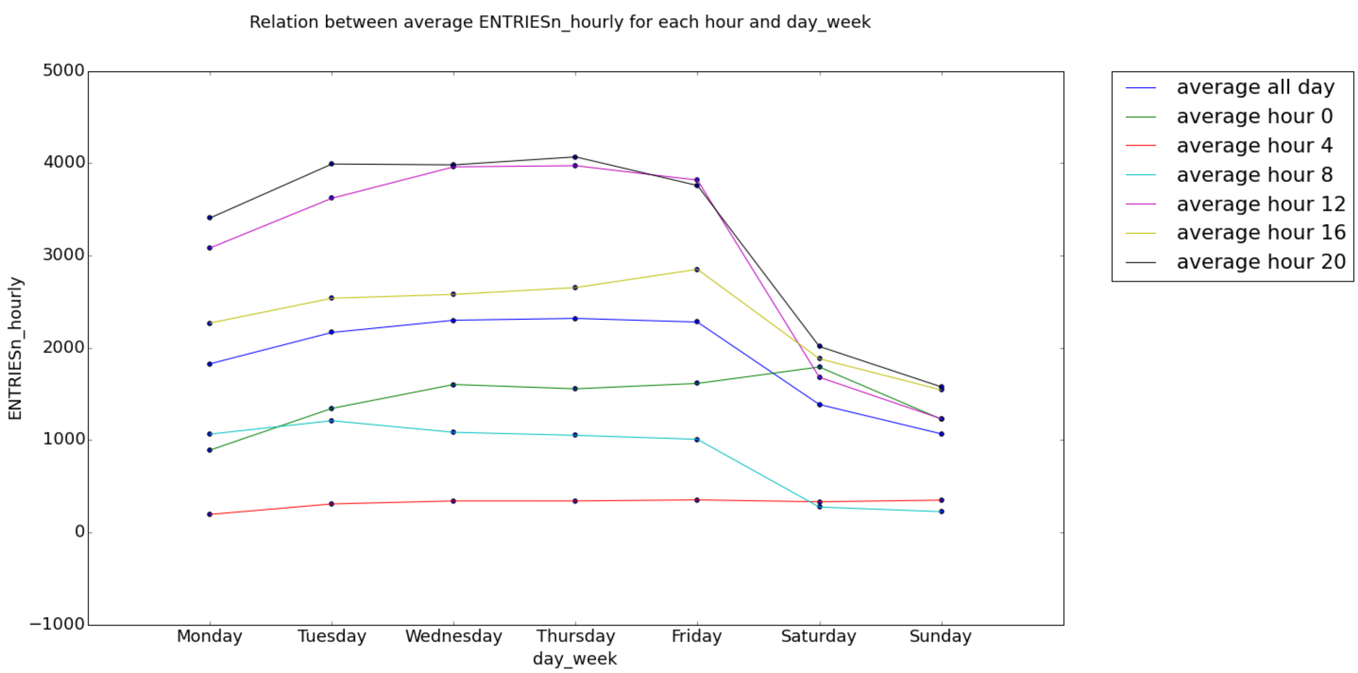
***3.2 One visualization can be more freeform. You should feel free to implement something that we discussed in class (e.g., scatter plots, line plots) or attempt to implement something more advanced if you'd like. Some suggestions are:***

Ridership by time-of-day

Ridership by day-of-week

**ANS:**

In [35]:
```
import matplotlib
matplotlib.rcParams.update({'font.size': 18})
agg = turnstile_weather.groupby(['day_week'], as_index=False).mean()

# from matplotlib.dates import WeekdayLocator
# fig, ax = plt.subplots()
# ax.xaxis.set_major_locator(WeekdayLocator(byweekday=MO))

x_data  = agg['day_week']
y_data = agg['ENTRIESn_hourly']
plt.plot(x_data, y_data,label="average all day")
plt.scatter(x=agg['day_week'], y=agg['ENTRIESn_hourly'])

for hour in range(0,25):
    agg = turnstile_weather[turnstile_weather.hour == hour].groupby(['day_week'], as_index=False).mean()
    x_data  = agg['day_week']
    y_data = agg['ENTRIESn_hourly']
    if np.mean(y_data) > 0  :
        plt.plot(x_data, y_data, label="average hour %s"%hour)
        plt.scatter(x=agg['day_week'], y=agg['ENTRIESn_hourly'])

plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.)

plt.suptitle('Relation between average ENTRIESn_hourly for each hour and day_week')


x = np.array([0,1,2,3,4,5,6])
my_xticks = ["Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday", "Sunday"]
plt.xticks(x, my_xticks)

plt.xlabel("day_week")
plt.ylabel("ENTRIESn_hourly")



fig = matplotlib.pyplot.gcf()
fig.set_size_inches(18.5, 10.5, forward=True)
plt.show()
print "As you can see the average ENTRIESn_hourly of hour 20 and 12 is obviously higher than other"
```




```
As you can see the average ENTRIESn_hourly of hour 20 and 12 is obviously higher than other
```

## Section 4. Conclusion

Please address the following questions in detail. Your answers should be 1-2 paragraphs long.

**4.1 From your analysis and interpretation of the data, do more people ride the NYC subway when it is raining or when it is not raining?**

**To find out how exact rain affect to ENTRIESn_hourly we need to change the set of features to rebuild the regression model which has only rain as parameter.**

