In [1]:

```
# import necessary package
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import cross_val_score


%matplotlib inline
import warnings
warnings.filterwarnings('ignore')
```

In [2]:

```
# Load data
seattle_calendar = pd.read_csv("C:/Users/neera/OneDrive/Desktop/Pranali_Project/cal
seattle_listing  = pd.read_csv("C:/Users/neera/OneDrive/Desktop/Pranali_Project/lis
seattle_review   = pd.read_csv("C:/Users/neera/OneDrive/Desktop/Pranali_Project/revi
```

In [3]:

```
seattle_calendar.head()
```

Out[3]:

	listing_id	date	available	price
0	241032	2016-01-04	t	$85.00
1	241032	2016-01-05	t	$85.00
2	241032	2016-01-06	f	NaN
3	241032	2016-01-07	f	NaN
4	241032	2016-01-08	f	NaN

In [4]:

```
seattle_calendar.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1393570 entries, 0 to 1393569
Data columns (total 4 columns):
 #   Column      Non-Null Count    Dtype 
---  ------      --------------    ----- 
 0   listing_id  1393570 non-null  int64 
 1   date        1393570 non-null  object
 2   available   1393570 non-null  object
 3   price       934542 non-null   object
dtypes: int64(1), object(3)
memory usage: 42.5+ MB
```

In [5]:

```
#  If the available values are f, the price values seems to be NaN. But it is only
calendar_q1_df = seattle_calendar.groupby('available')['price'].count().reset_inde
calendar_q1_df.columns = ['available', 'price_nonnull_count']
calendar_q1_df
```

Out[5]:

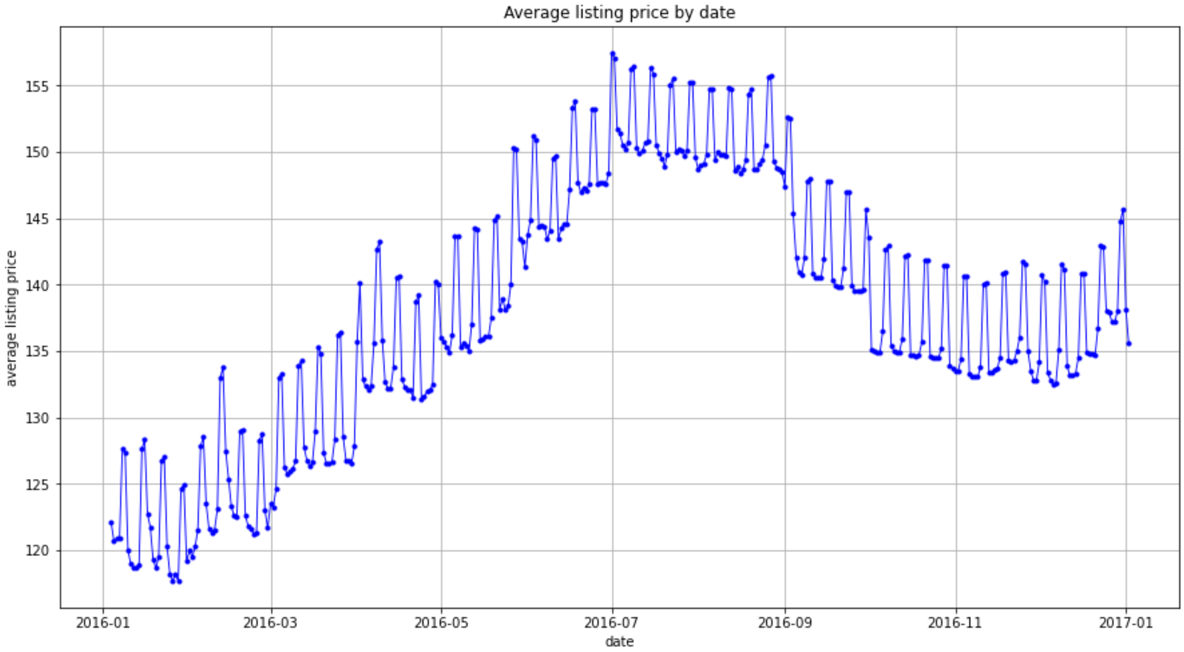
	available	price_nonnull_count
0	f	0
1	t	934542

In [6]:

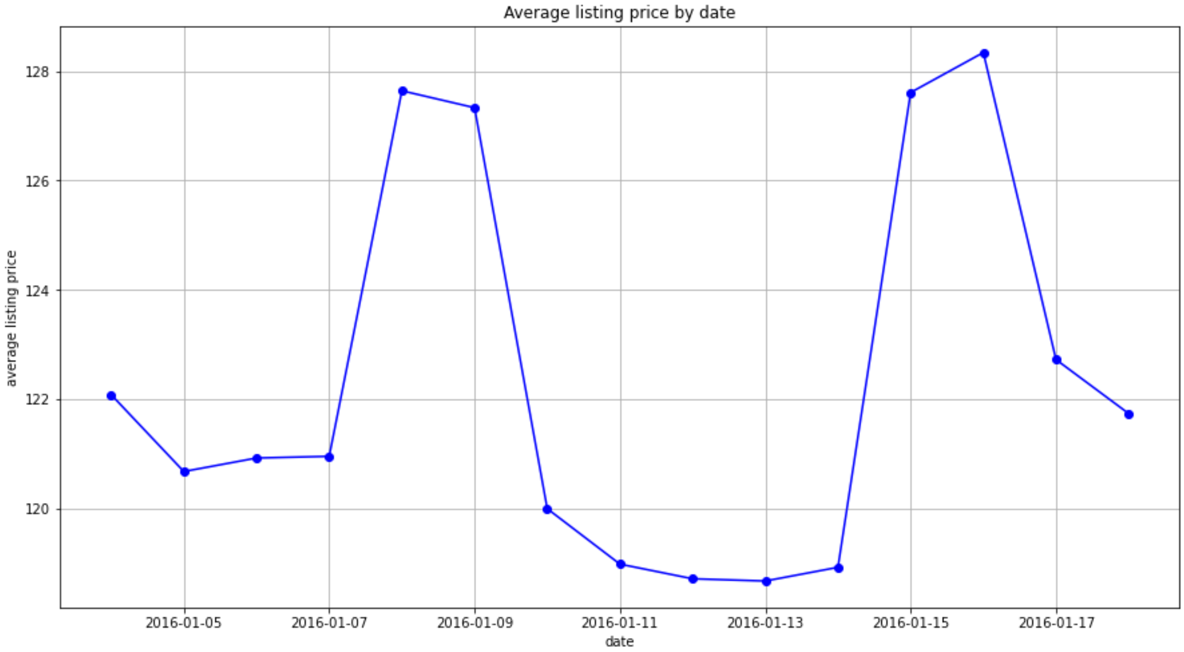
```
#  How many rows per each listing_id?
calendar_q2_df = seattle_calendar.groupby('listing_id')['date'].count().reset_inde
calendar_q2_df['date'].value_counts()
```

