Drill 3.3

July 4, 2018

In [1]:
```
# Import all libraries needed
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# Enable inline plotting
%matplotlib inline
```

0.1 Choose one variable and plot that variable four different ways.

In [2]:
```
# creat df ftom listings file
listings = pd.read_csv('listings.csv')
```

In [3]:
```
# print(listings.head(1))
listings.columns
```

Out[3]:
```
Index(['id', 'listing_url', 'scrape_id', 'last_scraped', 'name', 'summary',
       'space', 'description', 'experiences_offered', 'neighborhood_overview',
       'notes', 'transit', 'access', 'interaction', 'house_rules',
       'thumbnail_url', 'medium_url', 'picture_url', 'xl_picture_url',
       'host_id', 'host_url', 'host_name', 'host_since', 'host_location',
       'host_about', 'host_response_time', 'host_response_rate',
       'host_acceptance_rate', 'host_is_superhost', 'host_thumbnail_url',
       'host_picture_url', 'host_neighbourhood', 'host_listings_count',
       'host_total_listings_count', 'host_verifications',
       'host_has_profile_pic', 'host_identity_verified', 'street',
       'neighbourhood', 'neighbourhood_cleansed',
       'neighbourhood_group_cleansed', 'city', 'state', 'zipcode', 'market',
       'smart_location', 'country_code', 'country', 'latitude', 'longitude',
       'is_location_exact', 'property_type', 'room_type', 'accommodates',
       'bathrooms', 'bedrooms', 'beds', 'bed_type', 'amenities', 'square_feet',
       'price', 'weekly_price', 'monthly_price', 'security_deposit',
       'cleaning_fee', 'guests_included', 'extra_people', 'minimum_nights',
       'maximum_nights', 'calendar_updated', 'has_availability',
       'availability_30', 'availability_60', 'availability_90',
       'availability_365', 'calendar_last_scraped', 'number_of_reviews',
       'first_review', 'last_review', 'review_scores_rating',
```

```
       'review_scores_accuracy', 'review_scores_cleanliness',
       'review_scores_checkin', 'review_scores_communication',
       'review_scores_location', 'review_scores_value', 'requires_license',
       'license', 'jurisdiction_names', 'instant_bookable',
       'is_business_travel_ready', 'cancellation_policy',
       'require_guest_profile_picture', 'require_guest_phone_verification',
       'calculated_host_listings_count', 'reviews_per_month'],
      dtype='object')
```

In [20]:
```
# create df to focus on price
# get series for columns I want

Listing_ID = listings['id']
rawprice = listings['price']
neighborhood = listings['neighbourhood_cleansed']
zipcode = listings['zipcode']
propertytype = listings['property_type']
roomtype = listings['room_type']
accommodates = listings['accommodates']
avail_60 = listings['availability_60']

# put series together info df, then manipulating to make price a float
price_df = pd.concat({'Listing_ID':Listing_ID, 'rawprice':rawprice, 'neighborhood':neig
price_df['cleaned_price'] = price_df['rawprice'][1:]
price_df['cleaned_price'] = price_df['cleaned_price'].str.replace('$', '')
price_df['cleaned_price'] = price_df['cleaned_price'].str.replace(',', '')
price_df['cleaned_price'] = price_df['cleaned_price'].astype(float)
#price_df.info()
price_df = price_df.dropna()
#price_df.info()
#print(price_df.iloc[7, 2][1:])
price_df.describe()
#price_df.head(10)
#price_df[price_df['cleaned_price'] > 1000]
```

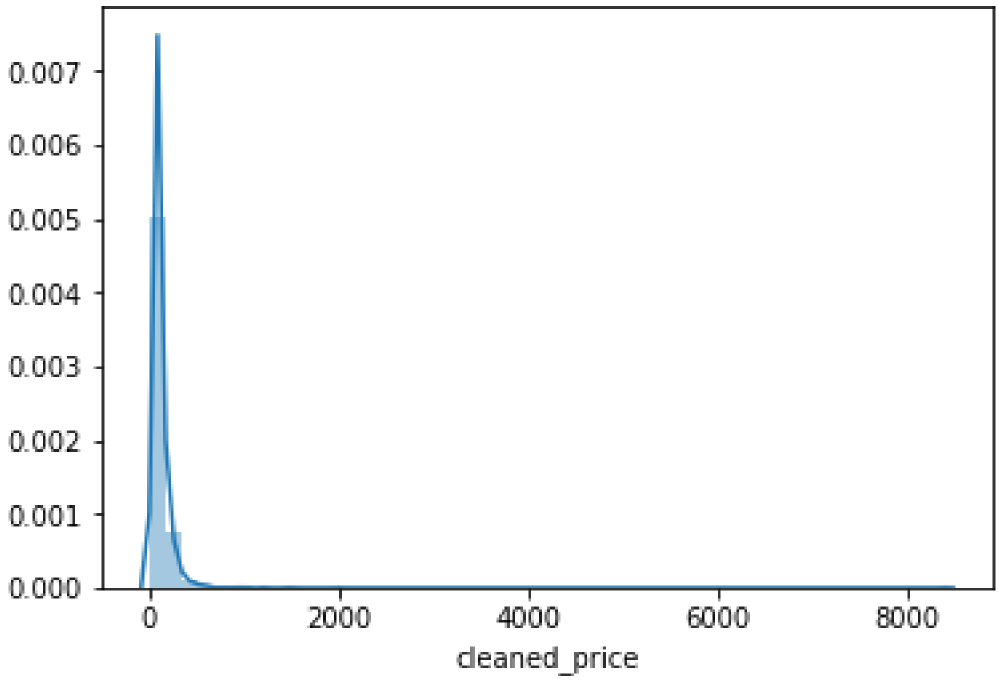
Out[20]:

	Listing_ID	accommodates	avail_60	zipcode	cleaned_price
count	4.730000e+03	4730.000000	4730.000000	4730.000000	4730.000000
mean	1.337572e+07	3.436998	20.866173	97214.051163	118.108879
std	7.372953e+06	1.890966	18.905751	10.912652	156.680858
min	1.289900e+04	1.000000	0.000000	97086.000000	0.000000
25%	6.938836e+06	2.000000	1.000000	97209.000000	65.000000
50%	1.433840e+07	3.000000	17.000000	97212.000000	90.000000
75%	2.001462e+07	4.000000	36.000000	97217.000000	130.000000
max	2.505202e+07	26.000000	60.000000	97403.000000	8400.000000

0.1.1 Histogram of Prices

In [12]:
```
sns.distplot(price_df['cleaned_price'])
```

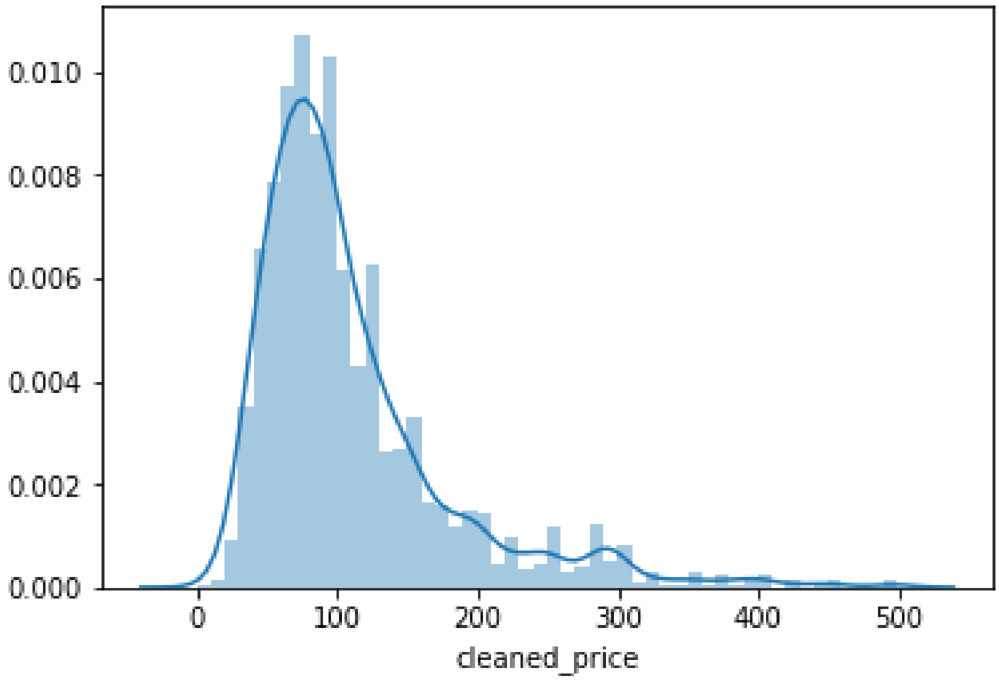
Out[12]: <matplotlib.axes._subplots.AxesSubplot at 0x11739c668>



Uh oh, I've found a problem in my SQL AirBnB assignment. I capped out at $999.

```
In [51]: drop_500 = price_df[price_df['cleaned_price'] < 500]
         sns.distplot(drop_500['cleaned_price'])
```

Out[51]: <matplotlib.axes._subplots.AxesSubplot at 0x118364518>

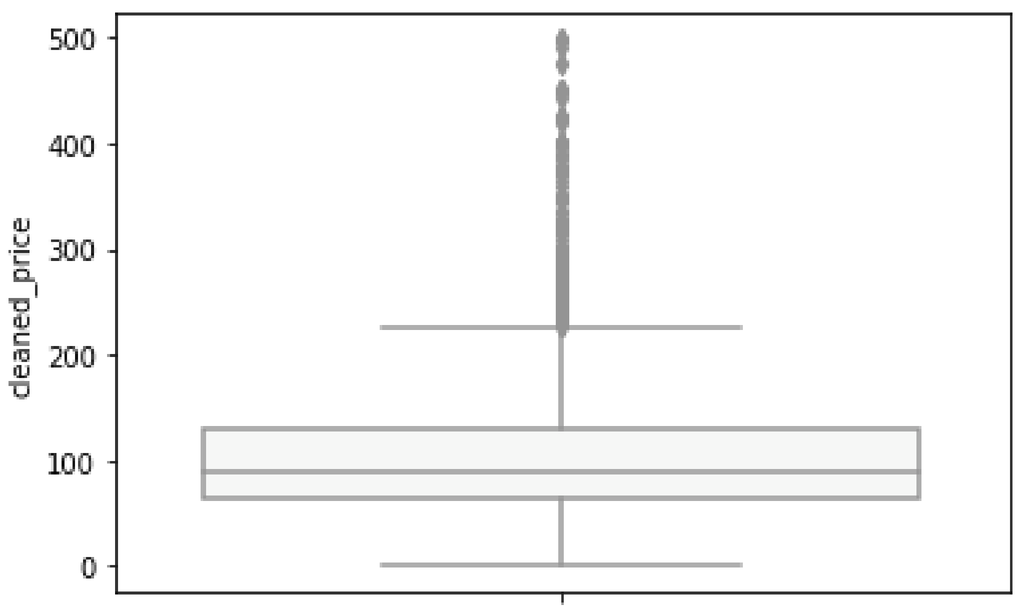


That's better. Confirmed on AirBnb that listing is not actually $8400 per night. Final chart actually shows listings < $500. My SQL problem still exists though.