In [23]:
```
X = turnstile_weather[['rain']]
y = np.array(turnstile_weather['ENTRIESn_hourly'])
olsmod_rain = sm.OLS(y, X)
olsres_rain = olsmod_rain.fit()
ynewpred_rain = olsres_rain.predict(X)
rain_coef = olsres_rain.params[0]
print(olsres_rain.summary())
```

```
                            OLS Regression Results                            
==============================================================================
Dep. Variable:                      y   R-squared:                       0.075
Model:                            OLS   Adj. R-squared:                  0.075
Method:                 Least Squares   F-statistic:                     3473.
Date:                Thu, 12 Nov 2015   Prob (F-statistic):               0.00
Time:                        01:27:01   Log-Likelihood:            -4.0693e+05
No. Observations:               42649   AIC:                         8.139e+05
Df Residuals:                   42648   BIC:                         8.139e+05
Df Model:                           1                                         
Covariance Type:            nonrobust                                         
==============================================================================
                 coef    std err          t      P>|t|      [95.0% Conf. Int.]
------------------------------------------------------------------------------
rain        2028.1960     34.413     58.936      0.000      1960.745  2095.647
==============================================================================
Omnibus:                    32835.376   Durbin-Watson:                   0.714
Prob(Omnibus):                  0.000   Jarque-Bera (JB):           792725.563
Skew:                           3.559   Prob(JB):                         0.00
Kurtosis:                      22.885   Cond. No.                         1.00
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

### ANS:

From rain's coeffiecent(2028.1960) in linear regression and P value is 0.000 so we conclude that :

**Rain positively affect on ENTRIESn_hourly (rain increase ENTRIESn_hourly) .**

**4.2 What analyses lead you to this conclusion? You should use results from both your statistical tests and your linear regression to support your analysis.**

### ANS:

From the null hypothesis testing results with 2 samples rain and with out rain return p value = 0.00000274106957 it mean that the possibility of distribution different has occured by chance is about 0.0002741% so we assume that

***The distributions of both populations are not equal.***

The p-value from linear regression summary (0.000) is less than the common alpha level of 0.05 and coef(2028.1960) is positive which indicates that:

***Rain positively affect on ENTRIESn_hourly (rain increase ENTRIESn_hourly) .***

# Section 5. Reflection

Please address the following questions in detail. Your answers should be 1-2 paragraphs long.

**5.1 Please discuss potential shortcomings of the methods of your analysis, including:**

Dataset, Analysis, such as the linear regression model or statistical test.

**ANS**

***Dataset:***

Folowing the data :

The data has only one month data and I quite sure that "month" has an effect on "ENTRIESn_hourly" so If we have more data in every month is possible that our model is more fit on the different month.

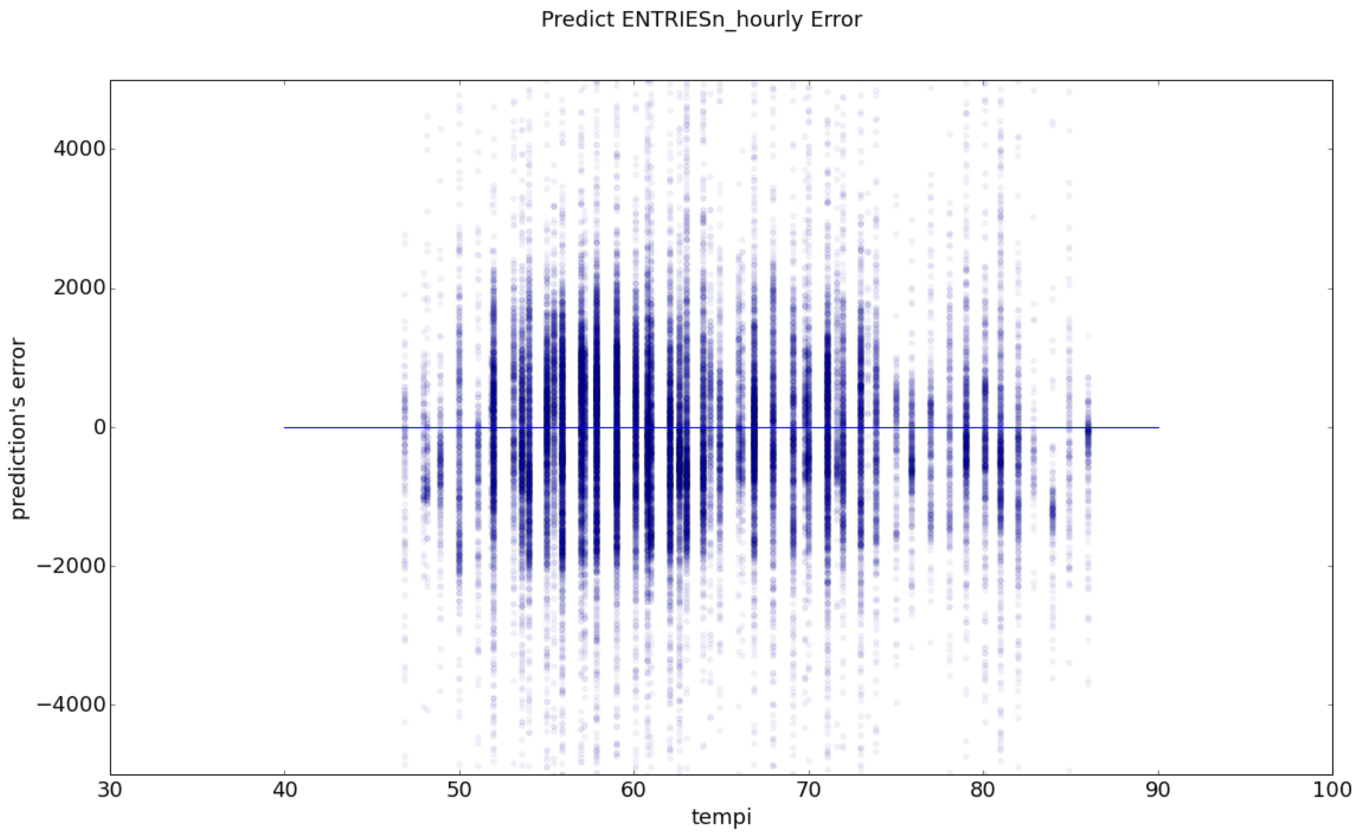
I am quite sure that "public holiday" has an effect on "ENTRIESn_hourly" so If we has the parameter, our prediction will be more significant accurate.

***Statistical Test:***

The Statistical Test is only comparing the differences between the conditions of one feature;rain, when clearly other are other variables that seem to affect the ridership even more.

***Regression Model***

In [31]:
```
def plot_residuals2(turnstile_weather, effect_parameter='tempi'):
    name_df1 = "Prediction by our models"
    turnstile_weather['Predict_ENTRIESn_hourly_error'] = (turnstile_weather['ENTRIESn_hourly'] - predictions)
    agg = turnstile_weather
#     agg = turnstile_weather.groupby([effect_parameter], as_index=False).mean()
    agg = agg.sort([effect_parameter])
    x_data  = agg[effect_parameter]
    y_data = agg['Predict_ENTRIESn_hourly_error']
    fig = matplotlib.pyplot.gcf()
    fig.set_size_inches(18.5, 10.5, forward=True)
#     plt.plot(x_data, y_data)
    plt.plot([40,90], [0,0])
    plt.xlabel("tempi")
    plt.ylabel("prediction's error")
    plt.suptitle('Predict ENTRIESn_hourly Error')
    plt.scatter(x=agg[effect_parameter], y=agg['Predict_ENTRIESn_hourly_error'], alpha = 0.05)
    x1,x2,y1,y2 = plt.axis()
    plt.axis((x1,x2,-5000,5000))
    plt.show()