Out[6]:
```
365    3818
Name: date, dtype: int64
```

In [7]:
```
# process data
calendar_q3_df = seattle_calendar.copy(deep=True)
calendar_q3_df.dropna(inplace=True)
calendar_q3_df['date'] = pd.to_datetime(calendar_q3_df['date'])
calendar_q3_df['price'] = calendar_q3_df['price'].map(lambda x: float(x[1:].replac

# apply aggregation
calendar_q3_df = calendar_q3_df.groupby('date')['price'].mean().reset_index()

# plot avg listings prices over time.
plt.figure(figsize=(15, 8))
plt.plot(calendar_q3_df.date, calendar_q3_df.price, color='b', marker='.', linewid
plt.title("Average listing price by date")
plt.xlabel('date')
plt.ylabel('average listing price')
plt.grid()
```

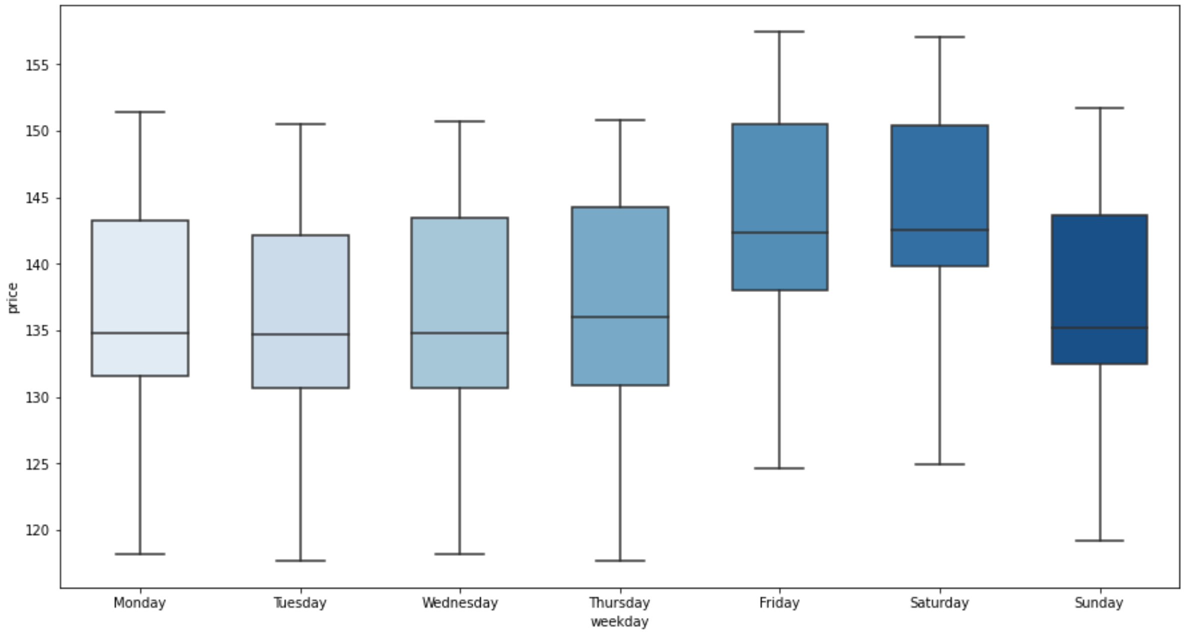


In [8]:
```
# plot more narrow range
plt.figure(figsize=(15, 8))
plt.plot(calendar_q3_df.date.values[:15], calendar_q3_df.price.values[:15], color=
plt.title("Average listing price by date")
plt.xlabel('date')
plt.ylabel('average listing price')
plt.grid()
```



In [9]:

```
# create weekday column
calendar_q3_df["weekday"] = calendar_q3_df["date"].dt.day_name()

# boxplot to see price distribution
plt.figure(figsize=(15, 8))
sns.boxplot(x = 'weekday',  y = 'price', data = calendar_q3_df, palette="Blues", wi
plt.show()
```



In [10]:

```
seattle_listing.head()
```

Out[10]:

	id	listing_url	scrape_id	last_scraped	name	sumı
0	241032	https://www.airbnb.com/rooms/241032	20160104002432	2016-01-04	Stylish Queen Anne Apartment	
1	953595	https://www.airbnb.com/rooms/953595	20160104002432	2016-01-04	Bright & Airy Queen Anne Apartment	Chem sens V rem the ir
2	3308979	https://www.airbnb.com/rooms/3308979	20160104002432	2016-01-04	New Modern House-Amazing water view	mc house in : Specta
3	7421966	https://www.airbnb.com/rooms/7421966	20160104002432	2016-01-04	Queen Anne Chateau	A char apart tha Q A
4	278830	https://www.airbnb.com/rooms/278830	20160104002432	2016-01-04	Charming craftsman 3 bdm house	Cozy f craf hou bea nei

5 rows × 92 columns

In [11]:

```
print(list(seattle_listing.columns.values))
```

```
['id', 'listing_url', 'scrape_id', 'last_scraped', 'name', 'summary', 'space', 'de
scription', 'experiences_offered', 'neighborhood_overview', 'notes', 'transit', 't
humbnail_url', 'medium_url', 'picture_url', 'xl_picture_url', 'host_id', 'host_ur
l', 'host_name', 'host_since', 'host_location', 'host_about', 'host_response_tim
e', 'host_response_rate', 'host_acceptance_rate', 'host_is_superhost', 'host_thumb
nail_url', 'host_picture_url', 'host_neighbourhood', 'host_listings_count', 'host_
total_listings_count', 'host_verifications', 'host_has_profile_pic', 'host_identit
y_verified', 'street', 'neighbourhood', 'neighbourhood_cleansed', 'neighbourhood_g
roup_cleansed', 'city', 'state', 'zipcode', 'market', 'smart_location', 'country_c
ode', 'country', 'latitude', 'longitude', 'is_location_exact', 'property_type', 'r
oom_type', 'accommodates', 'bathrooms', 'bedrooms', 'beds', 'bed_type', 'amenitie
s', 'square_feet', 'price', 'weekly_price', 'monthly_price', 'security_deposit',
'cleaning_fee', 'guests_included', 'extra_people', 'minimum_nights', 'maximum_nigh
ts', 'calendar_updated', 'has_availability', 'availability_30', 'availability_60',
'availability_90', 'availability_365', 'calendar_last_scraped', 'number_of_review
s', 'first_review', 'last_review', 'review_scores_rating', 'review_scores_accurac
y', 'review_scores_cleanliness', 'review_scores_checkin', 'review_scores_communica
tion', 'review_scores_location', 'review_scores_value', 'requires_license', 'licen
se', 'jurisdiction_names', 'instant_bookable', 'cancellation_policy', 'require_gue
st_profile_picture', 'require_guest_phone_verification', 'calculated_host_listings
_count', 'reviews_per_month']
```

In [12]:

```
print("Num of listings: ", seattle_listing.id.count())
print("Num of rows: ", seattle_listing.shape[0])
```

```
Num of listings:  3818
Num of rows:  3818
```

This shows the each rows represents unique listings.