0.1.2 Boxplot

```
In [52]: sns.boxplot(y='cleaned_price',data=drop_500, palette="PRGn")
```

```
Out[52]: <matplotlib.axes._subplots.AxesSubplot at 0x11898ff98>
```



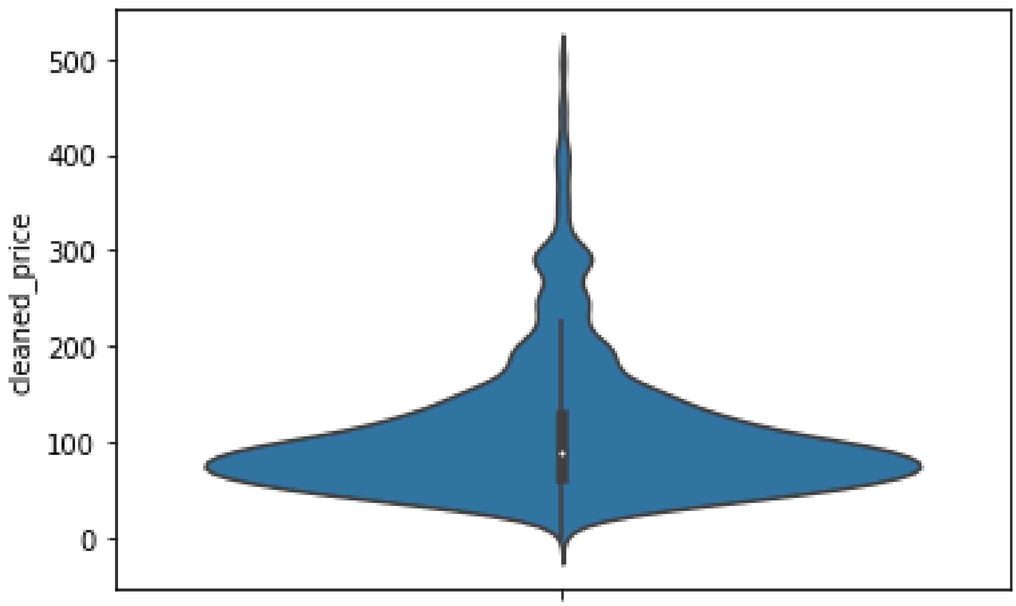
This is super ugly, but it does show that the vast majority of listings are < $200, with many outliers on the high end.

0.1.3 Violin plot

This result is similar to the boxplot, but the visualization gives a better feel for the amount at the different prices.

```
In [53]: sns.violinplot(y="cleaned_price", data=drop_500, split=True)
```

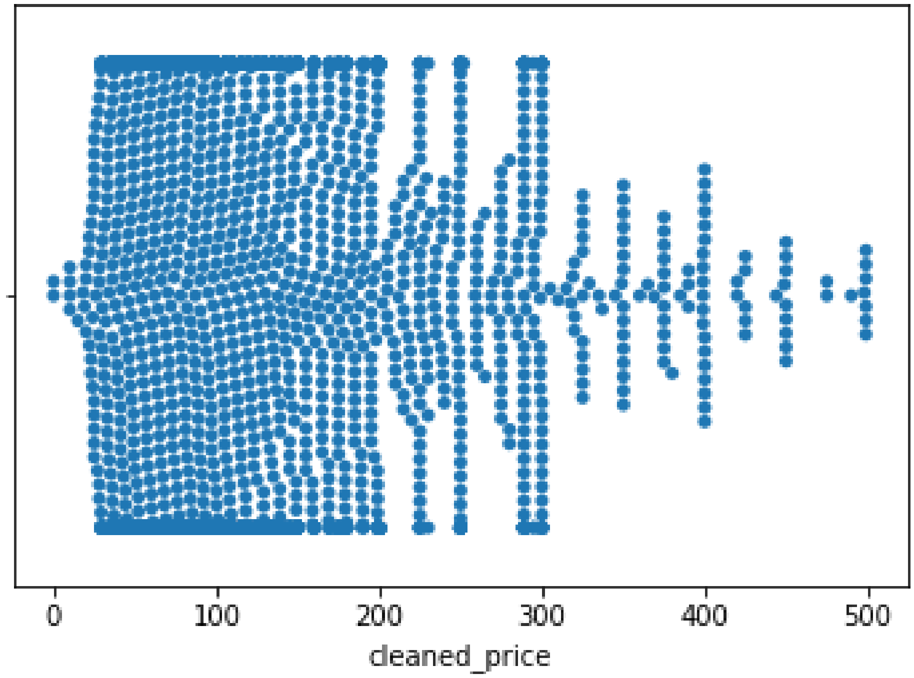
```
Out[53]: <matplotlib.axes._subplots.AxesSubplot at 0x1185b46a0>
```



0.1.4 Swarmplot

I wanted to see the swarmplot for prices. As you can see below, the large number of values make this less useful because there are too many points to fit into the chart.

```
In [54]: sns.swarmplot(x="cleaned_price", data=drop_500);
```



0.2 Choose two continuous variables, and plot them three different ways

For this section, I looked at another data set, "Multidimensional Poverty Measures" from the Oxford Poverty & Human Development Initiative. Specifically, I looked at the Intensity of Deprivation numbers of Urban vs Rural in the countries of the report. These numbers are the average amount below the poverty line those listed as poor are in their respective areas.