    return

plot_residuals2(turnstile_weather, effect_parameter='tempi')
```



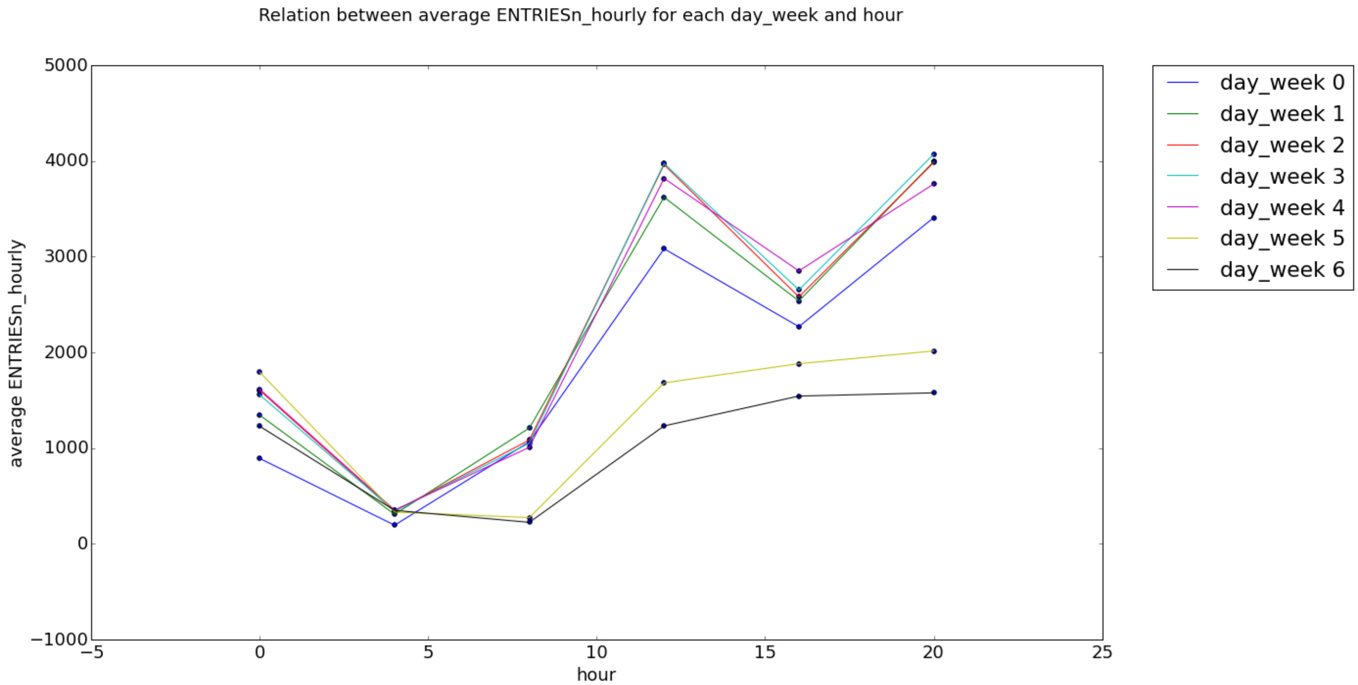

As you see "Predict ENTRIESn_hourly" is quite acurrate (0 horizontal line draw through the middle of intense blue dot cluster) so I think linear model in this context is good enough.

Any way if we find out other pattern such as cyclic or polynomial the linear model will be less appropriate.

***5.2 (Optional) Do you have any other insight about the dataset that you would like to share with us?***

**ANS:**

In [32]:
```
for day_week in range(0,7):
    agg = turnstile_weather[turnstile_weather.day_week == day_week].groupby(['hour'], as_index=False).mean()
    x_data  = agg['hour']
    y_data = agg['ENTRIESn_hourly']
    if np.mean(y_data) > 0  :
        plt.plot(x_data, y_data, label="day_week %s"%day_week)
        plt.scatter(x=agg['hour'], y=agg['ENTRIESn_hourly'])
plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.)
plt.xlabel("hour")
plt.ylabel("average ENTRIESn_hourly")
plt.suptitle('Relation between average ENTRIESn_hourly for each day_week and hour')
fig = matplotlib.pyplot.gcf()
fig.set_size_inches(18.5, 10.5, forward=True)
plt.show()
```

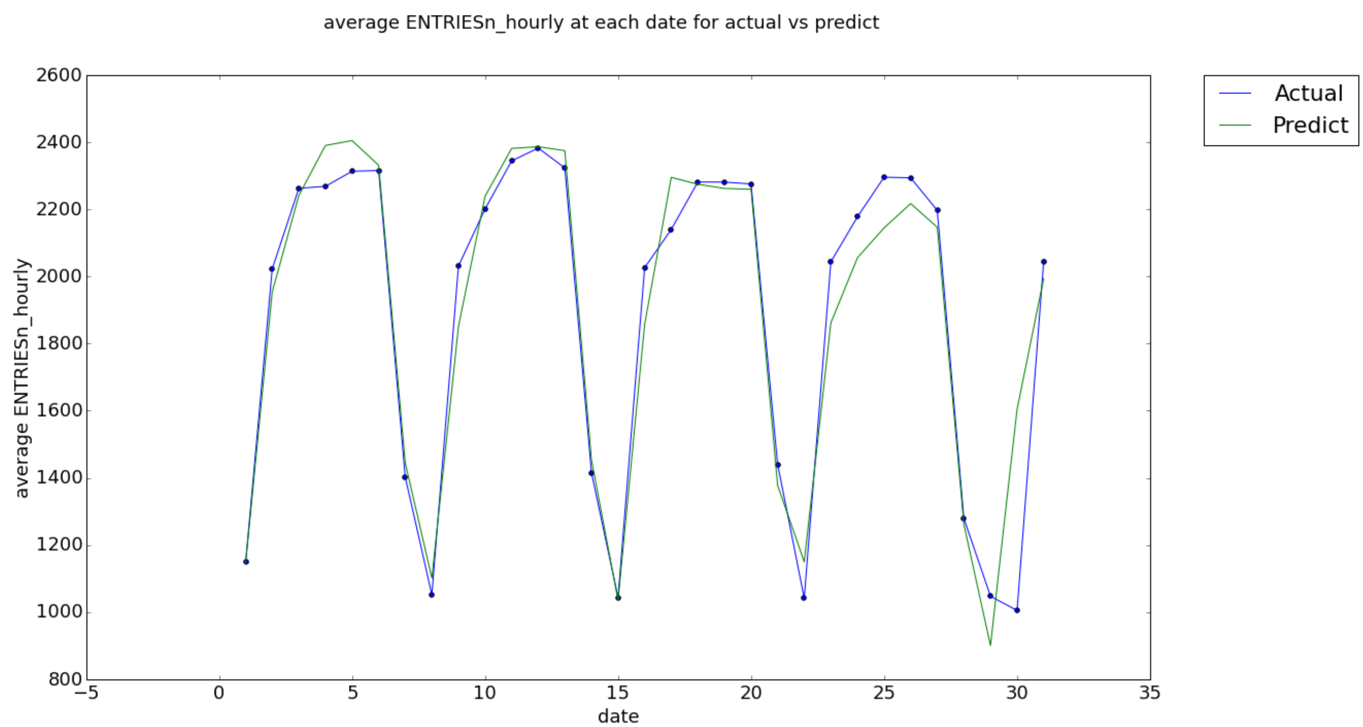



There are 2 patterns of Hourly_entries by hour first is day_week 0-4 second is dayweek 5-6.

Peek Hour for Hourly_entries are about at 12:00 and 20:00.

In [189]:
```
# agg2 = turnstile_weather.groupby(['conds'])
# for key, cond  in agg2['conds']:
# #     print key
#     if key in ['Clear','Light Drizzle','Heavy Rain','Light Rain']:
#         agg = turnstile_weather[turnstile_weather.conds == key].groupby(['hour'], as_index=False).mean()
#         x_data  = agg['hour']
#         y_data = agg['ENTRIESn_hourly']
# #         print np.mean(y_data)
#         if np.mean(y_data) > 0  :
#             plt.plot(x_data, y_data, label="cond %s"%key)
#             plt.scatter(x=agg['hour'], y=agg['ENTRIESn_hourly'])

# plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.)
# plt.xlabel("hour")
# plt.ylabel("average ENTRIESn_hourly")
# plt.suptitle('Relation between average ENTRIESn_hourly for each cond and hour')
# fig = matplotlib.pyplot.gcf()
# fig.set_size_inches(18.5, 10.5, forward=True)
# plt.show()
```

In [33]:
```
fig = matplotlib.pyplot.gcf()
fig.set_size_inches(18.5, 10.5)

turnstile_weather['predict_ENTRIESn_hourly'] = predictions
agg = turnstile_weather.groupby(['DATEn'], as_index=False).mean()
# agg['datetime_obj']  = pd.to_datetime(agg['DATEn'])
# agg['datetime_obj']  = pd.DatetimeIndex (agg['DATEn'])
# agg['datetime_obj']  = agg['DATEn'].astype('datetime64[ns]')
from time import strptime
# agg = agg[:30]
agg['datetime_obj']  = pd.to_datetime(agg['DATEn'])
# print agg['datetime_obj'][0].day
agg['datetime_num']  =  agg['datetime_obj'].apply(lambda x: x.day )

agg = agg.sort(['datetime_num'])
# print len(agg)

import matplotlib.dates as mdates
x_data = agg['datetime_num']
y_data = agg['ENTRIESn_hourly']
y2_data = agg['predict_ENTRIESn_hourly']

plt.scatter(x=agg['datetime_num'], y=agg['ENTRIESn_hourly'])
plt.plot(agg['datetime_num'], agg['ENTRIESn_hourly'],label="Actual")

plt.plot(agg['datetime_num'], agg['predict_ENTRIESn_hourly'],label="Predict")
plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.)