In [13]:

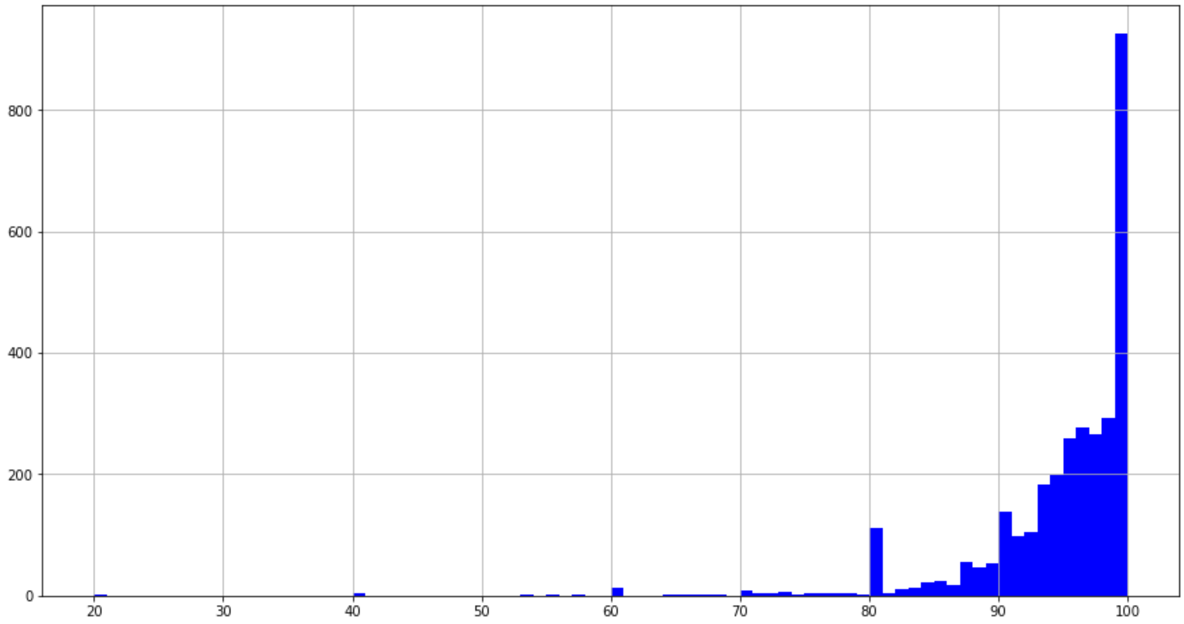
```
seattle_listing['review_scores_rating'].describe().reset_index()
```

Out[13]:

	index	review_scores_rating
0	count	3171.000000
1	mean	94.539262
2	std	6.606083
3	min	20.000000
4	25%	93.000000
5	50%	96.000000
6	75%	99.000000
7	max	100.000000

In [14]:

```
# cleaning data
listings_q1_df = seattle_listing['review_scores_rating'].dropna()

# plot histgram
plt.figure(figsize=(15, 8))
plt.hist(listings_q1_df.values, bins=80, color='b')
plt.grid()
```



In [15]:

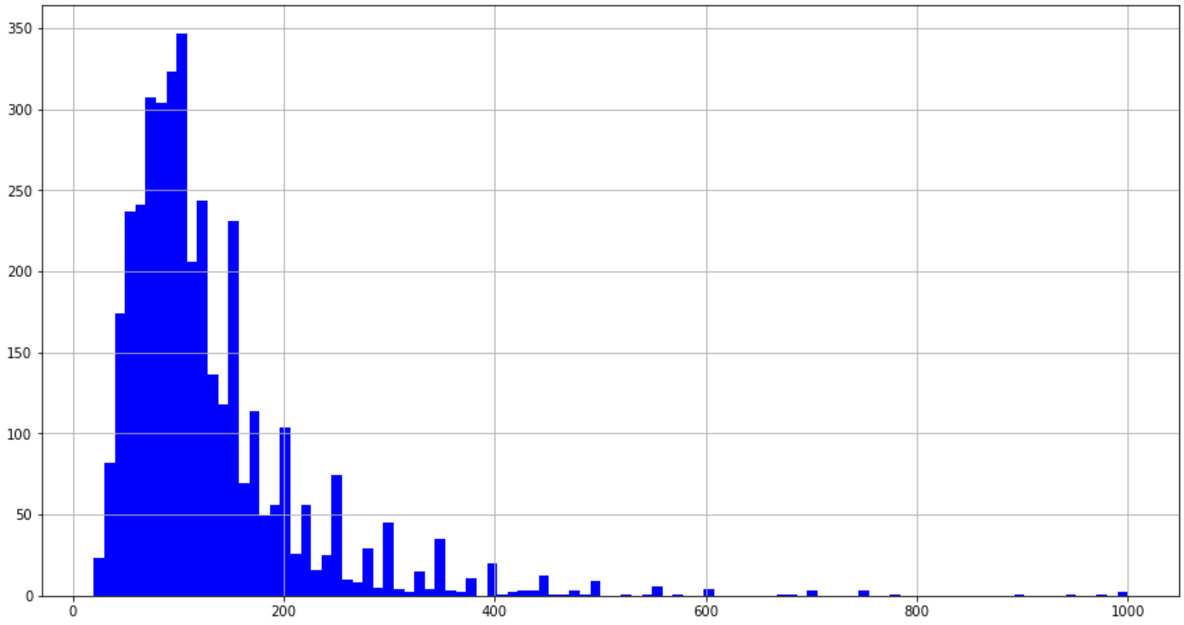
```
# cleaning data
listings_q2_df = seattle_listing.copy(deep=True)
listings_q2_df = listings_q2_df['price'].dropna().reset_index()
listings_q2_df['price'] = listings_q2_df['price'].map(lambda x: float(x[1:].replac

listings_q2_df['price'].describe().reset_index()
```

Out[15]:

	index	price
0	count	3818.000000
1	mean	127.976166
2	std	90.250022
3	min	20.000000
4	25%	75.000000
5	50%	100.000000
6	75%	150.000000
7	max	1000.000000

In [16]:

```
plt.figure(figsize=(15, 8))
plt.hist(listings_q2_df.price, bins=100, color='b')
plt.grid()
```



maximum_nights

In [17]:

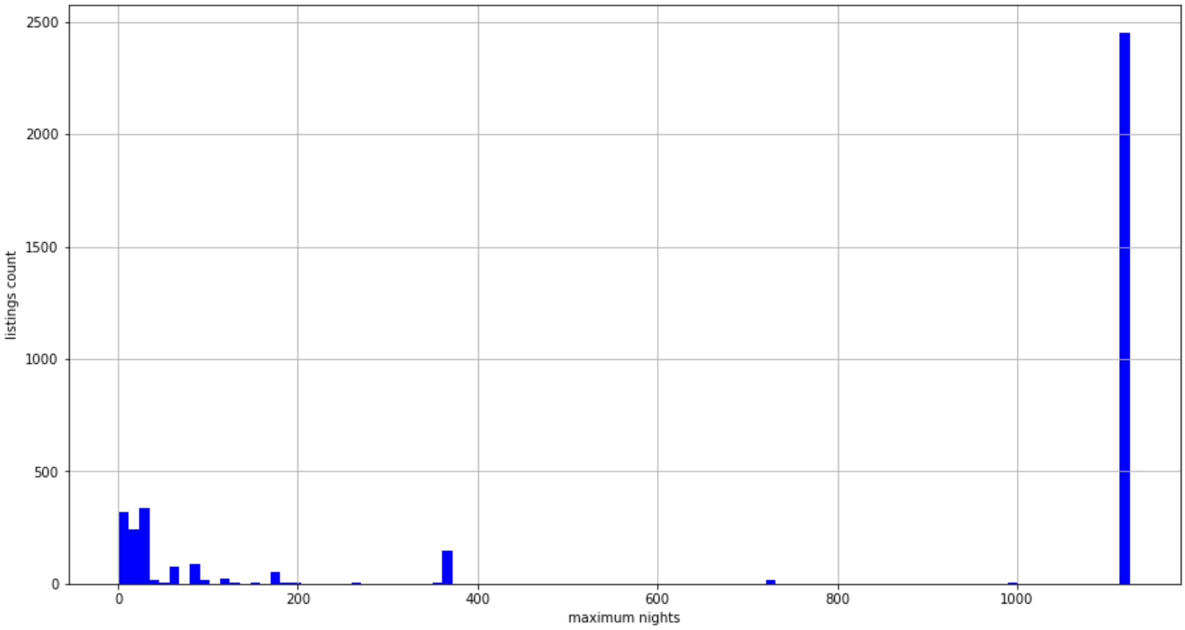
```
seattle_listing['maximum_nights'].describe().reset_index()
```

Out[17]:

	index	maximum_nights
0	count	3818.000000
1	mean	780.447617
2	std	1683.589007
3	min	1.000000
4	25%	60.000000
5	50%	1125.000000
6	75%	1125.000000
7	max	100000.000000

In [18]:

```
# eliminate outliers because maximum values are very large.
listings_q3_df = seattle_listing[seattle_listing['maximum_nights'] <= 1500]

plt.figure(figsize=(15, 8))
plt.hist(listings_q3_df.maximum_nights, bins=100, color='b')
plt.xlabel('maximum nights')
plt.ylabel('listings count')
plt.grid()
```



In [19]:

```
seattle_review.head()
```

Out[19]:

	listing_id	id	date	reviewer_id	reviewer_name	comments
0	7202016	38917982	2015-07-19	28943674	Bianca	Cute and cozy place. Perfect location to every...
1	7202016	39087409	2015-07-20	32440555	Frank	Kelly has a great room in a very central locat...
2	7202016	39820030	2015-07-26	37722850	Ian	Very spacious apartment, and in a great neighb...
3	7202016	40813543	2015-08-02	33671805	George	Close to Seattle Center and all it has to offe...
4	7202016	41986501	2015-08-10	34959538	Ming	Kelly was a great host and very accommodating ...

In [20]:

```
seattle_review.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 84849 entries, 0 to 84848
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype 
---  ------         --------------  ----- 
 0   listing_id     84849 non-null  int64 
 1   id             84849 non-null  int64 
 2   date           84849 non-null  object
 3   reviewer_id    84849 non-null  int64 
 4   reviewer_name  84849 non-null  object
 5   comments       84831 non-null  object
dtypes: int64(3), object(3)
memory usage: 3.9+ MB
```

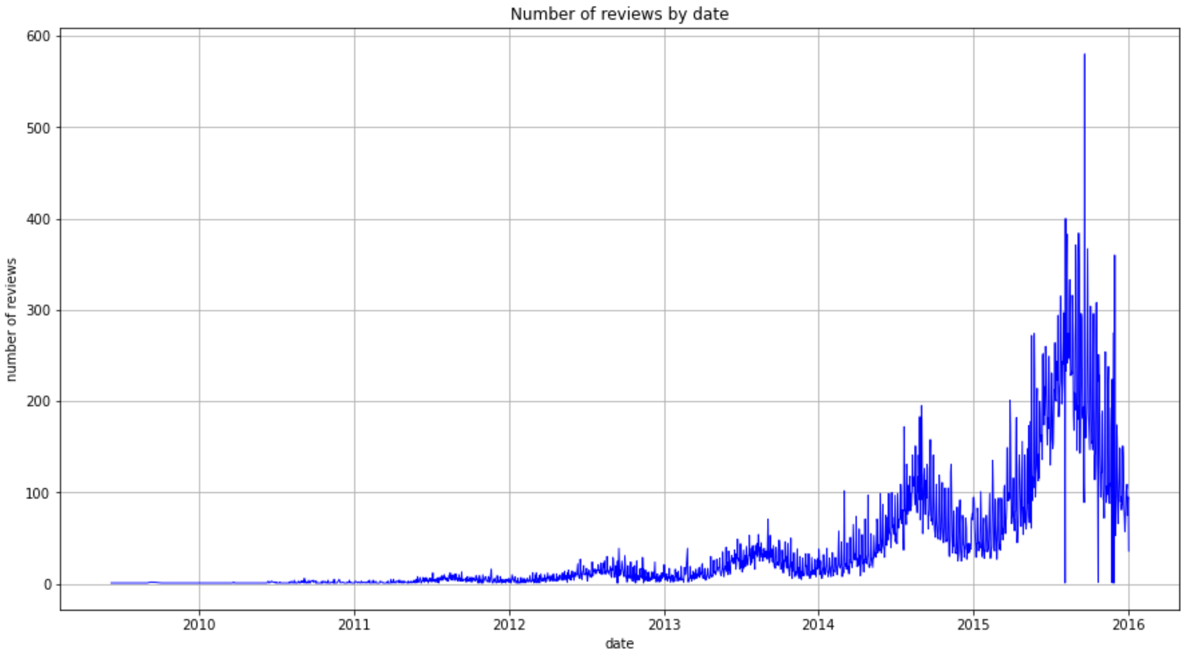
In [21]:

```
print("sample 1: ", seattle_review.comments.values[0], "\n")
print("sample 2: ", seattle_review.comments.values[3])
```

```
sample 1:  Cute and cozy place. Perfect location to everything!

sample 2:  Close to Seattle Center and all it has to offer - ballet, theater, muse
um, Space Needle, restaurants of all ilk just blocks away, and the Metropolitan (p
robably the coolest grocer you'll ever find). Easy to find and Kelly was warm, wel
coming, and really interesting to talk to.
```