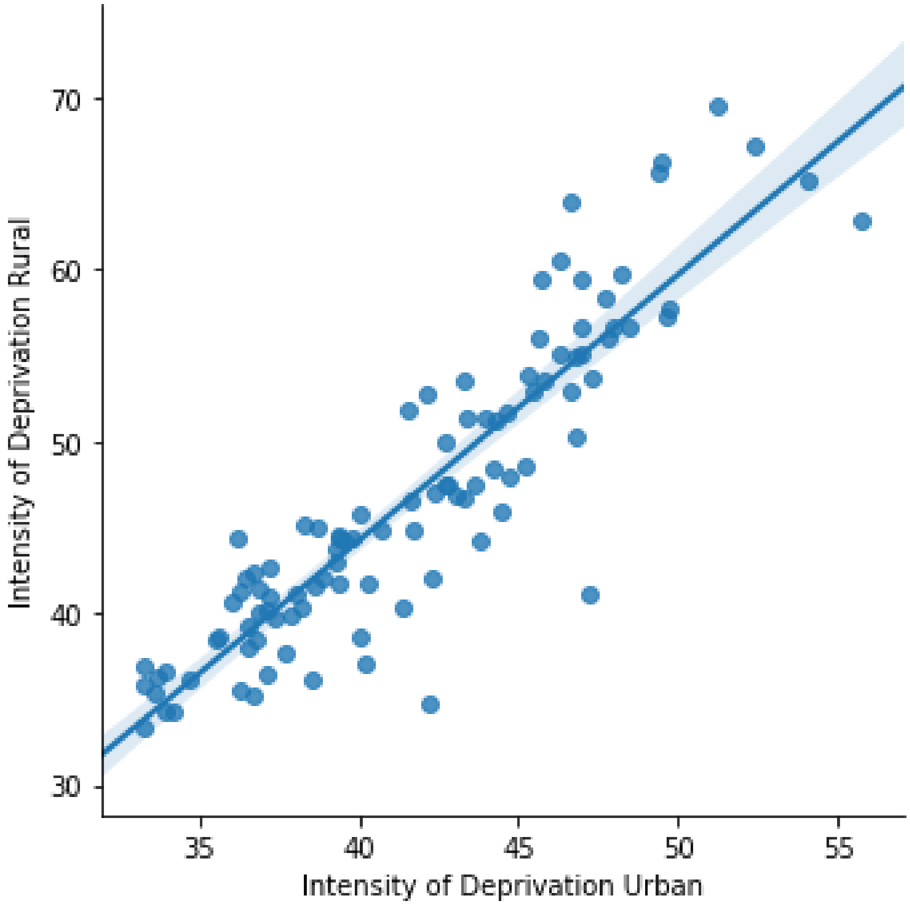
```
In [105]: #creating dataframe
          pov_df = pd.read_csv('MPI_national.csv')
          pov_df.head(1)

Out[105]:    ISO     Country  MPI Urban  Headcount Ratio Urban  \
          0  KAZ  Kazakhstan        0.0                    0.0

             Intensity of Deprivation Urban  MPI Rural  Headcount Ratio Rural  \
          0                            33.3        0.0                   0.09

             Intensity of Deprivation Rural
          0                            33.3
```

0.2.1 Scatterplot showing Rural vs. Urban

This shows us a few things. First, a country's average urban poor are very poor, so are the rural poor. It isn't likely that a country will have desperately poor urban poor and not too bad off rural poor. Second, the rural poor get much poorer than the urban poor.

```
In [112]: sns.lmplot(x="Intensity of Deprivation Urban", y="Intensity of Deprivation Rural", dat
```

```
Out[112]: <seaborn.axisgrid.FacetGrid at 0x1180e91d0>
```

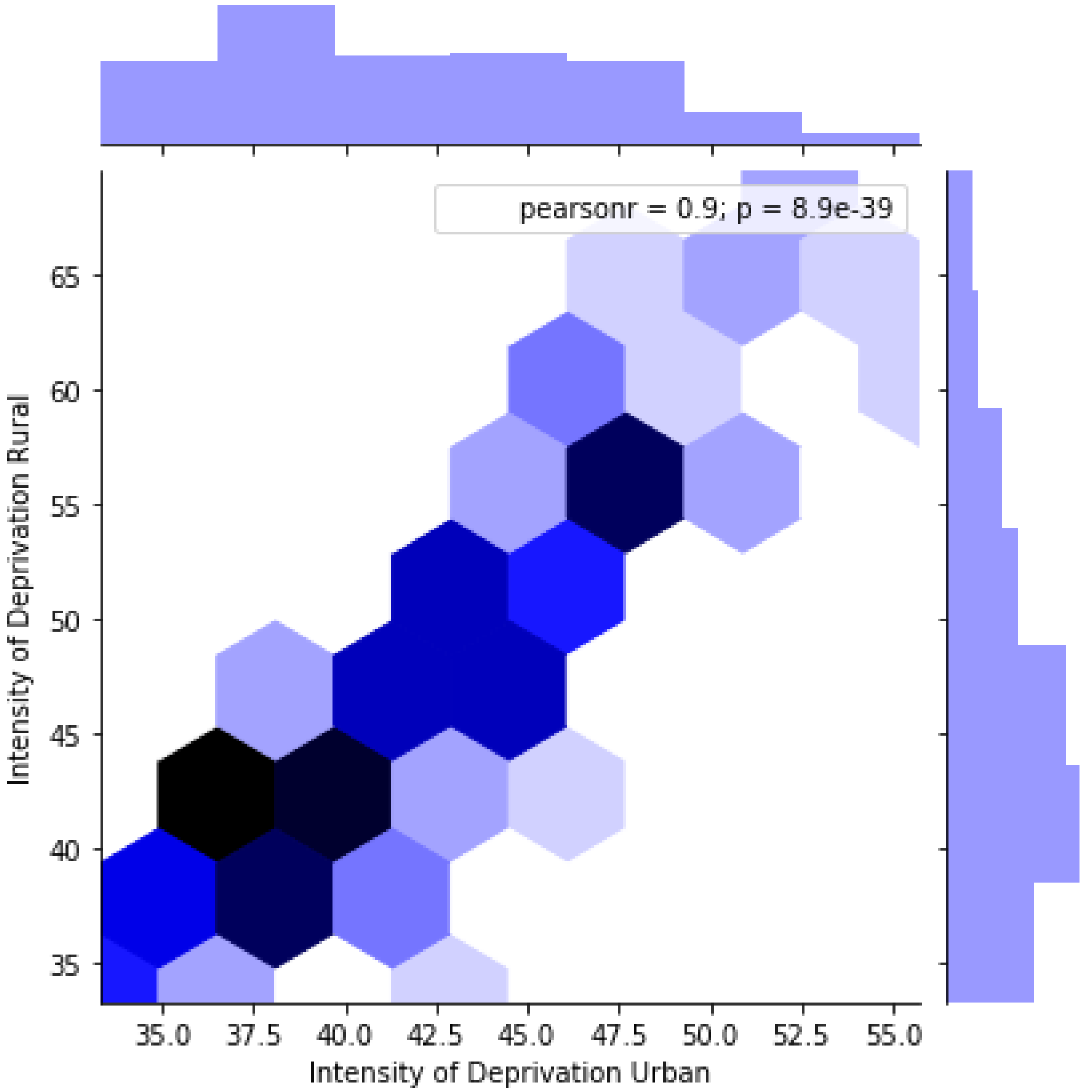


0.2.2 Hexbin comparison

Here we see a similar result, but the colors allow us to see intersections that are more dense, with the jointplot also providing a histogram for the counts of the two variables.

```
In [119]: sns.jointplot(x="Intensity of Deprivation Urban", y="Intensity of Deprivation Rural",
```

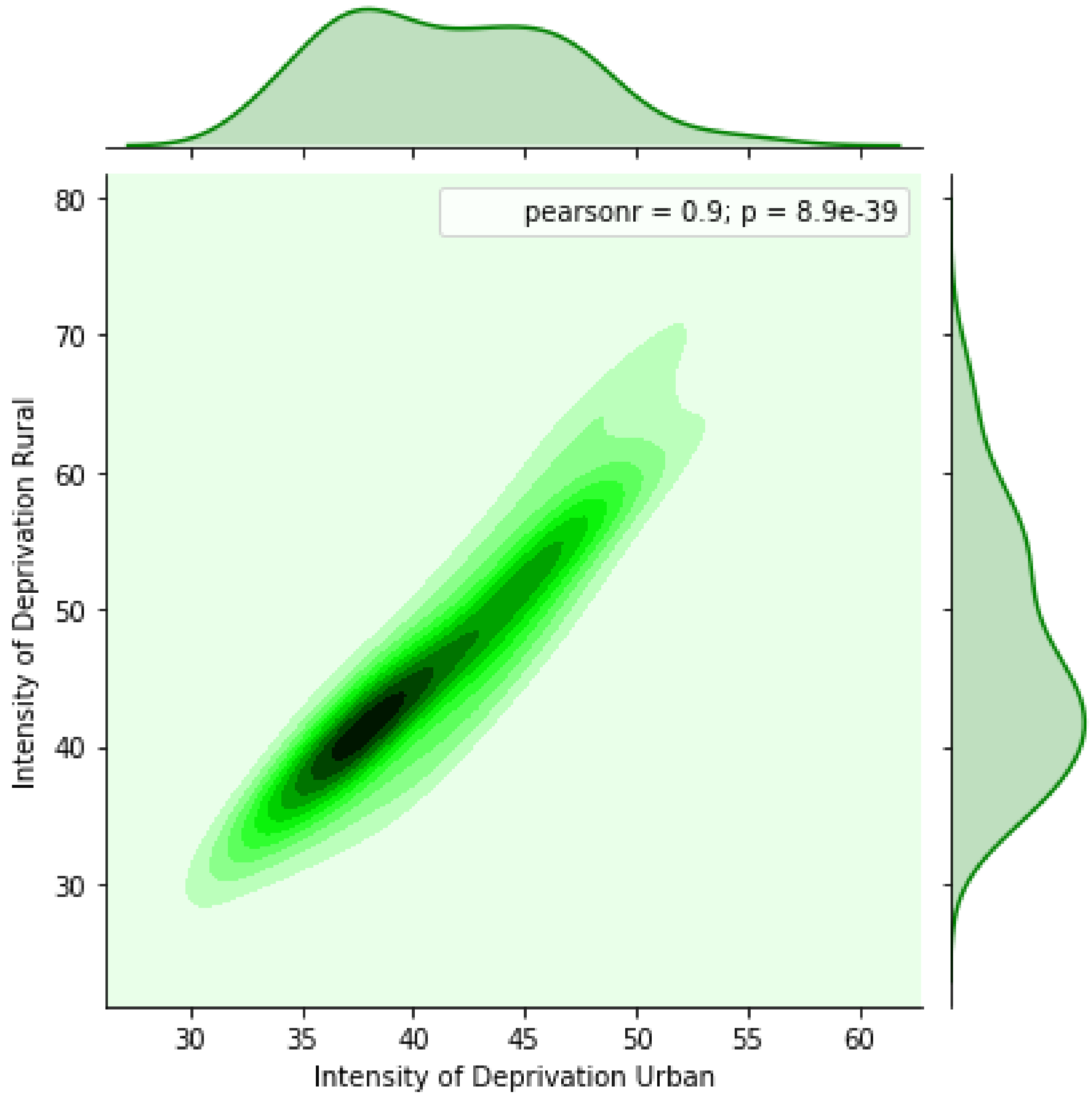
```
Out[119]: <seaborn.axisgrid.JointGrid at 0x11d86d048>
```



0.2.3 Kernal Density Estimation

Once again, this type of allows us to see the density of different intersections, but with smooth contours.

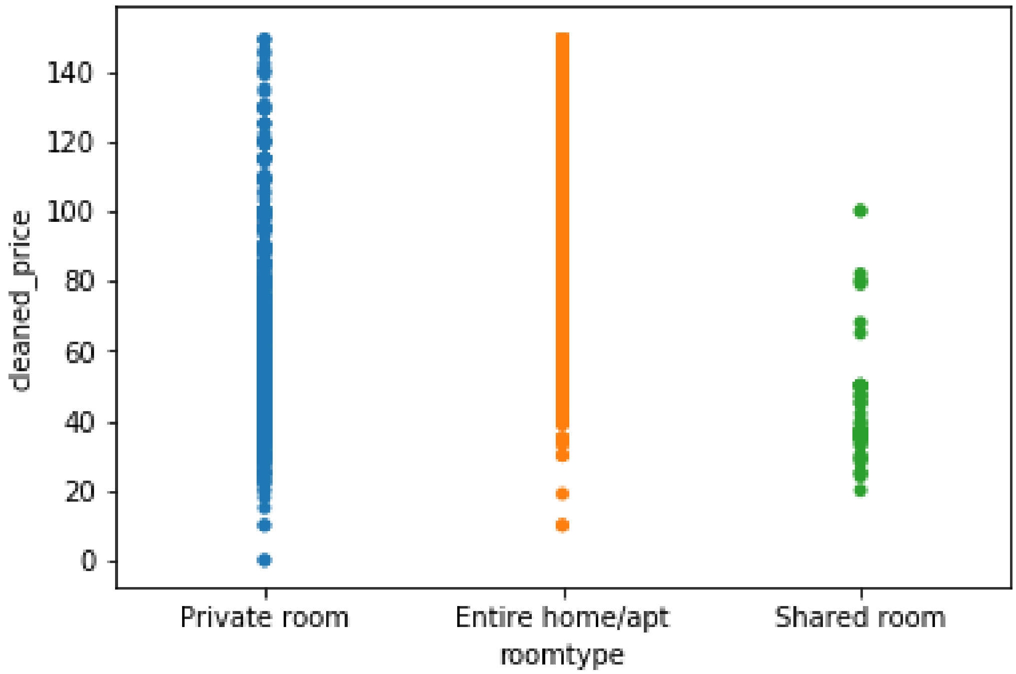
```
In [117]: sns.jointplot(x="Intensity of Deprivation Urban", y="Intensity of Deprivation Rural",

Out[117]: <seaborn.axisgrid.JointGrid at 0x11a23b320>
```



0.3 Choose one continuous variable and one categorical variable, and plot them six different ways.

0.3.1 Stripplot of roomtype and price

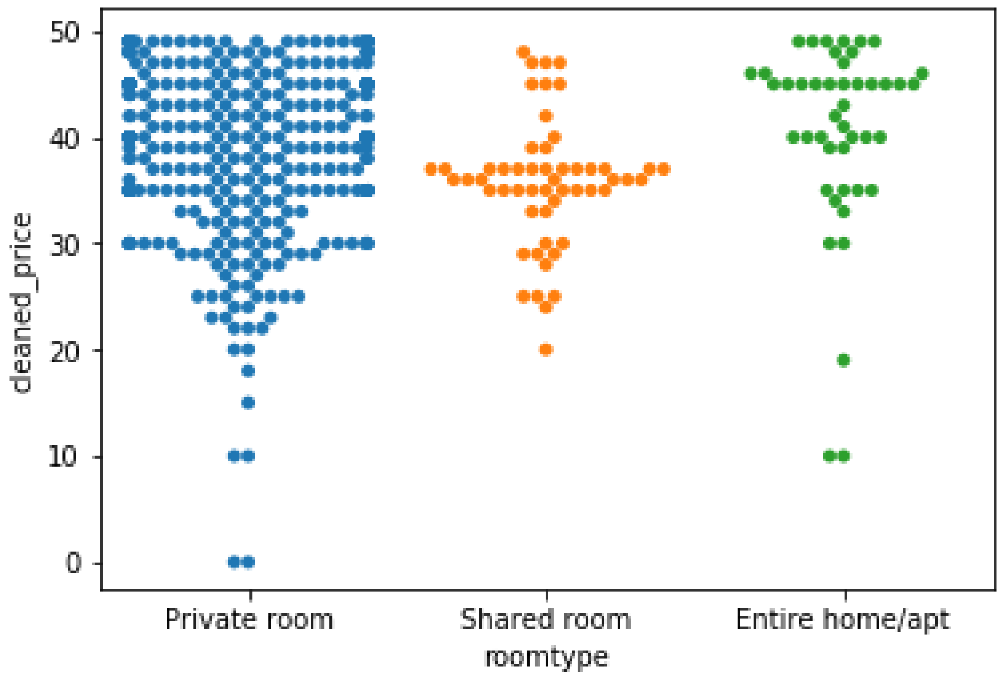
In [58]:
```
drop_150= drop_500[drop_500['cleaned_price']<150]
sns.stripplot(x="roomtype", y="cleaned_price", data=drop_150);
```



0.3.2 Swarmplot of roomtype and price

I wanted to look at the different comparisons on a swarmplot. However, The immediate issue was that the large number of data point caused the same crowding issue as in the single value chart in question 1. To at least get a feel for how the chart works, I only looked at those listings with a price less than $50.

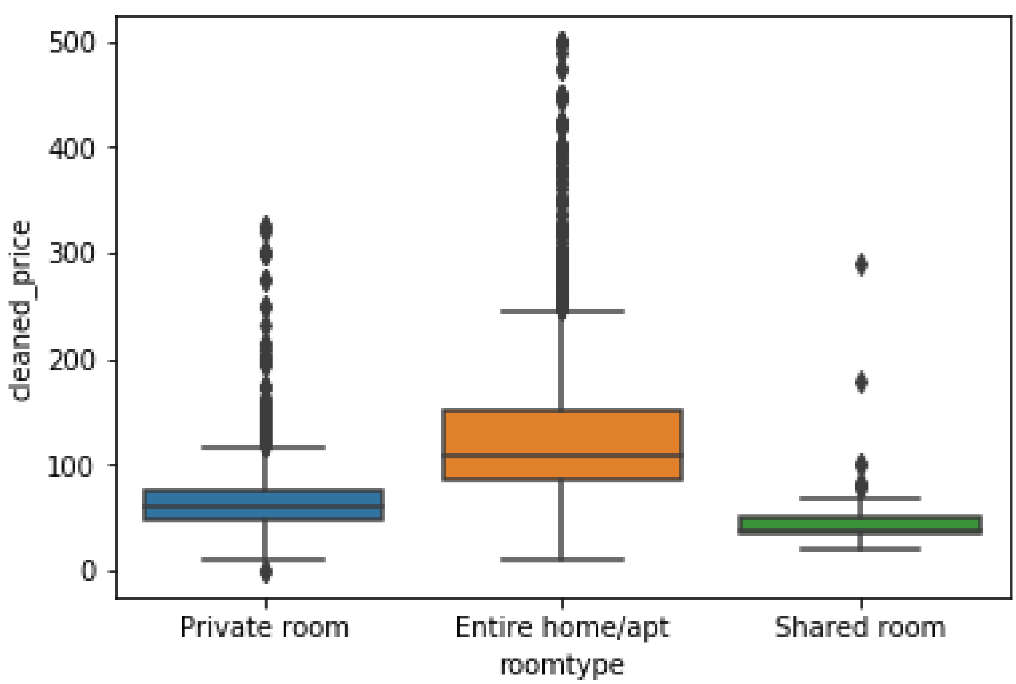
```
In [59]: drop_50 = drop_150[drop_150['cleaned_price'] < 50]
         sns.swarmplot(x="roomtype", y="cleaned_price", data=drop_50);
```



0.3.3 Boxpplot of roomtype and price

Another look at the data with boxplots. I again limited the listings, this time to < $500.

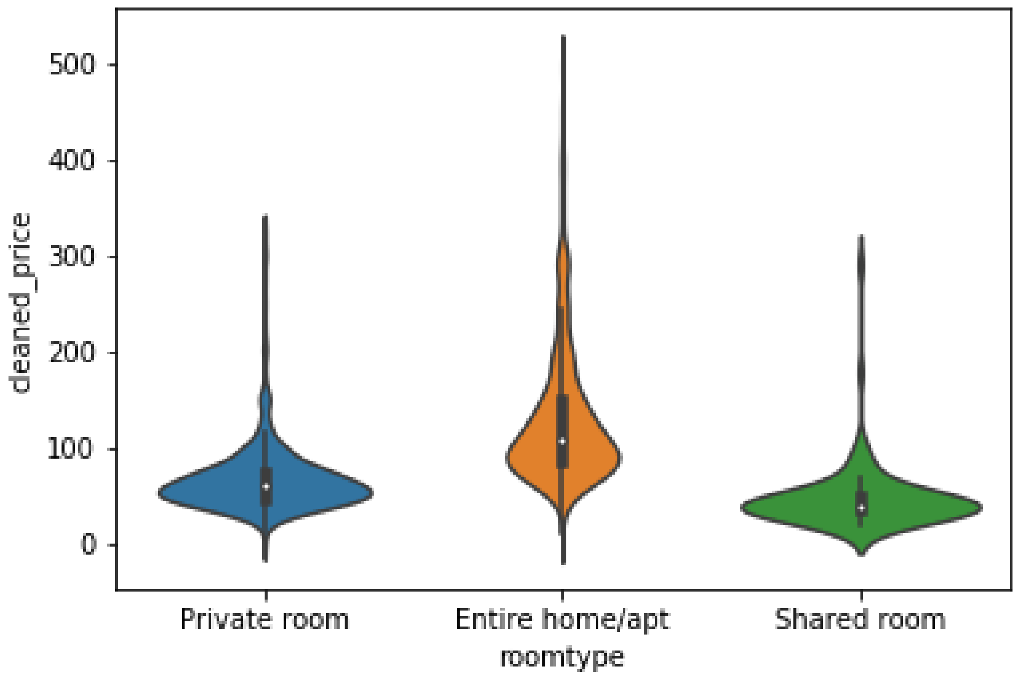
```
In [60]: sns.boxplot(x="roomtype", y="cleaned_price", data=drop_500);
```



0.3.4 Violinpplot of roomtype and price, same area

The first violin uses the default scale parameter of area, i.e., all the plots have the same area. This is nice because you get a feel for how each segment is spread out.

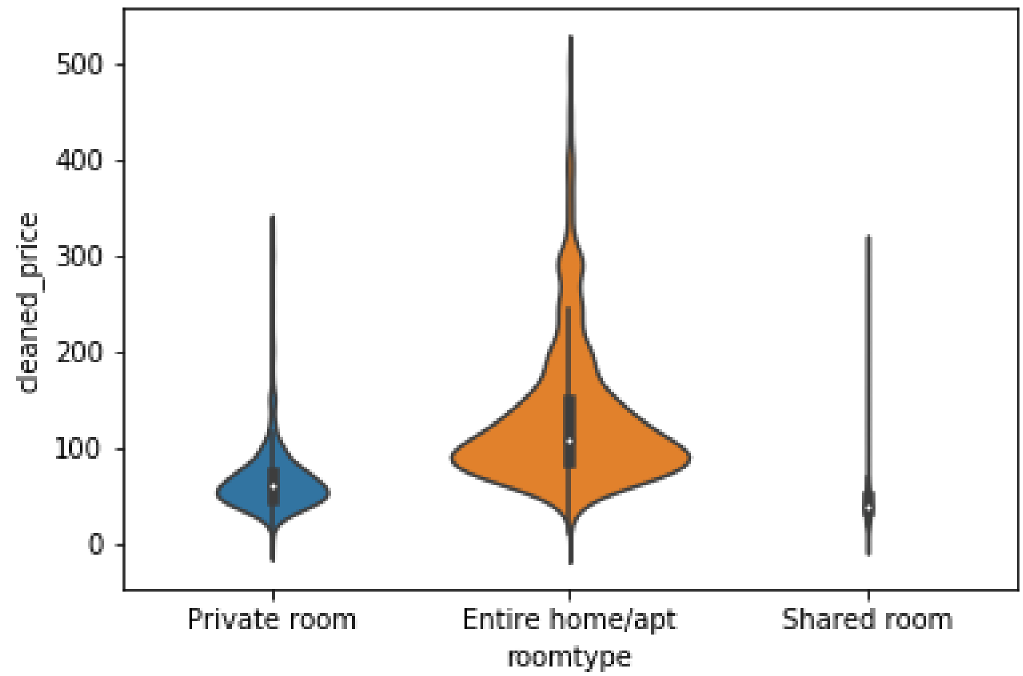
```
In [61]: sns.violinplot(x="roomtype", y="cleaned_price", data=drop_500, split=True);
```



0.3.5 Violinpplot of roomtype and price, using count

This is similar to the last one, but by using the raw count to scale the width, we're able to see how miniscule the "Shared room" category is.

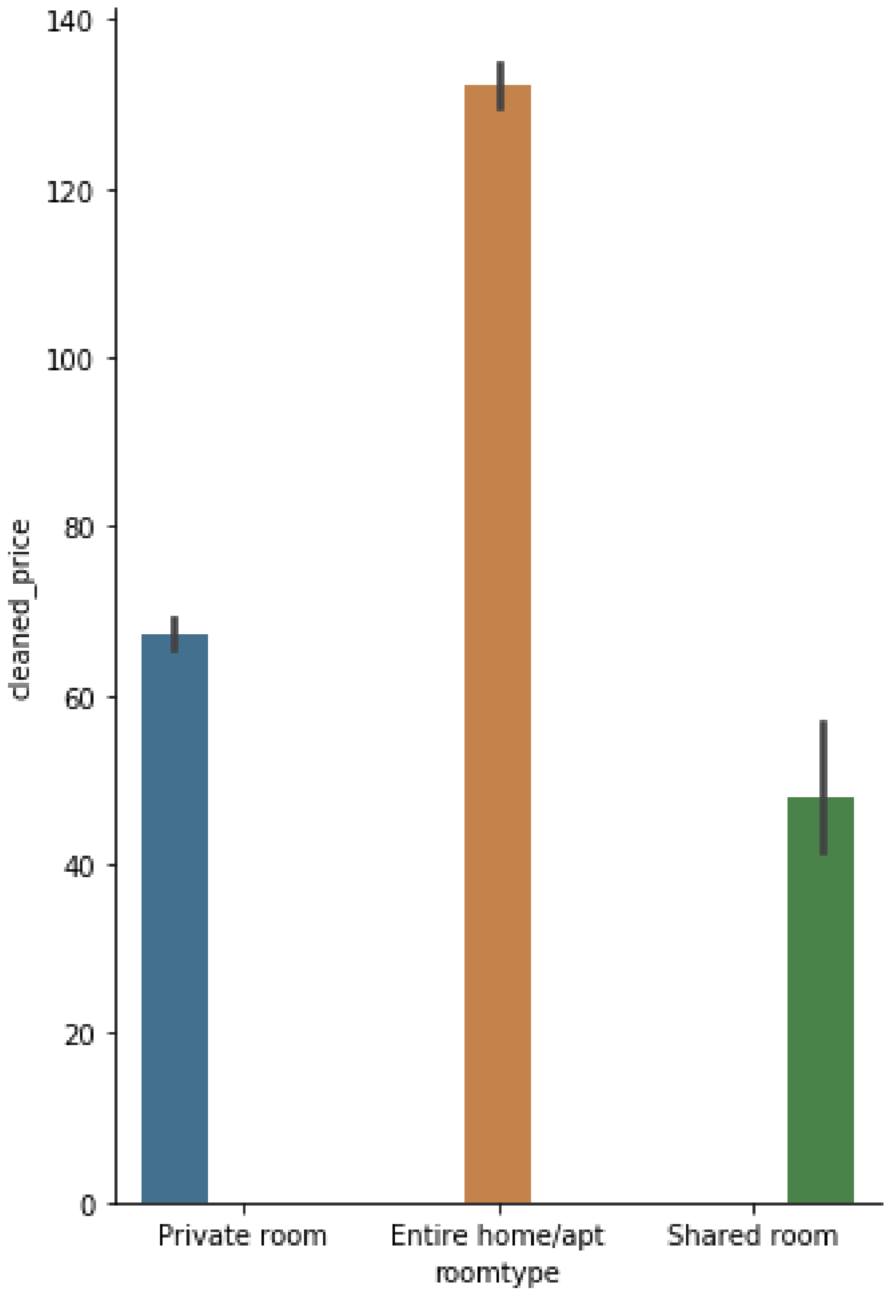
```
In [62]: sns.violinplot(x="roomtype", y="cleaned_price", scale='count', data=drop_500, split=Tru
```



0.3.6 Barplot

This chart allows us to see the mean prices of the different types of rooms (under $500), with 95% confidence interval bars.

```
In [68]: g = sns.factorplot(x="roomtype", y="cleaned_price", hue="roomtype",
                            data=drop_500, saturation=.5, legend=True,
                            kind="bar", size=7, ci=95, aspect=.7)
```



0.4 Challenge