plt.xlabel("date")
plt.ylabel("average ENTRIESn_hourly")
plt.suptitle('average ENTRIESn_hourly at each date for actual vs predict')
plt.show()
```




The pattern are cyclic.

## APPENDIX

#### My linear regression models ¶

In [20]:
```
print(olsres.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.546
Model:                            OLS   Adj. R-squared:                  0.543
Method:                 Least Squares   F-statistic:                     197.2
Date:                Thu, 12 Nov 2015   Prob (F-statistic):               0.00
Time:                        01:08:34   Log-Likelihood:            -3.8448e+05
No. Observations:               42649   AIC:                         7.695e+05
Df Residuals:                   42390   BIC:                         7.717e+05
Df Model:                         258
Covariance Type:            nonrobust
===================================================================================
                      coef    std err          t      P>|t|      [95.0% Conf. Int.]
-----------------------------------------------------------------------------------
const            1772.8534     63.005     28.138      0.000      1649.363  1896.344
rain              -32.4710     25.903     -1.254      0.210       -83.241    18.299
fog              -182.0574    100.800     -1.806      0.071      -379.628    15.513
tempi             -14.7173      1.362    -10.807      0.000       -17.386   -12.048
unit_R003       -1693.6559    153.828    -11.010      0.000     -1995.161 -1392.150
unit_R004       -1326.0177    150.816     -8.792      0.000     -1621.620 -1030.416
unit_R005       -1341.2832    152.123     -8.817      0.000     -1639.446 -1043.120
unit_R006       -1162.6273    148.755     -7.816      0.000     -1454.189  -871.065
unit_R007       -1526.3967    153.030     -9.974      0.000     -1826.339 -1226.454
unit_R008       -1528.3240    153.465     -9.959      0.000     -1829.118 -1227.530
unit_R009       -1526.5142    150.828    -10.121      0.000     -1822.140 -1230.888
unit_R011        5563.9756    146.934     37.867      0.000      5275.983  5851.969
unit_R012        6915.9853    146.146     47.322      0.000      6629.536  7202.435
unit_R013         814.4261    146.146      5.573      0.000       527.977  1100.875
unit_R016       -1009.1465    146.935     -6.868      0.000     -1297.143  -721.150
unit_R017        2429.4207    146.146     16.623      0.000      2142.971  2715.870
unit_R018        5997.2138    146.620     40.903      0.000      5709.836  6284.591
unit_R019        1464.7466    146.336     10.009      0.000      1177.925  1751.568
unit_R020        4605.5175    146.146     31.513      0.000      4319.068  4891.967
unit_R021        2915.5924    146.933     19.843      0.000      2627.601  3203.584
unit_R022        7749.9100    146.146     53.029      0.000      7463.461  8036.359
unit_R023        4385.0874    146.146     30.005      0.000      4098.638  4671.537
unit_R024        1426.6750    146.728      9.723      0.000      1139.085  1714.265
unit_R025        3561.6390    146.336     24.339      0.000      3274.817  3848.461
unit_R027        1199.5229    146.146      8.208      0.000       913.074  1485.972
unit_R029        5461.4960    146.146     37.370      0.000      5175.047  5747.945
unit_R030        1331.6896    146.146      9.112      0.000      1045.240  1618.139
unit_R031        2583.4584    146.146     17.677      0.000      2297.009  2869.908
unit_R032        2678.1519    146.538     18.276      0.000      2390.934  2965.370
unit_R033        6466.4691    146.146     44.247      0.000      6180.020  6752.918
unit_R034        -651.9222    152.836     -4.266      0.000      -951.484  -352.361
unit_R035        1028.8096    146.933      7.002      0.000       740.819  1316.800
unit_R036       -1057.2232    149.884     -7.054      0.000     -1350.999  -763.447
unit_R037        -933.6930    147.407     -6.334      0.000     -1222.613  -644.773
unit_R038       -1587.3834    150.723    -10.532      0.000     -1882.803 -1291.964
unit_R039       -1067.9196    154.730     -6.902      0.000     -1371.194  -764.645
unit_R040        -533.4530    147.011     -3.629      0.000      -821.598  -245.308
unit_R041        1331.5014    146.146      9.111      0.000      1045.052  1617.951
unit_R042       -1185.1288    148.549     -7.978      0.000     -1476.287  -893.970
unit_R043        1118.6358    146.146      7.654      0.000       832.186  1405.085
unit_R044        2910.0444    146.146     19.912      0.000      2623.595  3196.494
unit_R046        6576.4960    146.146     44.999      0.000      6290.047  6862.945
unit_R049        1004.3078    146.146      6.872      0.000       717.859  1290.757
unit_R050        2259.2547    146.935     15.376      0.000      1971.258  2547.251
unit_R051        3366.0498    146.146     23.032      0.000      3079.600  3652.499
unit_R052        -585.4247    152.039     -3.851      0.000      -883.423  -287.426
unit_R053        1392.0730    147.017      9.469      0.000      1103.917  1680.229
unit_R054        -307.8866    146.932     -2.095      0.036      -595.877   -19.896
unit_R055        6557.1628    146.226     44.843      0.000      6270.557  6843.769
unit_R056        -322.9592    146.934     -2.198      0.028      -610.952   -34.966
unit_R057        3114.4369    146.146     21.310      0.000      2827.988  3400.886
unit_R058       -1125.9778    146.541     -7.684      0.000     -1413.201  -838.755
unit_R059        -585.5432    149.796     -3.909      0.000      -879.147  -291.940
unit_R060        -982.7879    148.547     -6.616      0.000     -1273.943  -691.633
unit_R061       -1186.9780    154.195     -7.698      0.000     -1489.204  -884.752
unit_R062         968.2971    146.146      6.626      0.000       681.848  1254.746
unit_R063        -640.1265    153.744     -4.164      0.000      -941.468  -338.785
unit_R064        -931.9130    150.217     -6.204      0.000     -1226.340  -637.486
unit_R065        -940.5654    151.949     -6.190      0.000     -1238.388  -642.743
unit_R066       -1546.9678    152.839    -10.122      0.000     -1846.535 -1247.400
unit_R067        -937.0028    154.285     -6.073      0.000     -1239.404  -634.601