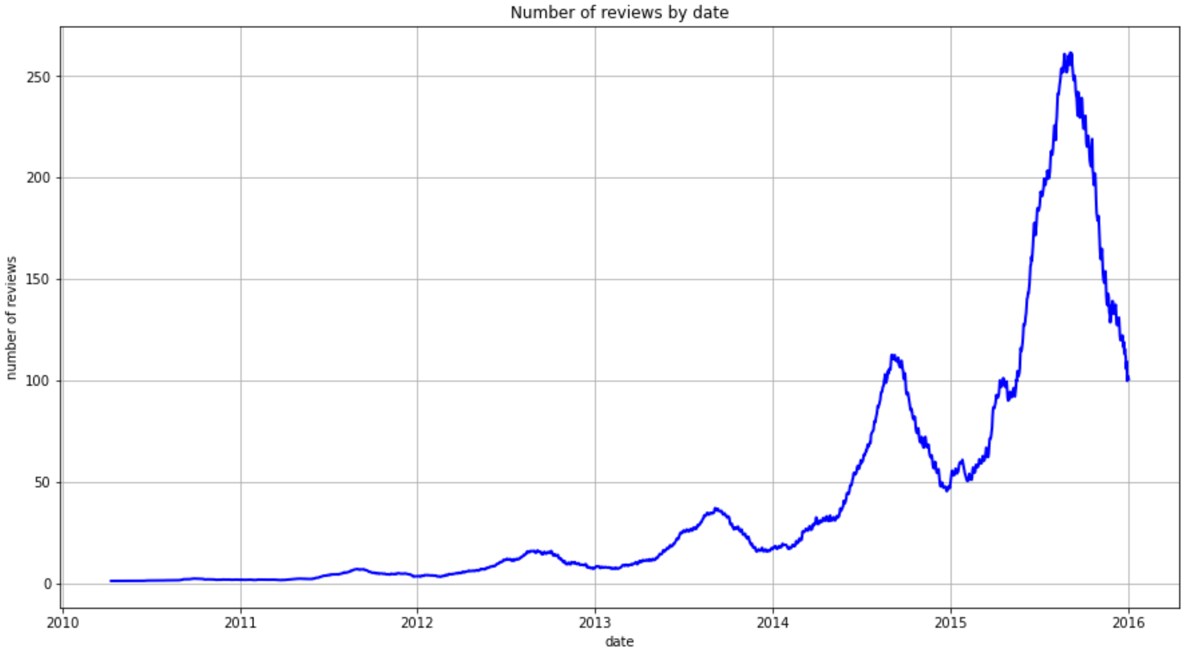
In [22]:

```
# convert date column's data type to date from object
review_q1_df = seattle_review.copy(deep=True)
review_q1_df.date = pd.to_datetime(review_q1_df.date)

review_q1_df = review_q1_df.groupby('date')['id'].count().reset_index()

# plot avg listings prices over time.
plt.figure(figsize=(15, 8))
plt.plot(review_q1_df.date, review_q1_df.id, color='b', linewidth=0.9)
plt.title("Number of reviews by date")
plt.xlabel('date')
plt.ylabel('number of reviews')
plt.grid()
```



In [23]:

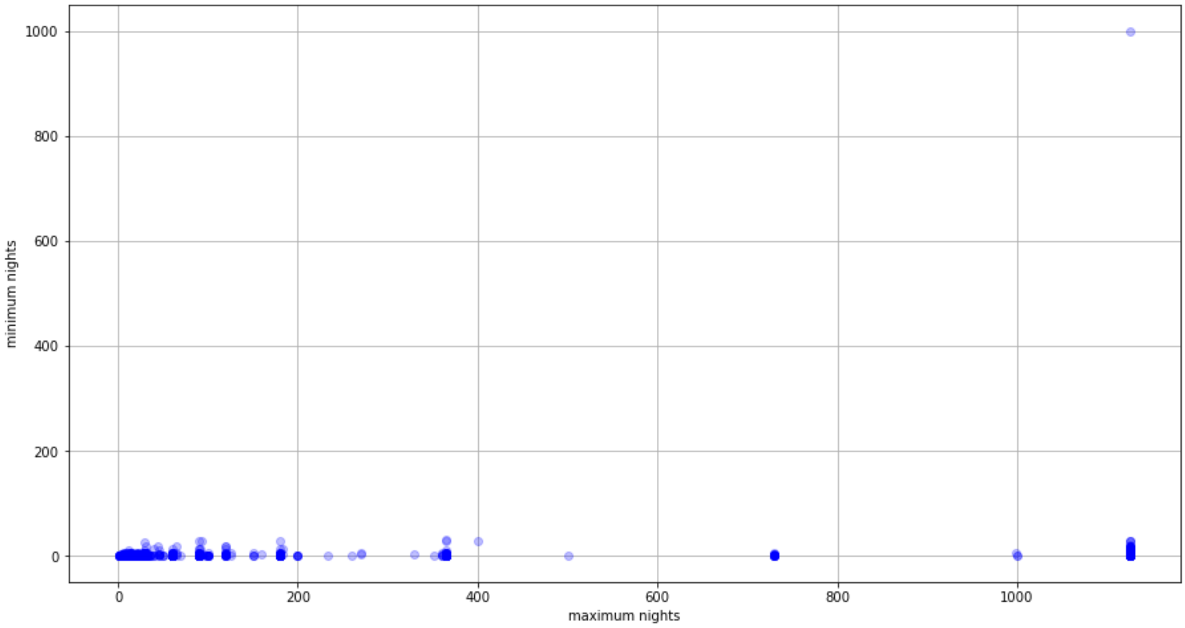
```
# create rolling mean column
review_q1_df["rolling_mean_30"] = review_q1_df.id.rolling(window=30).mean()

# plot avg listings prices over time.
plt.figure(figsize=(15, 8))
plt.plot(review_q1_df.date, review_q1_df.rolling_mean_30, color='b', linewidth=2.0
plt.title("Number of reviews by date")
plt.xlabel('date')
plt.ylabel('number of reviews')
plt.grid()
```



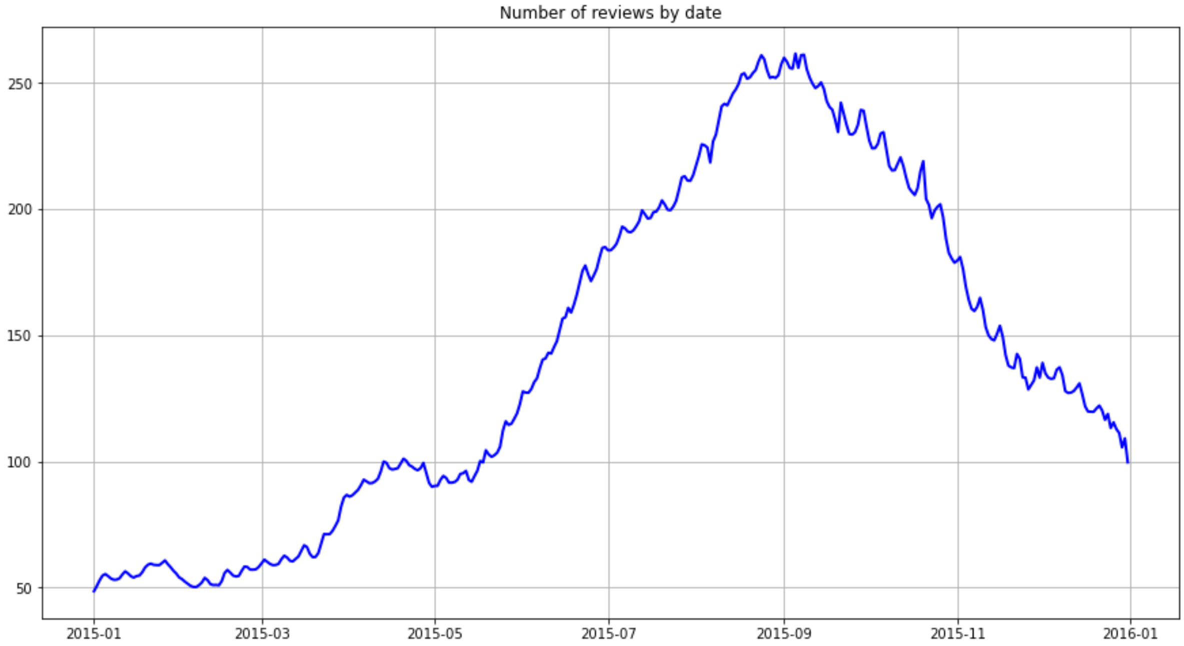
In [24]:
```
review_q1_df["year"] = review_q1_df.date.dt.year
years = review_q1_df.year.unique()

for year in years:
    if year >= 2010 and year < 2016:
        year_df = review_q1_df[review_q1_df.year == year]
        max_value = year_df.rolling_mean_30.max()
        max_date = year_df[year_df.rolling_mean_30 == max_value].date.dt.date.valu
        print(year, max_date, np.round(max_value, 1))
```

```
2010 2010-10-04 2.3
2011 2011-08-31 7.0
2012 2012-09-04 16.2
2013 2013-09-04 37.0
2014 2014-09-03 112.6
2015 2015-09-05 261.6
```

