In [ ]:

```
import pandas as pd
import numpy as np
import os
import matplotlib.pyplot as plt
import seaborn as sns
```

In [7]:

```
#Loading Dataset
df_trvl_rvw = pd.read_csv('tripadvisor_review.csv')
df_trvl_rvw = df_trvl_rvw.set_index('User ID')
df_trvl_rvw.head()
```

Out[7]:

	Category 1	Category 2	Category 3	Category 4	Category 5	Category 6	Category 7	Category 8	Category 9
User ID									
1	0.93	1.8	2.29	0.62	0.80	2.42	3.19	2.79	1.82
2	1.02	2.2	2.66	0.64	1.42	3.18	3.21	2.63	1.86
3	1.22	0.8	0.54	0.53	0.24	1.54	3.18	2.80	1.31
4	0.45	1.8	0.29	0.57	0.46	1.52	3.18	2.96	1.57
5	0.51	1.2	1.18	0.57	1.54	2.02	3.18	2.78	1.18

In [9]:

```
#rename the columns for easier understanding
re_name = ['art galleries'
,'dance clubs'
,'juice bars'
,'restaurants'
,'museums'
,'resorts'
,'parks'
,'beaches'
,'theaters'
,'religious']

#rename
df_trvl_rvw.columns = re_name
df_trvl_rvw.head()
```

Out[9]:

	art galleries	dance clubs	juice bars	restaurants	museums	resorts	parks	beaches	theaters	religious
User ID										
1	0.93	1.8	2.29	0.62	0.80	2.42	3.19	2.79	1.82	2.42
2	1.02	2.2	2.66	0.64	1.42	3.18	3.21	2.63	1.86	2.32
3	1.22	0.8	0.54	0.53	0.24	1.54	3.18	2.80	1.31	2.50
4	0.45	1.8	0.29	0.57	0.46	1.52	3.18	2.96	1.57	2.86
5	0.51	1.2	1.18	0.57	1.54	2.02	3.18	2.78	1.18	2.54

In [10]:

```
##Exploratory Data Analysis
display(df_trvl_rvw.info())
display(df_trvl_rvw.describe())
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 980 entries, 1 to 980
Data columns (total 10 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   art galleries  980 non-null    float64
 1   dance clubs    980 non-null    float64
 2   juice bars     980 non-null    float64
 3   restaurants    980 non-null    float64
 4   museums        980 non-null    float64
 5   resorts        980 non-null    float64
 6   parks          980 non-null    float64
 7   beaches        980 non-null    float64
 8   theaters       980 non-null    float64
 9   religious      980 non-null    float64
dtypes: float64(10)
memory usage: 84.2 KB
None
```

	art galleries	dance clubs	juice bars	restaurants	museums	resorts	parks	beac
count	980.000000	980.000000	980.000000	980.000000	980.000000	980.000000	980.000000	980.0000
mean	0.893194	1.352612	1.013306	0.532500	0.939735	1.842898	3.180939	2.835
std	0.326912	0.478280	0.788607	0.279731	0.437430	0.539538	0.007824	0.137
min	0.340000	0.000000	0.130000	0.150000	0.060000	0.140000	3.160000	2.420
25%	0.670000	1.080000	0.270000	0.410000	0.640000	1.460000	3.180000	2.740
50%	0.830000	1.280000	0.820000	0.500000	0.900000	1.800000	3.180000	2.820
75%	1.020000	1.560000	1.572500	0.580000	1.200000	2.200000	3.180000	2.910
max	3.220000	3.640000	3.620000	3.440000	3.300000	3.760000	3.210000	3.390

In [11]:

```
#All attribute should be float64 datatype and we have the same here so therfore the
print('Total missing values in dataset')
display(df_trvl_rvw.isnull().sum())
df_trvl_rvw = df_trvl_rvw.dropna()
```

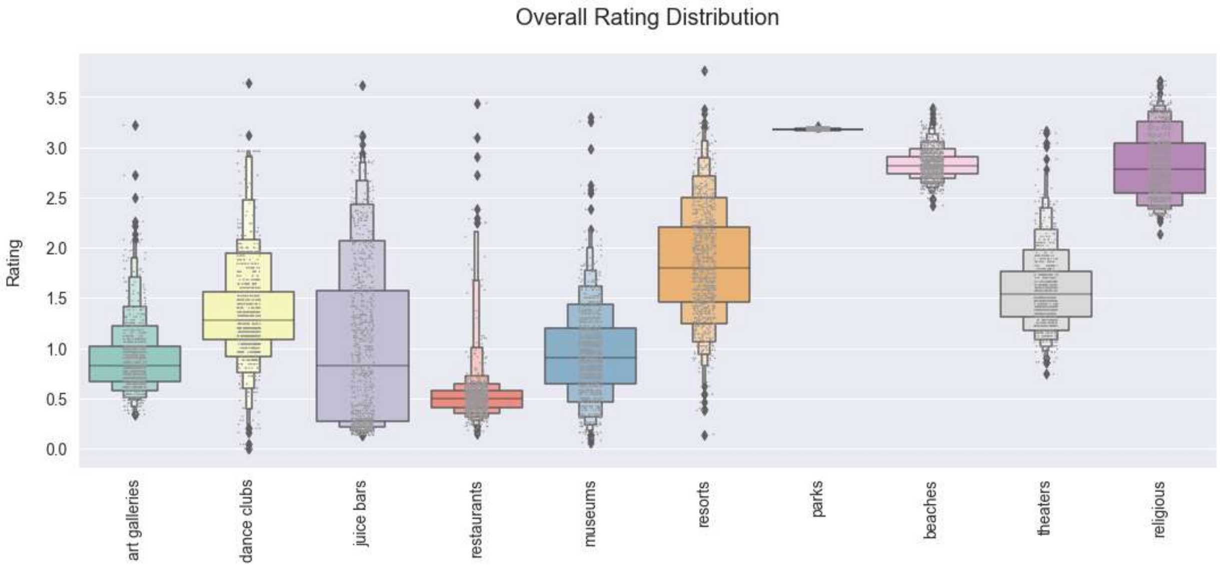
```
Total missing values in dataset
art galleries    0
dance clubs      0
juice bars       0
restaurants      0
museums          0
resorts          0
parks            0
beaches          0
theaters         0
religious        0
dtype: int64
```

In [12]:

```
##There are no missing values in our dataset
##Now let's look how our data is distributed
sns.set(style='darkgrid',palette = 'Set3',font_scale=1.25)
```

In [15]:

```
df_transform = pd.melt(df_trvl_rvw,value_vars=['art galleries'
,'dance clubs'
,'juice bars'
,'restaurants'
,'museums'
,'resorts'
,'parks'
,'beaches'
,'theaters'
,'religious'])
fig = plt.figure(figsize = (15,7))
garph = sns.boxenplot(x='variable',y='value',data=df_transform,palette = 'Set3')
garph = sns.stripplot(x='variable',y='value',data=df_transform,size=1.5, color=".6")
garph.set_xticklabels(garph.get_xticklabels(),rotation=90);
garph.set_title(f'Overall Rating Distribution',y=1.05,fontsize=20)
garph.set_xlabel("")
garph.set_ylabel("Rating",labelpad = 20)
fig.tight_layout(pad = 0.5)
plt.savefig('Overall Rating Distribution.png')
```



In [ ]:

```
#Since there is no much information regarding the attractions or the descriptive use
# reviews and then cluster them into different preferences
##K-Means Clustering : Find right number of cluster
#  we  will try K-Means clustering algorithm on 4 Scenarios and compared it result

# -> K-Means Clustering on original data (10 features)
# -> K-Means Clustering on scaled orignial data (10 features with scaled) by using
# -> K-Means Clustering on PCA component
# -> K-Means Clustering on PCA component with scaled data
```

In [17]:

```
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_samples, silhouette_score
from sklearn.decomposition import PCA
import matplotlib.ticker as ticker

#Original data
df_orig = df_trvl_rvw

#Scaled Data
df_scaled = StandardScaler().fit_transform(df_trvl_rvw)
```

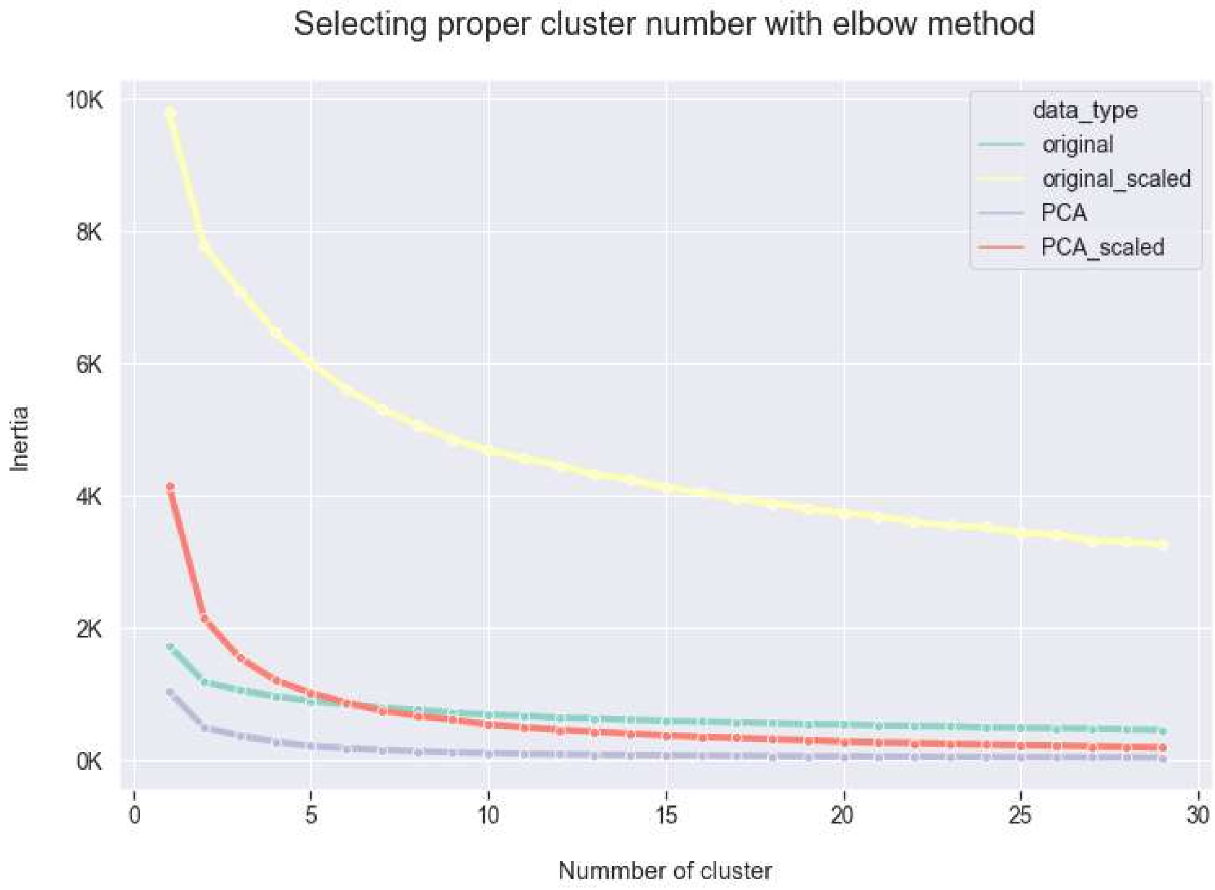
```
#PCA without scaling
pca = PCA(n_components = 2,random_state=42)
df_PCA = pca.fit_transform(df_orig)

#PCA with scaling
df_PCA_scaled = pca.fit_transform(df_scaled)

data_list = [df_orig,df_scaled,df_PCA,df_PCA_scaled]
inertia_list = []
list_k = list(range(1, 30))
```

In [18]:

```
#Run elbow to evaluate number of clusters
for i in range(len(data_list)):
    sse = []
    data = data_list[i]
    for k in list_k:
        km = KMeans(n_clusters=k,random_state=42)
        km.fit(data)
        sse.append(km.inertia_)
    inertia_list.append(sse)

result_ori = pd.DataFrame({'K':list_k,'Inertia':inertia_list[0],'data_type':'origina
result_ori_scaled = pd.DataFrame({'K':list_k,'Inertia':inertia_list[1],'data_type':'
result_PCA = pd.DataFrame({'K':list_k,'Inertia':inertia_list[2],'data_type':'PCA'})
result_PCA_scaled = pd.DataFrame({'K':list_k,'Inertia':inertia_list[3],'data_type':'
result = result_ori.append(result_ori_scaled).append(result_PCA).append(result_PCA_s
```

In [19]:

```
fig = plt.figure(figsize=(12,8))
graph = sns.lineplot(data=result,x='K',y='Inertia',hue='data_type',linewidth=4,marke
graph.set_xlabel('Nummber of cluster', labelpad = 20)
graph.set_ylabel('Inertia', labelpad = 20)
graph.yaxis.set_major_formatter(ticker.FuncFormatter(lambda x, pos: '{:,.0f}'.format
graph.tick_params(which="both", bottom=True)
graph.set_title('Selecting proper cluster number with elbow method',y=1.05, fontsize
```



In [20]:
```
pca.explained_variance_ratio_.sum()
```

Out[20]:
```
0.42402983746016876
```

In [21]:
```
#K-Mean clustering algorithm on only 2 principle components which represent only 42%
#(cummilative explianed variance ratio ~ 42%) may result in not so good clustering p
#In conclusion, I'll continue on K-Mean clustering analysis on PCA_scaled data with
```

In [22]:
```
pca = PCA(n_components = 2,random_state=42)
df_PCA_scaled = pca.fit_transform(df_scaled)

model = KMeans(n_clusters=4,random_state=42)
model.fit(df_PCA_scaled)
cluster = model.labels_
```

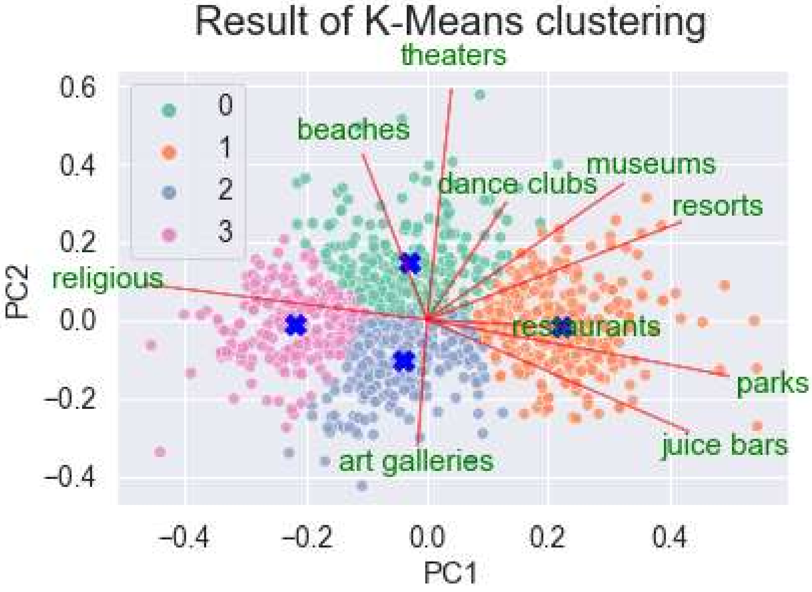
In [32]:
```
coeff = np.transpose(pca.components_[0:2, :])
n = coeff.shape[0]
labels = list(df_trvl_rvw.columns)
xs = df_PCA_scaled[:,0]
ys = df_PCA_scaled[:,1]
scalex = 1.0/(xs.max() - xs.min())
scaley = 1.0/(ys.max() - ys.min())
```

In [33]:
```
fig = plt.figure(figsize=(15,10))
```

```
<Figure size 1080x720 with 0 Axes>
```

In [37]:
```
#scatter plot of each data point
graph = sns.scatterplot(xs * scalex,ys * scaley,hue=cluster,palette='Set2',alpha=0.8
#add cluster centroid
plt.scatter(x=model.cluster_centers_[:,0]*scalex, y=model.cluster_centers_[:,1]*scal

#add EigenVector representing how each attraction categories related to PC1 and PC2
for i in range(n):
        plt.arrow(0, 0, coeff[i,0], coeff[i,1],color = 'r',alpha = 0.5)
        plt.text(coeff[i,0]* 1.15, coeff[i,1] * 1.15, labels[i], color = 'g', ha = '


graph.set_xlabel("PC{}".format(1))
graph.set_ylabel("PC{}".format(2))
graph.set_title('Result of K-Means clustering',y=1.05, fontsize=20);
plt.savefig('Clustering Result.png')
```

```
c:\users\shanty\pycharmprojects\sh_dataset\lib\site-packages\seaborn\_decorators.py:
36: FutureWarning: Pass the following variables as keyword args: x, y. From version
0.12, the only valid positional argument will be `data`, and passing other arguments
without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```



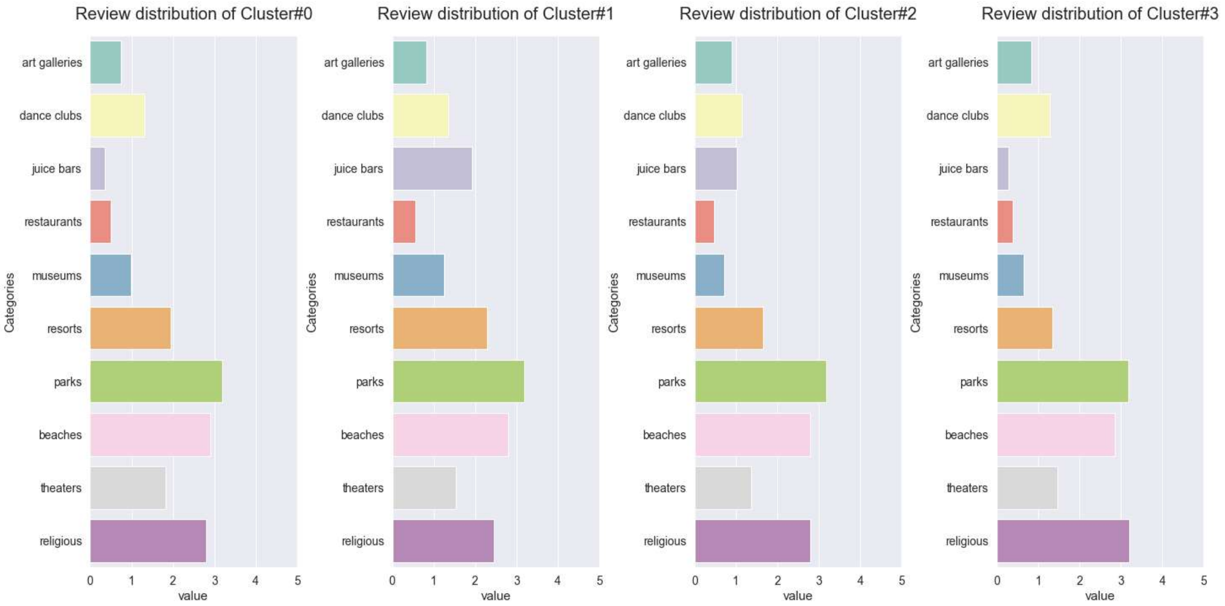
In [38]:

```
#K-Means clustering result in 4 clusters(segments) of user as follow,
#
#Cluster#0 (Green) : User who prefer nature like beaches , theatres and dance clubs
#Cluster#1 (Orange) : this users likes resorts, museums, parks, juice bars and resto
#Cluster#2 (Light Blue) : Art Lovers - dedicated to Art gallerries
#Cluster#3 (Pink) : From scatter plot, member of this cluster are not loosely spread
```

In [39]:

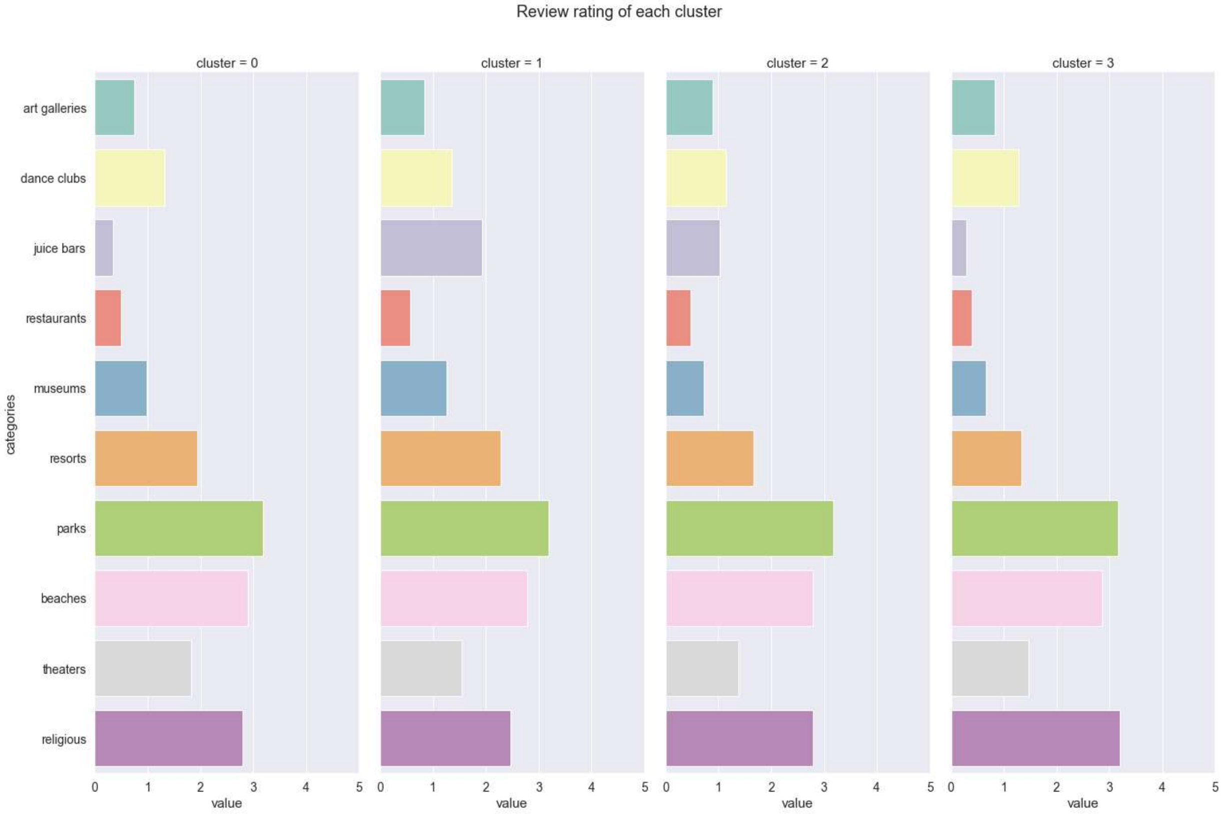
```
df_trvl_rvw['cluster'] = model.labels_
df_long = pd.melt(df_trvl_rvw, "cluster", var_name="categories")

fig = plt.figure(figsize = (20,10))
for i in range(len(df_long.cluster.unique())):
    plt.subplot(1,len(df_long.cluster.unique()),i+1)
    graph = sns.barplot(y='categories',x='value',data=df_long[df_long['cluster']==i]
#     g.set_xticklabels(g.get_xticklabels(),rotation=90);
    graph.set_title(f'Review distribution of Cluster#{i}',y=1.02,fontsize=20)
#     g.set_xlabel("")
    graph.set_ylabel('Categories')
    graph.set_xlim(0,5)
fig.tight_layout(pad=0.5)
plt.savefig
```

```
Out[39]: <function matplotlib.pyplot.savefig(*args, **kwargs)>
```



```
In [40]:
fig = plt.figure(figsize = (20,10))
graph = sns.catplot(y='categories',x='value',data=df_long,kind='bar',
                palette='Set3',ci=None,estimator=np.median,col = 'cluster',
               height = 12,aspect=0.4,
               facet_kws={'xlim':(0,5)})
fig.tight_layout(pad=0.5)
graph.fig.suptitle('Review rating of each cluster',y=1.05,ha='center')
plt.savefig('Review rating of each cluster')
```

```
<Figure size 1440x720 with 0 Axes>
```


Review rating of each cluster
cluster = 0
cluster = 1
cluster = 2
cluster = 3
categories
art galleries
dance clubs
juice bars
restaurants
museums
resorts
parks
beaches
theaters
religious
value