unit_R068       -1346.2432    154.291     -8.725      0.000     -1648.657 -1043.829
unit_R069        -851.8829    151.597     -5.619      0.000     -1149.016  -554.750
unit_R070          19.2756    146.146      0.132      0.895      -267.174   305.725
unit_R080        1843.6035    146.146     12.615      0.000      1557.154  2130.053
unit_R081        1794.3124    146.933     12.212      0.000      1506.321  2082.304
```

| | | | | | | |
|---|---|---|---|---|---|---|
| unit_R082 | -266.2321 | 146.933 | -1.812 | 0.070 | -554.223 | 21.759 |
| unit_R083 | 1358.0121 | 146.146 | 9.292 | 0.000 | 1071.563 | 1644.461 |
| unit_R084 | 8262.2863 | 146.146 | 56.534 | 0.000 | 7975.837 | 8548.736 |
| unit_R085 | 843.5019 | 146.935 | 5.741 | 0.000 | 555.507 | 1131.497 |
| unit_R086 | 831.6035 | 146.146 | 5.690 | 0.000 | 545.154 | 1118.053 |
| unit_R087 | -539.9019 | 147.735 | -3.655 | 0.000 | -829.465 | -250.338 |
| unit_R089 | -1255.3585 | 146.935 | -8.544 | 0.000 | -1543.355 | -967.362 |
| unit_R090 | -1370.5178 | 154.287 | -8.883 | 0.000 | -1672.923 | -1068.113 |
| unit_R091 | -658.5260 | 152.920 | -4.306 | 0.000 | -958.252 | -358.800 |
| unit_R092 | 219.1146 | 150.312 | 1.458 | 0.145 | -75.499 | 513.729 |
| unit_R093 | 247.3605 | 151.163 | 1.636 | 0.102 | -48.922 | 543.643 |
| unit_R094 | -7.9565 | 147.012 | -0.054 | 0.957 | -296.102 | 280.189 |
| unit_R095 | 398.7608 | 148.222 | 2.690 | 0.007 | 108.243 | 689.279 |
| unit_R096 | 585.2179 | 146.614 | 3.992 | 0.000 | 297.851 | 872.585 |
| unit_R097 | 1206.2517 | 146.620 | 8.227 | 0.000 | 918.874 | 1493.629 |
| unit_R098 | 66.1627 | 146.146 | 0.453 | 0.651 | -220.287 | 352.612 |
| unit_R099 | 623.1896 | 146.146 | 4.264 | 0.000 | 336.740 | 909.639 |
| unit_R100 | -1211.7167 | 147.823 | -8.197 | 0.000 | -1501.452 | -921.982 |
| unit_R101 | 1057.5820 | 146.146 | 7.236 | 0.000 | 771.133 | 1344.031 |
| unit_R102 | 1946.4207 | 146.146 | 13.318 | 0.000 | 1659.971 | 2232.870 |
| unit_R103 | -377.8285 | 152.029 | -2.485 | 0.013 | -675.808 | -79.849 |
| unit_R104 | -418.4314 | 147.008 | -2.846 | 0.004 | -706.571 | -130.292 |
| unit_R105 | 1596.1197 | 146.146 | 10.921 | 0.000 | 1309.670 | 1882.569 |
| unit_R106 | -711.6355 | 154.284 | -4.612 | 0.000 | -1014.036 | -409.235 |
| unit_R107 | -1303.7224 | 154.759 | -8.424 | 0.000 | -1607.054 | -1000.391 |
| unit_R108 | 3485.2863 | 146.146 | 23.848 | 0.000 | 3198.837 | 3771.736 |
| unit_R111 | 1482.4100 | 146.146 | 10.143 | 0.000 | 1195.961 | 1768.859 |
| unit_R112 | -73.0956 | 146.614 | -0.499 | 0.618 | -360.462 | 214.271 |
| unit_R114 | -858.9462 | 146.731 | -5.854 | 0.000 | -1146.541 | -571.351 |
| unit_R115 | -487.7642 | 146.336 | -3.333 | 0.001 | -774.586 | -200.943 |
| unit_R116 | 1461.5552 | 146.146 | 10.001 | 0.000 | 1175.106 | 1748.004 |
| unit_R117 | -899.2615 | 153.821 | -5.846 | 0.000 | -1200.754 | -597.769 |
| unit_R119 | 92.4627 | 149.038 | 0.620 | 0.535 | -199.654 | 384.580 |
| unit_R120 | -249.0357 | 151.154 | -1.648 | 0.099 | -545.301 | 47.229 |
| unit_R121 | -283.3014 | 150.220 | -1.886 | 0.059 | -577.736 | 11.133 |
| unit_R122 | 817.9531 | 148.227 | 5.518 | 0.000 | 527.425 | 1108.482 |
| unit_R123 | -114.8121 | 148.549 | -0.773 | 0.440 | -405.972 | 176.348 |
| unit_R124 | -1103.4509 | 152.389 | -7.241 | 0.000 | -1402.137 | -804.765 |
| unit_R126 | 123.4638 | 146.146 | 0.845 | 0.398 | -162.986 | 409.913 |
| unit_R127 | 3064.1734 | 146.146 | 20.967 | 0.000 | 2777.724 | 3350.623 |
| unit_R137 | 703.0714 | 146.226 | 4.808 | 0.000 | 416.465 | 989.678 |
| unit_R139 | 798.1584 | 146.540 | 5.447 | 0.000 | 510.937 | 1085.380 |
| unit_R163 | 1599.9100 | 146.146 | 10.947 | 0.000 | 1313.461 | 1886.359 |
| unit_R172 | 165.9476 | 146.146 | 1.135 | 0.256 | -120.502 | 452.397 |
| unit_R179 | 5046.7218 | 146.146 | 34.532 | 0.000 | 4760.273 | 5333.171 |
| unit_R181 | 23.1992 | 148.959 | 0.156 | 0.876 | -268.763 | 315.162 |
| unit_R183 | -1020.1679 | 154.745 | -6.593 | 0.000 | -1323.471 | -716.865 |
| unit_R184 | -758.2779 | 152.836 | -4.961 | 0.000 | -1057.839 | -458.717 |
| unit_R186 | -653.5763 | 148.139 | -4.412 | 0.000 | -943.933 | -363.220 |
| unit_R188 | 599.8546 | 146.538 | 4.093 | 0.000 | 312.636 | 887.073 |
| unit_R189 | -343.3938 | 149.795 | -2.292 | 0.022 | -636.995 | -49.793 |
| unit_R194 | 262.1646 | 148.965 | 1.760 | 0.078 | -29.809 | 554.138 |
| unit_R196 | -400.7090 | 147.331 | -2.720 | 0.007 | -689.481 | -111.937 |
| unit_R198 | 383.7891 | 146.934 | 2.612 | 0.009 | 95.796 | 671.782 |
| unit_R199 | -1016.0686 | 148.963 | -6.821 | 0.000 | -1308.039 | -724.098 |
| unit_R200 | -690.4024 | 147.910 | -4.668 | 0.000 | -980.310 | -400.495 |
| unit_R202 | 490.4928 | 147.013 | 3.336 | 0.001 | 202.345 | 778.641 |
| unit_R203 | 29.8105 | 150.643 | 0.198 | 0.843 | -265.454 | 325.075 |
| unit_R204 | -272.5524 | 146.146 | -1.865 | 0.062 | -559.002 | 13.897 |
| unit_R205 | -228.9959 | 147.817 | -1.549 | 0.121 | -518.720 | 60.728 |
| unit_R207 | 277.4172 | 146.540 | 1.893 | 0.058 | -9.803 | 564.638 |
| unit_R208 | 773.3186 | 147.810 | 5.232 | 0.000 | 483.608 | 1063.029 |
| unit_R209 | -964.1558 | 153.738 | -6.271 | 0.000 | -1265.485 | -662.827 |
| unit_R210 | -1199.4997 | 149.801 | -8.007 | 0.000 | -1493.113 | -905.886 |
| unit_R211 | 684.8777 | 146.146 | 4.686 | 0.000 | 398.428 | 971.327 |
| unit_R212 | -35.0703 | 146.538 | -0.239 | 0.811 | -322.288 | 252.147 |
| unit_R213 | -576.2380 | 149.375 | -3.858 | 0.000 | -869.016 | -283.460 |
| unit_R214 | -1089.2390 | 154.199 | -7.064 | 0.000 | -1391.472 | -787.006 |
| unit_R215 | -126.4400 | 146.934 | -0.861 | 0.390 | -414.434 | 161.554 |
| unit_R216 | -955.4743 | 146.933 | -6.503 | 0.000 | -1243.465 | -667.483 |
| unit_R217 | -752.0586 | 153.741 | -4.892 | 0.000 | -1053.395 | -450.722 |
| unit_R218 | 249.6617 | 146.618 | 1.703 | 0.089 | -37.712 | 537.035 |
| unit_R219 | -469.3956 | 147.013 | -3.193 | 0.001 | -757.543 | -181.248 |
| unit_R220 | -240.3965 | 146.146 | -1.645 | 0.100 | -526.846 | 46.053 |
| unit_R221 | -342.2014 | 154.201 | -2.219 | 0.026 | -644.438 | -39.965 |
| unit_R223 | 389.1639 | 146.614 | 2.654 | 0.008 | 101.797 | 676.531 |
| unit_R224 | -999.5511 | 150.222 | -6.654 | 0.000 | -1293.988 | -705.114 |
| unit_R225 | -1148.2865 | 148.142 | -7.751 | 0.000 | -1438.648 | -857.925 |