In [25]:
```
listings_q3_df["min_max_night_diff"] = listings_q3_df.maximum_nights - listings_q3_

plt.figure(figsize=(15, 8))
plt.plot(listings_q3_df.maximum_nights, listings_q3_df.minimum_nights, color='b', 
plt.xlabel('maximum nights')
plt.ylabel('minimum nights')
plt.grid()
```



In [26]:
```
review_q2_df = review_q1_df[review_q1_df.year == 2015]

plt.figure(figsize=(15, 8))
plt.plot(review_q2_df.date, review_q2_df.rolling_mean_30, color='b', linewidth=2.0
plt.title("Number of reviews by date")
plt.grid()
```



In [27]:
```
prepare_df = seattle_listing.copy(deep=True)
```

In [28]:
```
# check null count
df_length = prepare_df.shape[0]

for col in prepare_df.columns:
    null_count = prepare_df[col].isnull().sum()
    if null_count == 0:
        continue

    null_ratio = np.round(null_count/df_length * 100, 2)
    print("{} has {} null values ({}%)".format(col, null_count, null_ratio))
```

```
summary has 177 null values (4.64%)
space has 569 null values (14.9%)
neighborhood_overview has 1032 null values (27.03%)
notes has 1606 null values (42.06%)
transit has 934 null values (24.46%)
thumbnail_url has 320 null values (8.38%)
medium_url has 320 null values (8.38%)
xl_picture_url has 320 null values (8.38%)
host_name has 2 null values (0.05%)
host_since has 2 null values (0.05%)
host_location has 8 null values (0.21%)
host_about has 859 null values (22.5%)
host_response_time has 523 null values (13.7%)
host_response_rate has 523 null values (13.7%)
host_acceptance_rate has 773 null values (20.25%)
host_is_superhost has 2 null values (0.05%)
host_thumbnail_url has 2 null values (0.05%)
host_picture_url has 2 null values (0.05%)
host_neighbourhood has 300 null values (7.86%)
host_listings_count has 2 null values (0.05%)
host_total_listings_count has 2 null values (0.05%)
host_has_profile_pic has 2 null values (0.05%)
host_identity_verified has 2 null values (0.05%)
neighbourhood has 416 null values (10.9%)
zipcode has 7 null values (0.18%)
property_type has 1 null values (0.03%)
bathrooms has 16 null values (0.42%)
bedrooms has 6 null values (0.16%)
beds has 1 null values (0.03%)
square_feet has 3721 null values (97.46%)
weekly_price has 1809 null values (47.38%)
monthly_price has 2301 null values (60.27%)
security_deposit has 1952 null values (51.13%)
cleaning_fee has 1030 null values (26.98%)
first_review has 627 null values (16.42%)
last_review has 627 null values (16.42%)
review_scores_rating has 647 null values (16.95%)
review_scores_accuracy has 658 null values (17.23%)
review_scores_cleanliness has 653 null values (17.1%)
review_scores_checkin has 658 null values (17.23%)
review_scores_communication has 651 null values (17.05%)
review_scores_location has 655 null values (17.16%)
review_scores_value has 656 null values (17.18%)
license has 3818 null values (100.0%)
reviews_per_month has 627 null values (16.42%)
```

In [29]:

```
# detect need drop columns
drop_cols = [col for col in prepare_df.columns if prepare_df[col].isnull().sum()/d

# drop null
prepare_df.drop(drop_cols, axis=1, inplace=True)
prepare_df.dropna(subset=['host_since'], inplace=True)

# check after
for col in prepare_df.columns:
    null_count = prepare_df[col].isnull().sum()
    if null_count == 0:
        continue

    null_ratio = np.round(null_count/df_length * 100, 2)
    print("{} has {} null values ({}%)".format(col, null_count, null_ratio))
```

```
summary has 177 null values (4.64%)
space has 568 null values (14.88%)
neighborhood_overview has 1031 null values (27.0%)
transit has 933 null values (24.44%)
thumbnail_url has 320 null values (8.38%)
medium_url has 320 null values (8.38%)
xl_picture_url has 320 null values (8.38%)
host_location has 6 null values (0.16%)
host_about has 857 null values (22.45%)
host_response_time has 521 null values (13.65%)
host_response_rate has 521 null values (13.65%)
host_acceptance_rate has 771 null values (20.19%)
host_neighbourhood has 298 null values (7.81%)
neighbourhood has 416 null values (10.9%)
zipcode has 7 null values (0.18%)
property_type has 1 null values (0.03%)
bathrooms has 16 null values (0.42%)
bedrooms has 6 null values (0.16%)
beds has 1 null values (0.03%)
cleaning_fee has 1029 null values (26.95%)
first_review has 625 null values (16.37%)
last_review has 625 null values (16.37%)
review_scores_rating has 645 null values (16.89%)
review_scores_accuracy has 656 null values (17.18%)
review_scores_cleanliness has 651 null values (17.05%)
review_scores_checkin has 656 null values (17.18%)
review_scores_communication has 649 null values (17.0%)
review_scores_location has 653 null values (17.1%)
review_scores_value has 654 null values (17.13%)
reviews_per_month has 625 null values (16.37%)
```

In [30]:

```python
drop_cols = ['listing_url', 'scrape_id', 'last_scraped', 'name', 'summary', 'space
                'transit', 'medium_url', 'picture_url', 'xl_picture_url', 'host_id
                'host_picture_url', 'street', 'city', 'state', 'zipcode', 'market'
                'calendar_updated', 'calendar_last_scraped', 'first_review', 'last_

prepare_df.drop(drop_cols, axis=1, inplace=True)
```