| unit_R226 | -1039.8991 | 152.920 | -6.800 | 0.000 | -1339.626 | -740.172 |
| unit_R227 | -621.1330 | 146.146 | -4.250 | 0.000 | -907.582 | -334.684 |

| | | | | | | |
|---|---|---|---|---|---|---|
| unit_R228 | -607.4424 | 153.291 | -3.963 | 0.000 | -907.896 | -306.989 |
| unit_R229 | -1176.1316 | 152.388 | -7.718 | 0.000 | -1474.816 | -877.448 |
| unit_R230 | -1215.1252 | 150.644 | -8.066 | 0.000 | -1510.390 | -919.860 |
| unit_R231 | -754.9510 | 148.139 | -5.096 | 0.000 | -1045.307 | -464.595 |
| unit_R232 | -720.6861 | 151.949 | -4.743 | 0.000 | -1018.510 | -422.862 |
| unit_R233 | -653.2971 | 155.132 | -4.211 | 0.000 | -957.359 | -349.235 |
| unit_R234 | -1443.0872 | 153.750 | -9.386 | 0.000 | -1744.441 | -1141.733 |
| unit_R235 | 852.1249 | 146.541 | 5.815 | 0.000 | 564.902 | 1139.348 |
| unit_R236 | -213.2885 | 147.813 | -1.443 | 0.149 | -503.004 | 76.427 |
| unit_R237 | -1106.3668 | 153.361 | -7.214 | 0.000 | -1406.957 | -805.776 |
| unit_R238 | 371.6645 | 146.618 | 2.535 | 0.011 | 84.291 | 659.038 |
| unit_R239 | -823.9717 | 146.146 | -5.638 | 0.000 | -1110.421 | -537.522 |
| unit_R240 | 935.0171 | 147.330 | 6.346 | 0.000 | 646.248 | 1223.786 |
| unit_R242 | -1183.5001 | 149.380 | -7.923 | 0.000 | -1476.289 | -890.711 |
| unit_R243 | -368.5493 | 151.593 | -2.431 | 0.015 | -665.674 | -71.424 |
| unit_R244 | -135.5100 | 151.589 | -0.894 | 0.371 | -432.627 | 161.607 |
| unit_R246 | -1067.5961 | 153.289 | -6.965 | 0.000 | -1368.045 | -767.147 |
| unit_R247 | -1617.8973 | 156.083 | -10.366 | 0.000 | -1923.823 | -1311.971 |
| unit_R248 | 1380.4881 | 146.540 | 9.421 | 0.000 | 1093.267 | 1667.709 |
| unit_R249 | -369.0032 | 148.548 | -2.484 | 0.013 | -660.161 | -77.845 |
| unit_R250 | -758.4554 | 149.375 | -5.078 | 0.000 | -1051.234 | -465.677 |
| unit_R251 | -450.9028 | 147.331 | -3.060 | 0.002 | -739.674 | -162.132 |
| unit_R252 | -765.3310 | 146.932 | -5.209 | 0.000 | -1053.322 | -477.341 |
| unit_R253 | -998.3039 | 152.054 | -6.565 | 0.000 | -1296.332 | -700.276 |
| unit_R254 | 858.1375 | 146.620 | 5.853 | 0.000 | 570.760 | 1145.515 |
| unit_R255 | -969.8776 | 147.916 | -6.557 | 0.000 | -1259.797 | -679.959 |
| unit_R256 | -658.5975 | 146.933 | -4.482 | 0.000 | -946.589 | -370.606 |
| unit_R257 | 163.2003 | 146.146 | 1.117 | 0.264 | -123.249 | 449.650 |
| unit_R258 | -201.1006 | 147.331 | -1.365 | 0.172 | -489.872 | 87.671 |
| unit_R259 | -800.6017 | 148.547 | -5.390 | 0.000 | -1091.757 | -509.446 |
| unit_R260 | -653.5131 | 159.531 | -4.096 | 0.000 | -966.196 | -340.830 |
| unit_R261 | -185.0256 | 149.878 | -1.235 | 0.217 | -478.790 | 108.738 |
| unit_R262 | -1308.2648 | 156.078 | -8.382 | 0.000 | -1614.180 | -1002.349 |
| unit_R263 | -1597.9984 | 149.374 | -10.698 | 0.000 | -1890.774 | -1305.222 |
| unit_R264 | -1251.1566 | 146.540 | -8.538 | 0.000 | -1538.378 | -963.936 |
| unit_R265 | -884.8970 | 152.393 | -5.807 | 0.000 | -1183.589 | -586.205 |
| unit_R266 | -828.7194 | 146.615 | -5.652 | 0.000 | -1116.088 | -541.351 |
| unit_R269 | -843.9177 | 146.935 | -5.743 | 0.000 | -1131.913 | -555.923 |
| unit_R270 | -1265.1981 | 153.288 | -8.254 | 0.000 | -1565.645 | -964.751 |
| unit_R271 | -1381.9898 | 152.834 | -9.042 | 0.000 | -1681.547 | -1082.432 |
| unit_R273 | -378.7498 | 153.745 | -2.463 | 0.014 | -680.093 | -77.406 |
| unit_R274 | -787.9245 | 151.945 | -5.186 | 0.000 | -1085.740 | -490.109 |
| unit_R275 | -854.2737 | 149.888 | -5.699 | 0.000 | -1148.057 | -560.490 |
| unit_R276 | -314.2621 | 146.146 | -2.150 | 0.032 | -600.711 | -27.813 |
| unit_R277 | -1339.9156 | 157.531 | -8.506 | 0.000 | -1648.679 | -1031.153 |
| unit_R278 | -1380.1016 | 152.397 | -9.056 | 0.000 | -1678.802 | -1081.401 |
| unit_R279 | -985.4616 | 149.042 | -6.612 | 0.000 | -1277.587 | -693.336 |
| unit_R280 | -1114.6907 | 156.551 | -7.120 | 0.000 | -1421.534 | -807.848 |
| unit_R281 | -450.9962 | 148.544 | -3.036 | 0.002 | -742.145 | -159.847 |
| unit_R282 | -143.6877 | 146.935 | -0.978 | 0.328 | -431.683 | 144.308 |
| unit_R284 | -956.9343 | 146.932 | -6.513 | 0.000 | -1244.925 | -668.944 |
| unit_R285 | -1186.5201 | 154.374 | -7.686 | 0.000 | -1489.096 | -883.944 |
| unit_R287 | -1221.9907 | 154.663 | -7.901 | 0.000 | -1525.133 | -918.848 |
| unit_R291 | 78.9691 | 146.146 | 0.540 | 0.589 | -207.480 | 365.418 |
| unit_R294 | -800.4005 | 149.798 | -5.343 | 0.000 | -1094.007 | -506.794 |
| unit_R295 | -1179.9310 | 169.657 | -6.955 | 0.000 | -1512.463 | -847.399 |
| unit_R300 | 488.0336 | 146.146 | 3.339 | 0.001 | 201.584 | 774.483 |
| unit_R303 | -503.7708 | 148.551 | -3.391 | 0.001 | -794.934 | -212.608 |
| unit_R304 | -645.3072 | 146.934 | -4.392 | 0.000 | -933.301 | -357.314 |
| unit_R307 | -1454.3298 | 154.751 | -9.398 | 0.000 | -1757.644 | -1151.016 |
| unit_R308 | -979.1828 | 151.601 | -6.459 | 0.000 | -1276.324 | -682.042 |
| unit_R309 | -943.8603 | 151.171 | -6.244 | 0.000 | -1240.158 | -647.563 |
| unit_R310 | -492.7003 | 152.909 | -3.222 | 0.001 | -792.404 | -192.996 |
| unit_R311 | -1356.7305 | 150.650 | -9.006 | 0.000 | -1652.008 | -1061.453 |
| unit_R312 | -1406.0248 | 147.332 | -9.543 | 0.000 | -1694.799 | -1117.251 |
| unit_R313 | -1747.1601 | 155.213 | -11.257 | 0.000 | -2051.381 | -1442.939 |
| unit_R318 | -1199.7076 | 147.735 | -8.121 | 0.000 | -1489.271 | -910.144 |
| unit_R319 | -396.9874 | 148.960 | -2.665 | 0.008 | -688.952 | -105.023 |
| unit_R321 | -649.3696 | 146.146 | -4.443 | 0.000 | -935.819 | -362.920 |
| unit_R322 | 30.9239 | 148.962 | 0.208 | 0.836 | -261.044 | 322.892 |
| unit_R323 | -497.0385 | 151.950 | -3.271 | 0.001 | -794.863 | -199.214 |
| unit_R325 | -1435.9524 | 150.749 | -9.525 | 0.000 | -1731.424 | -1140.481 |
| unit_R330 | -765.4439 | 149.794 | -5.110 | 0.000 | -1059.043 | -471.845 |
| unit_R335 | -1428.0644 | 155.802 | -9.166 | 0.000 | -1733.439 | -1122.690 |
| unit_R336 | -1782.2849 | 155.804 | -11.439 | 0.000 | -2087.663 | -1476.907 |
| unit_R337 | -1726.7415 | 153.940 | -11.217 | 0.000 | -2028.466 | -1425.017 |
| unit_R338 | -1853.5135 | 151.268 | -12.253 | 0.000 | -2150.001 | -1557.026 |
| unit_R341 | -1263.0971 | 147.523 | -8.562 | 0.000 | -1552.245 | -973.950 |
| unit_R344 | -1310.4959 | 155.589 | -8.423 | 0.000 | -1615.453 | -1005.538 |
| unit_R345 | -1297.5825 | 150.651 | -8.613 | 0.000 | -1592.861 | -1002.304 |