In [31]:

```python
prepare_df.columns
```

Out[31]:

```
Index(['id', 'experiences_offered', 'thumbnail_url', 'host_since',
       'host_location', 'host_response_time', 'host_response_rate',
       'host_acceptance_rate', 'host_is_superhost', 'host_neighbourhood',
       'host_listings_count', 'host_total_listings_count',
       'host_has_profile_pic', 'host_identity_verified', 'neighbourhood',
       'neighbourhood_cleansed', 'neighbourhood_group_cleansed',
       'is_location_exact', 'property_type', 'room_type', 'accommodates',
       'bathrooms', 'bedrooms', 'beds', 'bed_type', 'price', 'cleaning_fee',
       'guests_included', 'extra_people', 'minimum_nights', 'maximum_nights',
       'has_availability', 'availability_30', 'availability_60',
       'availability_90', 'availability_365', 'number_of_reviews',
       'review_scores_rating', 'review_scores_accuracy',
       'review_scores_cleanliness', 'review_scores_checkin',
       'review_scores_communication', 'review_scores_location',
       'review_scores_value', 'requires_license', 'jurisdiction_names',
       'instant_bookable', 'cancellation_policy',
       'require_guest_profile_picture', 'require_guest_phone_verification',
       'calculated_host_listings_count', 'reviews_per_month'],
      dtype='object')
```

In [32]:

```python
drop_cols = []
for col in prepare_df.columns:
    if prepare_df[col].nunique() == 1:
        drop_cols.append(col)
```

```
prepare_df.drop(drop_cols, axis=1, inplace=True)
prepare_df.columns
```

Out[32]:

```
Index(['id', 'thumbnail_url', 'host_since', 'host_location',
       'host_response_time', 'host_response_rate', 'host_acceptance_rate',
       'host_is_superhost', 'host_neighbourhood', 'host_listings_count',
       'host_total_listings_count', 'host_has_profile_pic',
       'host_identity_verified', 'neighbourhood', 'neighbourhood_cleansed',
       'neighbourhood_group_cleansed', 'is_location_exact', 'property_type',
       'room_type', 'accommodates', 'bathrooms', 'bedrooms', 'beds',
       'bed_type', 'price', 'cleaning_fee', 'guests_included', 'extra_people',
       'minimum_nights', 'maximum_nights', 'availability_30',
       'availability_60', 'availability_90', 'availability_365',
       'number_of_reviews', 'review_scores_rating', 'review_scores_accuracy',
       'review_scores_cleanliness', 'review_scores_checkin',
       'review_scores_communication', 'review_scores_location',
       'review_scores_value', 'instant_bookable', 'cancellation_policy',
       'require_guest_profile_picture', 'require_guest_phone_verification',
       'calculated_host_listings_count', 'reviews_per_month'],
      dtype='object')
```

In [33]:

```
# available days count each listings
listing_avalilable = seattle_calendar.groupby('listing_id')['price'].count().reset_
listing_avalilable.columns = ["id", "available_count"]

# merge
prepare_df = prepare_df.merge(listing_avalilable, how='left', on='id')

# create target column
prepare_df['host_since_year'] = pd.to_datetime(prepare_df['host_since']).dt.year
prepare_df["easily_accomodated"] = prepare_df.accommodates / (prepare_df.available_
```

In [34]:

```
print("Before: {} columns".format(prepare_df.shape[1]))

drop_cols = ['host_since', 'accommodates', 'availability_30', 'availability_60', ''
                'number_of_reviews', 'review_scores_rating', 'available_count', 'r

prepare_df.drop(drop_cols, axis=1, inplace=True)
print("After: {} columns".format(prepare_df.shape[1]))
```

```
Before: 51 columns
After: 39 columns
```

In [35]:

```
# convert true or false value to 1 or 0
dummy_cols = ['host_is_superhost', 'require_guest_phone_verification', 'require_gu
               'host_has_profile_pic', 'host_identity_verified', 'is_location_exact

for col in dummy_cols:
    prepare_df[col] = prepare_df[col].map(lambda x: 1 if x == 't' else 0)

# create dummy valuables
dummy_cols = ['host_location', 'host_neighbourhood', 'neighbourhood', 'neighbourho
              'property_type', 'room_type', 'bed_type', 'cancellation_policy', 'hos

prepare_df = pd.get_dummies(prepare_df, columns=dummy_cols, dummy_na=True)
```

In [36]:

```
df_length = prepare_df.shape[0]

for col in prepare_df.columns:
    null_count = prepare_df[col].isnull().sum()
    if null_count == 0:
        continue
    null_ratio = np.round(null_count/df_length * 100, 2)
    print("{} has {} null values ({}%)".format(col, null_count, null_ratio))
```

```
thumbnail_url has 320 null values (8.39%)
host_response_rate has 521 null values (13.65%)
host_acceptance_rate has 771 null values (20.2%)
bathrooms has 16 null values (0.42%)
bedrooms has 6 null values (0.16%)
beds has 1 null values (0.03%)
cleaning_fee has 1029 null values (26.97%)
review_scores_accuracy has 656 null values (17.19%)
review_scores_cleanliness has 651 null values (17.06%)
review_scores_checkin has 656 null values (17.19%)
review_scores_communication has 649 null values (17.01%)
review_scores_location has 653 null values (17.11%)
```

In [37]:

```
prepare_df["is_thumbnail_setted"] = 1 - prepare_df.thumbnail_url.isnull()
prepare_df.drop('thumbnail_url', axis=1, inplace=True)
prepare_df.host_response_rate = prepare_df.host_response_rate.fillna('0%').map(lamb
prepare_df.host_acceptance_rate = prepare_df.host_acceptance_rate.fillna('0%').map
prepare_df.bathrooms.fillna(0, inplace=True)
prepare_df.bedrooms.fillna(0, inplace=True)
prepare_df.beds.fillna(0, inplace=True)
prepare_df.cleaning_fee.fillna('$0', inplace=True)
prepare_df.review_scores_accuracy.fillna(0, inplace=True)
prepare_df.review_scores_cleanliness.fillna(0, inplace=True)
prepare_df.review_scores_checkin.fillna(0, inplace=True)
prepare_df.review_scores_communication.fillna(0, inplace=True)
prepare_df.review_scores_location.fillna(0, inplace=True)
```

In [38]:

```
for col in prepare_df.columns:
    if prepare_df[col].dtypes == 'object':
        print(col)
```

```
price
cleaning_fee
extra_people
```

In [39]:

```
prepare_df.price = prepare_df.price.map(lambda x: float(x[1:].replace(',', '')))
prepare_df.cleaning_fee = prepare_df.cleaning_fee.map(lambda x: float(x[1:].replac
prepare_df.extra_people = prepare_df.extra_people.map(lambda x: float(x[1:].replac
```

In [40]:

```
X = prepare_df.drop(['id', 'easily_accomodated'], axis=1)
y = prepare_df.easily_accomodated.values

rf = RandomForestRegressor(n_estimators=100, max_depth=5)
scores = cross_val_score(rf, X, y, cv=5)
```

In [41]:

```
scores
```

Out[41]:

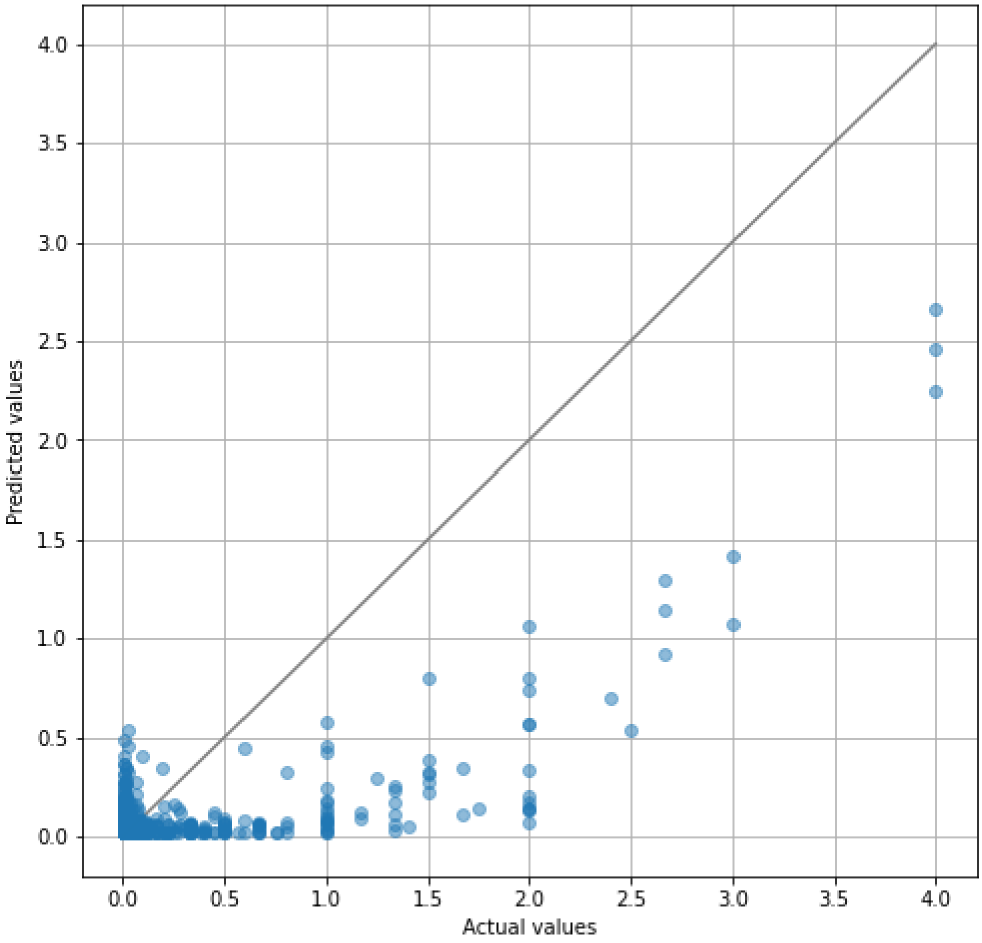
```
array([-0.11895504,  0.08684807,  0.0775067 , -0.01070099,  0.15766708])
```

In [42]:

```
rf.fit(X, y)
predictions = rf.predict(X)
```

In [43]:

```
plt.figure(figsize=(8, 8))

plt.plot((0, 4), (0, 4), color='gray')
plt.plot(y, predictions, linewidth=0, marker='o', alpha=0.5)
plt.grid()
plt.xlim((-0.2, 4.2))
plt.ylim((-0.2, 4.2))
plt.xlabel("Actual values")
plt.ylabel("Predicted values")
plt.show()
```



In [44]:

```
X = prepare_df.drop(['id', 'easily_accomodated'], axis=1)
y = np.log(prepare_df.easily_accomodated.values)

rf = RandomForestRegressor(n_estimators=100, max_depth=5)
scores = cross_val_score(rf, X, y, cv=5)
print(scores)
```

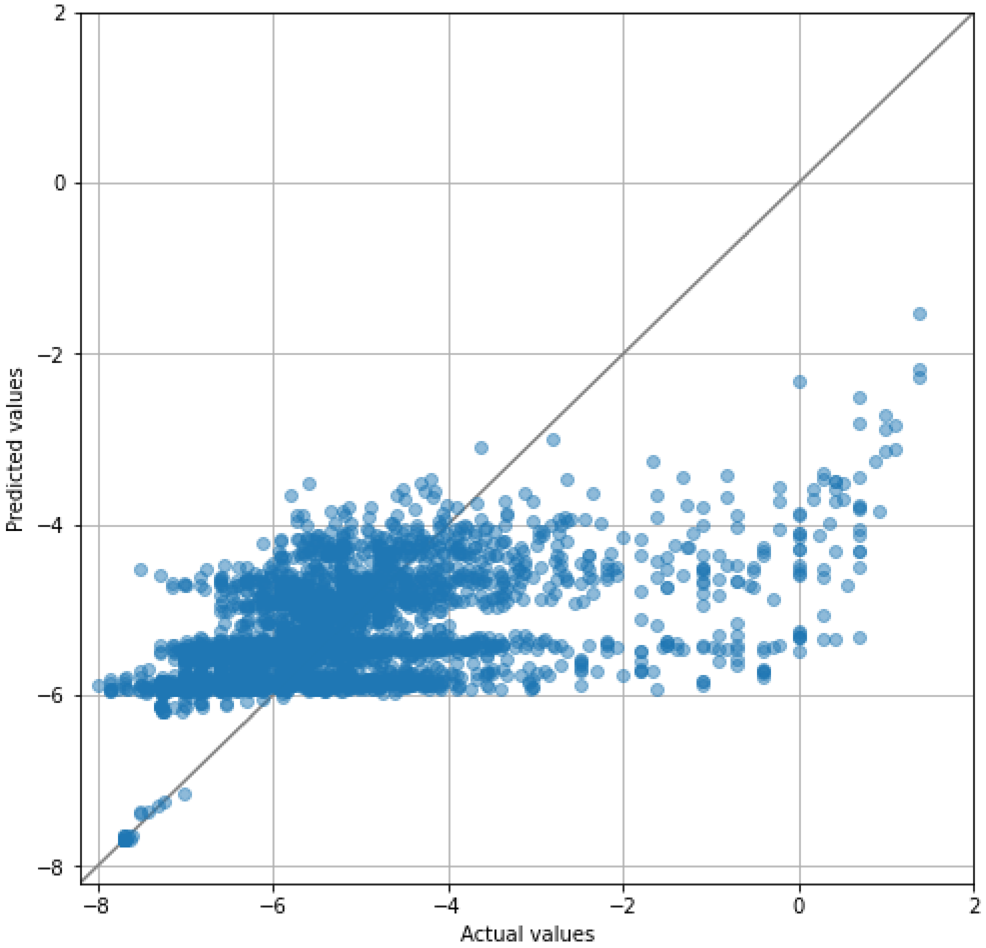
```
[0.2495328  0.13892357 0.17834679 0.10671301 0.20360773]
```

In [45]:

```
rf.fit(X, y)
predictions = rf.predict(X)
```

In [46]:

```
plt.figure(figsize=(8, 8))

plt.plot((-10, 10), (-10, 10), color='gray')
plt.plot(y, predictions, linewidth=0, marker='o', alpha=0.5)
plt.grid()
plt.xlim((-8.2, 2))
plt.ylim((-8.2, 2))
plt.xlabel("Actual values")
plt.ylabel("Predicted values")
plt.show()
```



```
In [ ]:
```