```
unit_R346         -516.5363    151.075     -3.419      0.001      -812.647  -220.426
unit_R348        -1660.5788    151.154    -10.986      0.000     -1956.844 -1364.314
unit_R354        -1621.9985    154.387    -10.506      0.000     -1924.600 -1319.397
unit_R356         -681.1600    150.413     -4.529      0.000      -975.973  -386.347
unit_R358        -1588.2745    154.383    -10.288      0.000     -1890.869 -1285.680
unit_R370        -1290.9864    150.216     -8.594      0.000     -1585.413  -996.560
unit_R371        -1075.1879    151.950     -7.076      0.000     -1373.012  -777.364
unit_R372        -1092.1437    153.289     -7.125      0.000     -1392.593  -791.695
unit_R373        -1144.8127    153.732     -7.447      0.000     -1446.131  -843.495
unit_R382         -888.3056    149.967     -5.923      0.000     -1182.245  -594.367
unit_R424        -1442.6820    155.141     -9.299      0.000     -1746.762 -1138.602
unit_R429         -768.6038    147.409     -5.214      0.000     -1057.529  -479.679
unit_R453          -12.8098    156.543     -0.082      0.935      -319.637   294.017
unit_R454        -1689.8554    154.199    -10.959      0.000     -1992.088 -1387.622
unit_R455        -1743.6190    153.748    -11.341      0.000     -2044.968 -1442.270
unit_R456        -1557.9066    149.800    -10.400      0.000     -1851.518 -1264.295
unit_R459        -1723.8581    212.426     -8.115      0.000     -2140.218 -1307.499
unit_R464        -1919.0066    153.449    -12.506      0.000     -2219.770 -1618.243
hour_0            -146.5138     22.942     -6.386      0.000      -191.482  -101.546
hour_4           -1262.6161     23.046    -54.787      0.000     -1307.786 -1217.446
hour_8            -933.0556     25.152    -37.096      0.000      -982.355  -883.757
hour_12           1556.9001     27.914     55.774      0.000      1502.188  1611.613
hour_16            853.3062     28.019     30.455      0.000       798.389   908.223
hour_20           1704.8326     23.671     72.021      0.000      1658.436  1751.229
day_week_0         201.5390     24.149      8.346      0.000       154.207   248.871
day_week_1         611.4333     26.223     23.317      0.000       560.036   662.831
day_week_2         637.1959     27.351     23.297      0.000       583.588   690.804
day_week_3         638.0988     26.061     24.485      0.000       587.019   689.179
day_week_4         609.2279     27.107     22.475      0.000       556.098   662.357
day_week_5        -286.9554     27.344    -10.494      0.000      -340.550  -233.361
day_week_6        -637.6860     24.220    -26.329      0.000      -685.157  -590.215
wspdi_group_0.0    407.5296     43.545      9.359      0.000       322.180   492.879
wspdi_group_5.0    420.9405     37.125     11.338      0.000       348.175   493.706
wspdi_group_10.0   439.0652     37.128     11.826      0.000       366.293   511.838
wspdi_group_15.0   447.9262     40.897     10.953      0.000       367.767   528.085
wspdi_group_20.0   425.9491     69.628      6.118      0.000       289.477   562.421
wspdi_group_25.0  -368.5571    166.255     -2.217      0.027      -694.421   -42.694
==============================================================================
Omnibus:                    30246.342   Durbin-Watson:                   1.525
Prob(Omnibus):                  0.000   Jarque-Bera (JB):          1103688.475
Skew:                           2.979   Prob(JB):                         0.00
Kurtosis:                      27.199   Cond. No.                     1.36e+17
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The smallest eigenvalue is 9.36e-27. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.
```

In [ ]: