Dimensionality Reduction with PCA

In [1]:
```python
#import libraries necessary to conduct primary componant analysis in python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
```

In [2]:
```python
#load the telecommunications data set into a pandas dataframe and inspect the first few lines
data = pd.read_csv('/Users/benjaminmcdaniel/Desktop/MSDA/D212/teleco_data/churn_clean.csv')

#set pandas options to read full print and display output
pd.set_option('display.max_rows', None)
pd.set_option('display.max_columns', None)
pd.set_option('display.width', None)
pd.set_option('display.max_colwidth', None)

#inspect the first few lines of the data set as a pandas dataframe
data.head()
```

Out[2]:

	CaseOrder	Customer_id	Interaction	UID	City	State	County	Zip	Lat	Lng	Population	Area	TimeZone	Job	Children
0	1	K409198	aa90260b-4141-4a24-8e36-b04ce1f4f77b	e885b299883d4f9fb18e39c75155d990	Point Baker	AK	Prince of Wales-Hyder	99927	56.25100	-133.37571	38	Urban	America/Sitka	Environmental health practitioner	0
1	2	S120509	fb76459f-c047-4a9d-8af9-e0f7d4ac2524	f2de8bef964785f41a2959829830fb8a	West Branch	MI	Ogemaw	48661	44.32893	-84.24080	10446	Urban	America/Detroit	Programmer, multimedia	1
2	3	K191035	344d114c-3736-4be5-98f7-c72c281e2d35	f1784cfa9f6d92ae816197eb175d3c71	Yamhill	OR	Yamhill	97148	45.35589	-123.24657	3735	Urban	America/Los_Angeles	Chief Financial Officer	4
3	4	D90850	abfa2b40-2d43-4994-b15a-989b8c79e311	dc8a365077241bb5cd5ccd305136b05e	Del Mar	CA	San Diego	92014	32.96687	-117.24798	13863	Suburban	America/Los_Angeles	Solicitor	1
4	5	K662701	68a861fd-0d20-4e51-a587-8a90407ee574	aabb64a116e83fdc4befc1fbab1663f9	Needville	TX	Fort Bend	77461	29.38012	-95.80673	11352	Suburban	America/Chicago	Medical illustrator	0

Reduce the Data Set

In [3]:
```python
#select the continuous numeric features as a new dataframe for primary componant analysis
df_continuous = data[['Lat', 'Lng', 'Population', 'Children', 'Age', 'Income', 'Outage_sec_perweek', 'Email',
               'Contacts', 'Yearly_equip_failure', 'Tenure', 'MonthlyCharge', 'Bandwidth_GB_Year']]
```

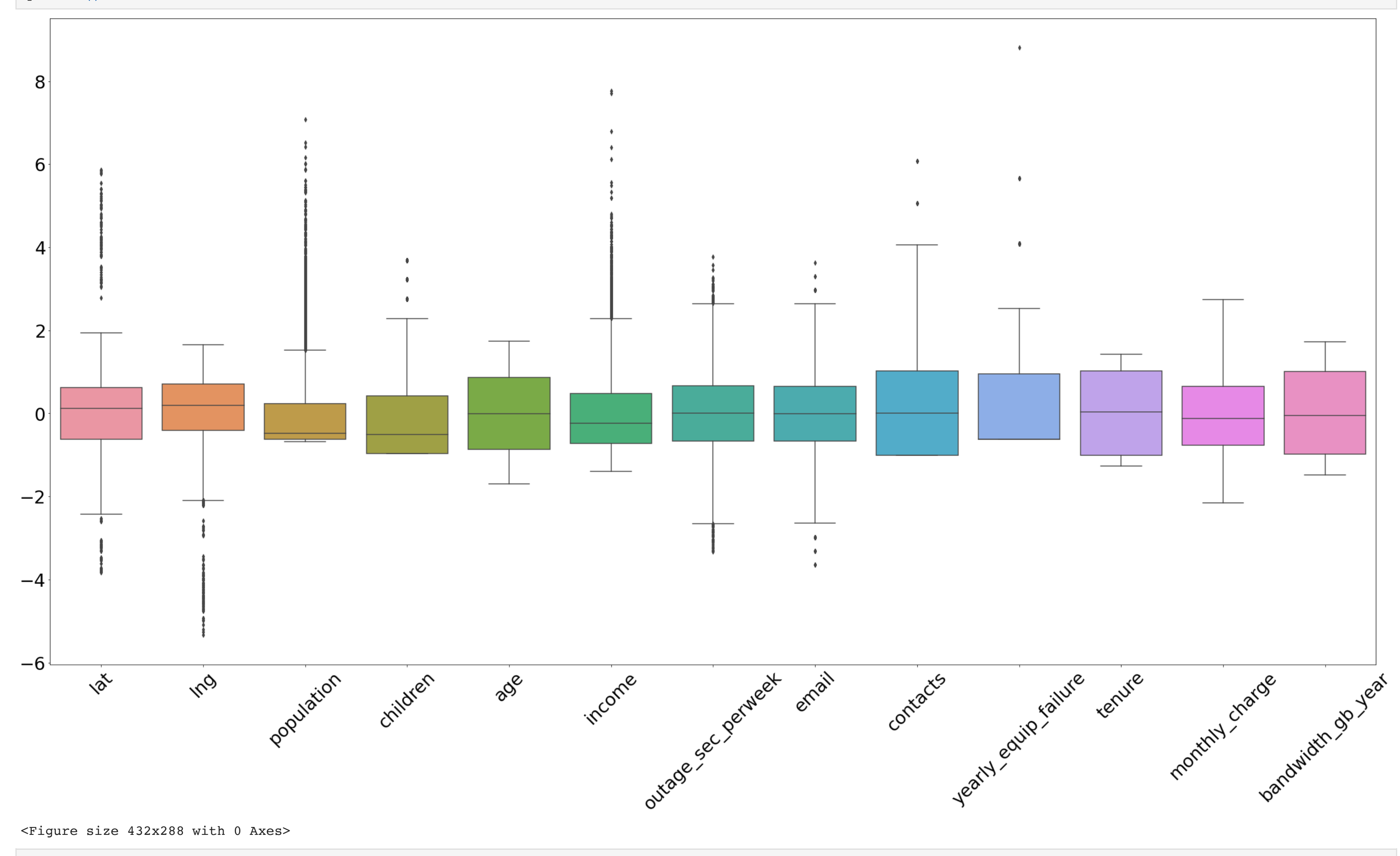
In [4]:
```python
#rename features to conform with pythonic variable conventions
df_continuous = df_continuous.rename(columns = {'Lat':'lat','Lng':'lng', 'Population':'population', 'Children':'children', 'Age':'age',
                        'Income':'income', 'Outage_sec_perweek':'outage_sec_perweek', 'Email':'email',
                        'Contacts':'contacts','Yearly_equip_failure':'yearly_equip_failure', 'Tenure':'tenure',
                        'MonthlyCharge':'monthly_charge', 'Bandwidth_GB_Year':'bandwidth_gb_year'})
```

In [5]:
```python
#inspect the renamed columns of the dataframe
print(df_continuous.columns)
```

```
Index(['lat', 'lng', 'population', 'children', 'age', 'income',
       'outage_sec_perweek', 'email', 'contacts', 'yearly_equip_failure',
       'tenure', 'monthly_charge', 'bandwidth_gb_year'],
      dtype='object')
```

Transform the Data Set

In [6]:
```python
#instatiate a standard scaler object adn fit/transform the dataframe
scaler = StandardScaler()
scaled_df = scaler.fit_transform(df_continuous)
cleaned_data_set = pd.DataFrame(scaled_df, columns=df_continuous.columns)
cleaned_data_set.to_csv('/Users/benjaminmcdaniel/Desktop/MSDA/D212/task_2/cleaned_data_set.csv')
plt.figure(figsize=(40,20))
plt.rcParams.update({'font.size': 30})

ax = sns.boxplot(data = cleaned_data_set)
ax.set_xticklabels(ax.get_xticklabels(),rotation = 45)
plt.show()
plt.clf()
```




```
<Figure size 432x288 with 0 Axes>
```

In [7]:
```python
#prepare to collect pca data in a pandas dataframe
pc_columns = []
#loop for item PC names here
for i, col in enumerate(df_continuous.columns):
    pc_columns.append('PC'+str(i+1))
print(pc_columns)
```

```
['PC1', 'PC2', 'PC3', 'PC4', 'PC5', 'PC6', 'PC7', 'PC8', 'PC9', 'PC10', 'PC11', 'PC12', 'PC13']
```

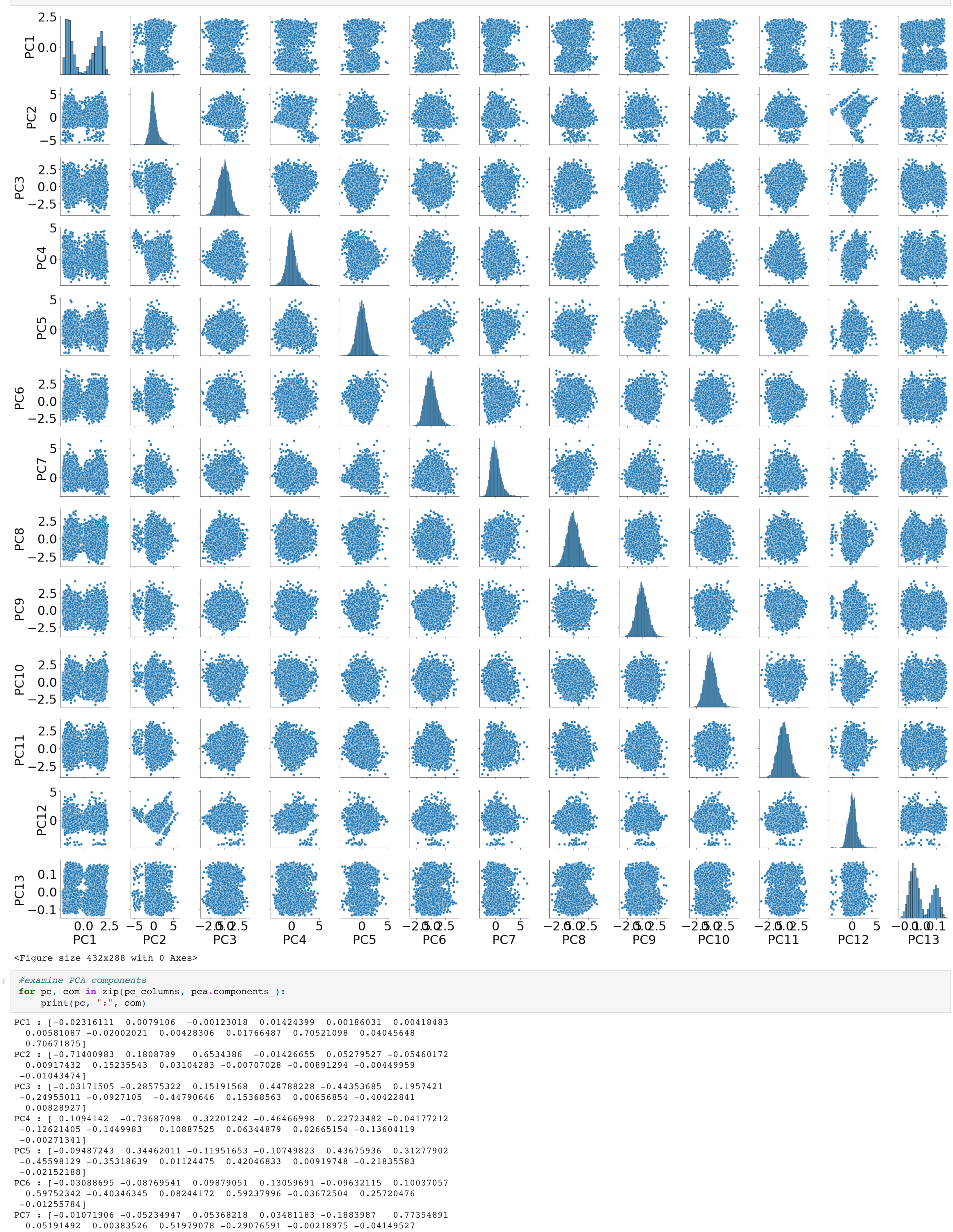
Primary Componant Analysis

In [8]:
```python
#instantiate a PCA object fit to scaled data create empty lists to hold explained variance and cumsum variance
pca = PCA()
pca.fit(scaled_df)
ex_var_ratio = []
cumulative_var_ratio = []
for pc, evr in zip(pc_columns, pca.explained_variance_ratio_):
    ex_var_ratio.append([pc,evr])
print('Explained Variance Ratios:')
display(ex_var_ratio)


pc1=pca.transform(scaled_df)


for pc, cum in zip(pc_columns, pca.explained_variance_ratio_.cumsum()):
    cumulative_var_ratio.append([pc, cum])

print('Cumulative Variance Ratio: ')
display(cumulative_var_ratio)
```

```
Explained Variance Ratios:
[['PC1', 0.15343888168137707],
 ['PC2', 0.09492523456536195],
 ['PC3', 0.08105110761065369],
 ['PC4', 0.08035307994903408],
 ['PC5', 0.07870659594194752],
 ['PC6', 0.07774467247017867],
 ['PC7', 0.07686174226454635],
 ['PC8', 0.07635199341481065],
 ['PC9', 0.07572558479879354],
 ['PC10', 0.0739838627914553],
 ['PC11', 0.07387686615210366],
 ['PC12', 0.05655995468957029],
 ['PC13', 0.0004204236701672844]]
Cumulative Variance Ratio:
[['PC1', 0.15343888168137707],
 ['PC2', 0.24836411624673901],
 ['PC3', 0.3294152238573927],
 ['PC4', 0.40976830380642676],
 ['PC5', 0.48847489974837427],
 ['PC6', 0.5662195722185529],
 ['PC7', 0.6430813144830992],
 ['PC8', 0.7194333078979099],
 ['PC9', 0.7951588926967035],
 ['PC10', 0.8691427554881588],
 ['PC11', 0.9430196216402624],
 ['PC12', 0.9995795763298327],
 ['PC13', 1.0]]
```

In [9]:
```python
# This changes the numpy array output back to a dataframe
pc_df = pd.DataFrame(pc1, columns=pc_columns)
```

In [10]:
```python
# Create a pairplot of the principal component dataframe
sns.pairplot(pc_df)
plt.show()
plt.clf()
```



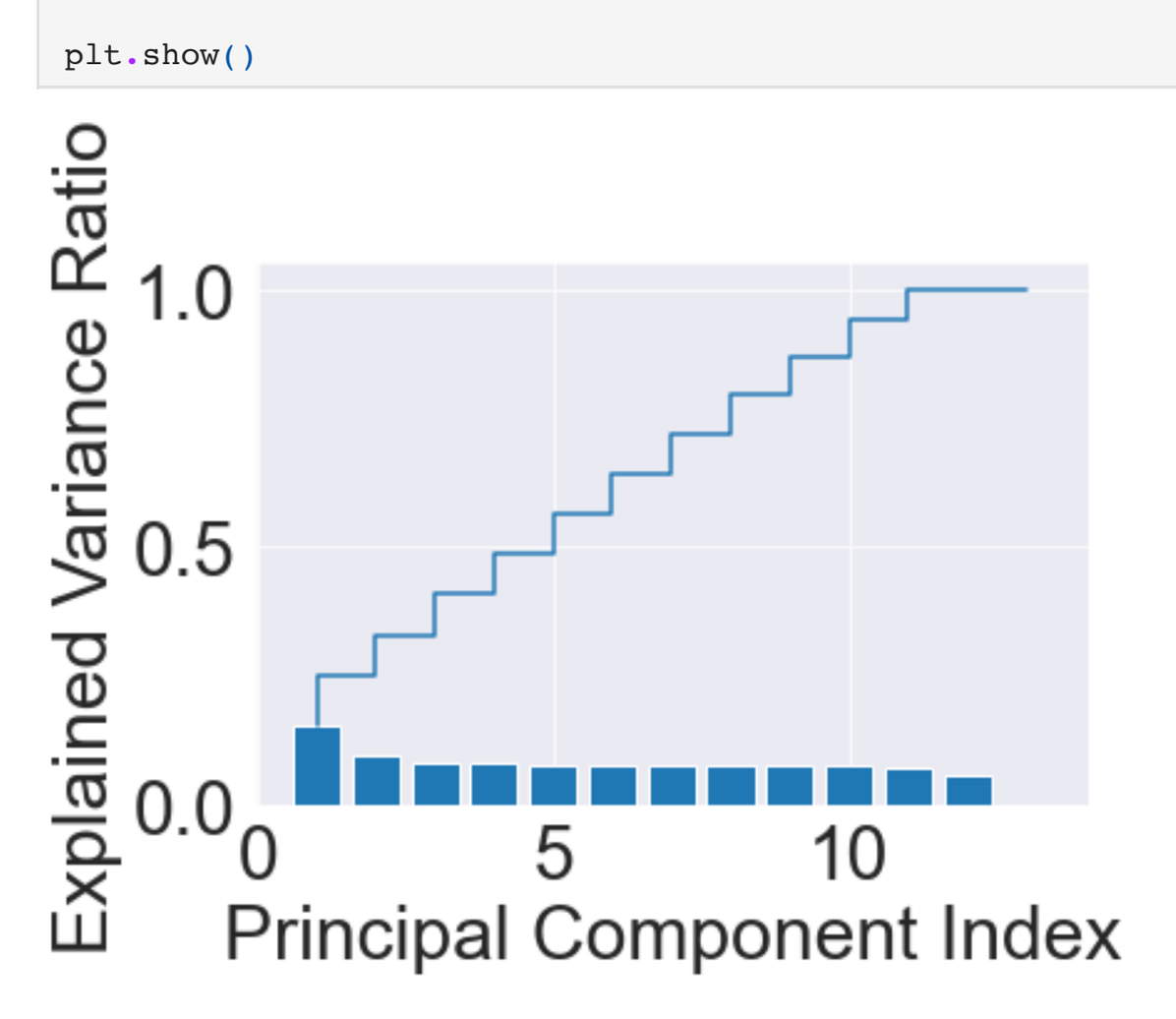

```
<Figure size 432x288 with 0 Axes>
```

In [11]:
```python
#examine PCA components
for pc, com in zip(pc_columns, pca.components_):
    print(pc, ":", com)
```

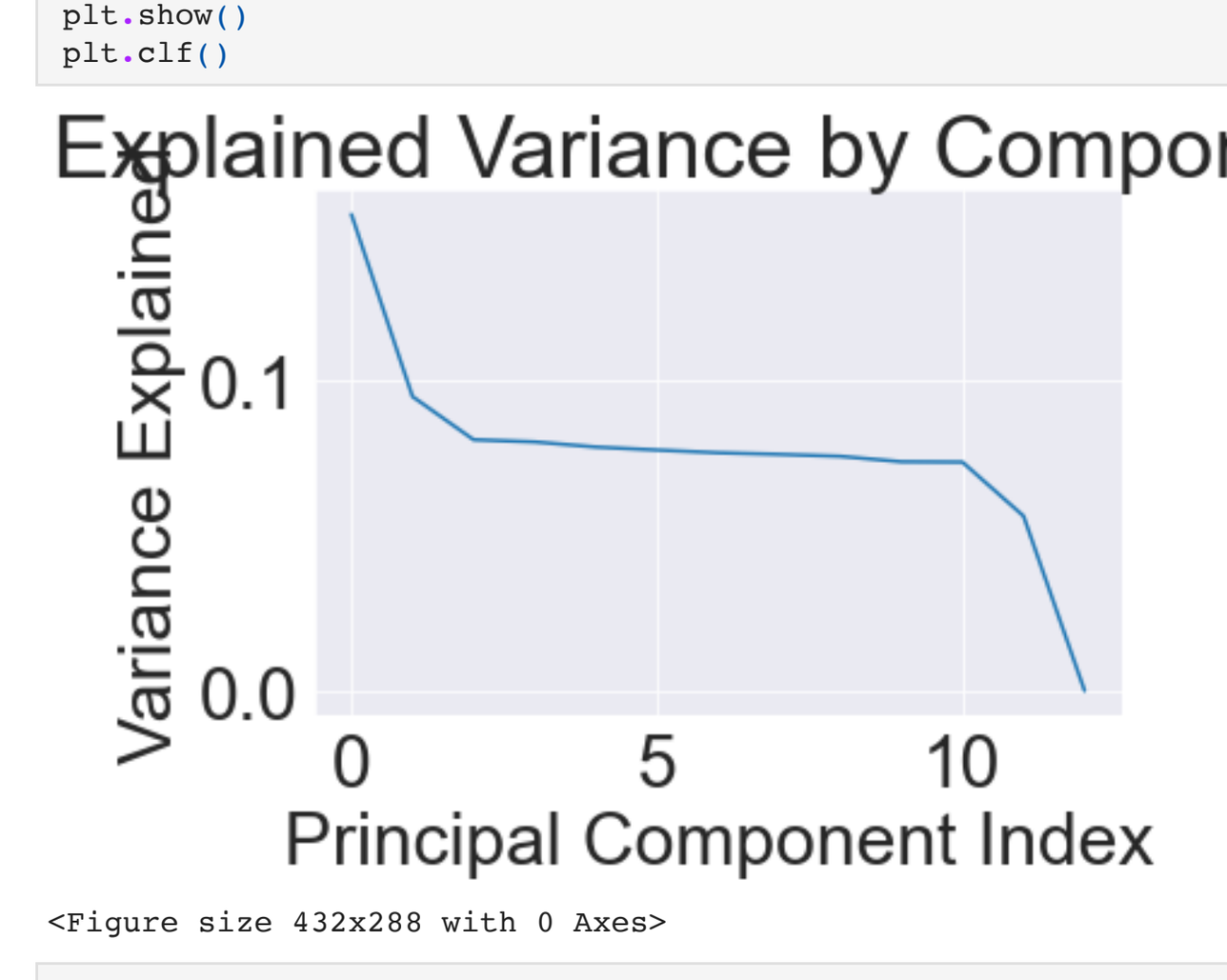
```
PC1 : [-0.02316111  0.0079106  -0.00123018  0.01424399  0.00186031  0.00418483
  0.00581087 -0.02002021  0.00428306  0.01766487  0.70521098  0.04045648
  0.70671875]
PC2 : [-0.71400983  0.1808789   0.6534386  -0.01426655  0.05279527 -0.05460172
  0.00917432  0.15235543  0.03104283 -0.00707028 -0.00891294 -0.00449959
 -0.01043474]
PC3 : [-0.03171505 -0.28575322  0.15191568  0.44788228 -0.44353685  0.1957421
 -0.24955011 -0.0927105  -0.44790646  0.15368563  0.00656854 -0.40422841
  0.00828927]
PC4 : [ 0.1094142  -0.73687098  0.32201242 -0.46466998  0.22723482 -0.04177212
 -0.12621405 -0.1449983   0.10887525  0.06344879  0.02665154 -0.13604119
 -0.00271341]
PC5 : [-0.09487243  0.34462011 -0.11951653 -0.10749823  0.43675936  0.31277902
 -0.45598129 -0.35318639  0.01124475  0.42046833  0.00919748 -0.21835583
 -0.02152188]
PC6 : [-0.03088695 -0.08769541  0.09879051  0.13059691 -0.09632115  0.10037057
  0.59752342 -0.40346345  0.08244172  0.59237996 -0.03672504  0.25720476
 -0.01255784]
PC7 : [-0.01071906 -0.05234947  0.05368218  0.03481183 -0.1883987   0.77354891
  0.05191492  0.00383526  0.51979078 -0.29076591 -0.00218975 -0.04149527
  0.00390181]
PC8 : [-0.02037542 -0.08649929  0.07916123 -0.0655314   0.09348401  0.33546672
 -0.18465811 -0.12537522 -0.51097448 -0.19466458 -0.03843266  0.71412342
  0.00292596]
PC9 : [ 0.09027321 -0.17228541 -0.02739246  0.19245871  0.34289226  0.24666275
  0.05705562  0.76062196 -0.05269535  0.3970884   0.00380566  0.06066918
  0.00279843]
PC10 : [ 0.01861947 -0.1513015   0.05530402  0.43747056 -0.08359556 -0.2758521
 -0.51540599 -0.05214585  0.49460068  0.14341945 -0.03733868  0.40527977
  0.00567312]
PC11 : [ 0.05395798 -0.11227961  0.100818    0.56562605  0.61489189 -0.03374202
  0.22330369 -0.24798548 -0.02819421 -0.37694323  0.00549149 -0.14437566
 -0.00610163]
PC12 : [ 0.67437588  0.3751382   0.63172851 -0.01179389 -0.03772866  0.00664508
 -0.03415521  0.02735702 -0.01187787  0.03888001  0.01039276 -0.00602083
  0.00942943]
PC13 : [ 1.07726186e-03  7.88245116e-04 -6.95438537e-05 -2.15967978e-02
  2.23595547e-02 -9.40589140e-04  2.71429940e-04  2.74100814e-04
 -9.46507467e-04 -8.30740386e-05 -7.05253552e-01 -4.57658858e-02
  7.06790504e-01]
```

Visualize Feature Importance

In [12]:
```python
#plot pca for visualization of feature importance
var = pca.explained_variance_ratio_
var_sum = pca.explained_variance_ratio_.cumsum()
sns.set_style('darkgrid')
plt.bar(range(1,14), var, label = 'Componant Explained Variance')
plt.step(range(1,14),var_sum, label='Cumulative Explained Variance')
plt.ylabel('Explained Variance Ratio')
plt.xlabel('Principal Component Index')

plt.show()
```



In [13]:
```python
#elbow plot for determination of principal componants
plt.plot(var)
plt.xlabel('Principal Component Index')
plt.ylabel('Variance Explained')
plt.title('Explained Variance by Component')
plt.show()
plt.clf()
```




```
<Figure size 432x288 with 0 Axes>
```

In [14]:
```python
#create a dataframe with the transformed scaled data assign column names to reflect pc's aka rotation plot
data_pca = pd.DataFrame(pca.components_.T, columns=[pc_columns], index = df_continuous.columns)
display(data_pca)
```

	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8	PC9	PC10	PC11	PC12	PC13
lat	-0.023161	-0.714010	-0.031715	0.109414	-0.094872	-0.030887	-0.010719	-0.020375	0.090273	0.018619	0.053958	0.674376	0.001077
lng	0.007911	0.180879	-0.285753	-0.736871	0.344620	-0.087695	-0.052349	-0.086499	-0.172285	-0.151301	-0.112280	0.375138	0.000788
population	-0.001230	0.653439	0.151916	0.322012	-0.119517	0.098791	0.053682	0.079161	-0.027392	0.055304	0.100818	0.631729	-0.000070
children	0.014244	-0.014267	0.447882	-0.464670	-0.107498	0.130597	0.034812	-0.065531	0.192459	0.437471	0.565626	-0.011794	-0.021597
age	0.001860	0.052795	-0.443537	0.227235	0.436759	-0.096321	-0.188399	0.093484	0.342892	-0.083596	0.614892	-0.037729	0.022360
income	0.004185	-0.054602	0.195742	-0.041772	0.312779	0.100371	0.773549	0.335467	0.246663	-0.275852	-0.033742	0.006645	-0.000941
outage_sec_perweek	0.005811	0.009174	-0.249550	-0.126214	-0.455981	0.597523	0.051915	-0.184658	0.057056	-0.515406	0.223304	-0.034155	0.000271
email	-0.020020	0.152355	-0.092711	-0.144998	-0.353186	-0.403463	0.003835	-0.125375	0.760622	-0.052146	-0.247985	0.027357	0.000274
contacts	0.004283	0.031043	-0.447906	0.108875	0.011245	0.082442	0.519791	-0.510974	-0.052695	0.494601	-0.028194	-0.011878	-0.000947
yearly_equip_failure	0.017665	-0.007070	0.153686	0.063449	0.420468	0.592380	-0.290766	-0.194665	0.397088	0.143419	-0.376943	0.038880	-0.000083
tenure	0.705211	-0.008913	0.006569	0.026652	0.009197	-0.036725	-0.002190	-0.038433	0.003806	-0.037339	0.005491	0.010393	-0.705254
monthly_charge	0.040456	-0.004500	-0.404228	-0.136041	-0.218356	0.257205	-0.041495	0.714123	0.060669	0.405280	-0.144376	-0.006021	-0.045766
bandwidth_gb_year	0.706719	-0.010435	0.008289	-0.002713	-0.021522	-0.012558	0.003902	0.002926	0.002798	0.005673	-0.006102	0.009429	0.706791

In [15]:
```python
#generate covariance matrix of standardized data set for exploration
cov_matrix = np.dot(scaled_df.T, scaled_df) / data.shape[0]
print(cov_matrix)
```

```
[[ 1.00000000e+00 -1.00639442e-01 -2.20598060e-01 -4.51506697e-04
  -2.94264831e-03  6.56316072e-03  4.82696580e-03 -3.00416885e-02
  -2.21328053e-03 -7.12525316e-03 -1.57430084e-02  1.40600319e-03
  -1.69490060e-02]
 [-1.00639442e-01  1.00000000e+00 -4.82939143e-02  1.36342145e-02
   1.21165551e-02 -2.99704648e-03  5.64627984e-03  7.87216322e-03
   8.11816381e-03 -5.57697742e-03  3.61807449e-03  1.17653294e-02
   3.44582446e-03]
 [-2.20598060e-01 -4.82939143e-02  1.00000000e+00 -5.87684543e-03
   1.05382732e-02 -8.63859322e-03  5.48327402e-03  1.79615467e-02
   4.01876430e-03 -4.48289564e-03 -3.55943835e-03 -4.77827357e-03
  -3.90182535e-03]
 [-4.51506697e-04  1.36342145e-02 -5.87684543e-03  1.00000000e+00
  -2.97315400e-02  9.94235386e-03  1.88925544e-03  4.47880100e-03
  -2.07760327e-02  7.32058656e-03 -5.09131776e-03 -9.78139863e-03
   2.55848165e-02]
 [-2.94264831e-03  1.21165551e-02  1.05382732e-02 -2.97315400e-02
   1.00000000e+00 -4.09060180e-03 -8.04671909e-03  1.58791708e-03
   1.50676237e-02  8.57734824e-03  1.69792731e-02  1.07285115e-02
  -1.47236478e-02]
 [ 6.56316072e-03 -2.99704648e-03 -8.63859322e-03  9.94235386e-03
  -4.09060180e-03  1.00000000e+00 -1.00105457e-02 -9.26749692e-03
   1.23319883e-03  5.42327631e-03  2.11436699e-03 -3.01396498e-03
   3.67354963e-03]
 [ 4.82696580e-03  5.64627984e-03  5.48327402e-03  1.88925544e-03
  -8.04671909e-03 -1.00105457e-02  1.00000000e+00  3.99372981e-03
   1.50916827e-02  2.90872554e-03  2.93195838e-03  2.04960735e-02
   4.17566138e-03]
 [-3.00416885e-02  7.87216322e-03  1.79615467e-02  4.47880100e-03
   1.58791708e-03 -9.26749692e-03  3.99372981e-03  1.00000000e+00
   3.04036164e-03 -1.63543439e-02 -1.44678779e-02  1.99655434e-03
  -1.45791480e-02]
 [-2.21328053e-03  8.11816381e-03  4.01876430e-03 -2.07760327e-02
   1.50676237e-02  1.23319883e-03  1.50916827e-02  3.04036164e-03
   1.00000000e+00 -6.03224933e-03  2.82009067e-03  4.25864680e-03
   3.29872381e-03]
 [-7.12525316e-03 -5.57697742e-03 -4.48289564e-03  7.32058656e-03
   8.57734824e-03  5.42327631e-03  2.90872554e-03 -1.63543439e-02
  -6.03224933e-03  1.00000000e+00  1.24349112e-02 -7.17227639e-03
   1.20336931e-02]
 [-1.57430084e-02  3.61807449e-03 -3.55943835e-03 -5.09131776e-03
   1.69792731e-02  2.11436699e-03  2.93195838e-03 -1.44678779e-02
   2.82009067e-03  1.24349112e-02  1.00000000e+00 -3.33681041e-03
   9.91495192e-01]
 [ 1.40600319e-03  1.17653294e-02 -4.77827357e-03 -9.78139863e-03
   1.07285115e-02 -3.01396498e-03  2.04960735e-02  1.99655434e-03
   4.25864680e-03 -7.17227639e-03 -3.33681041e-03  1.00000000e+00
   6.04064308e-02]
 [-1.69490060e-02  3.44582446e-03 -3.90182535e-03  2.55848165e-02
  -1.47236478e-02  3.67354963e-03  4.17566138e-03 -1.45791480e-02
   3.29872381e-03  1.20336931e-02  9.91495192e-01  6.04064308e-02
   1.00000000e+00]]
```

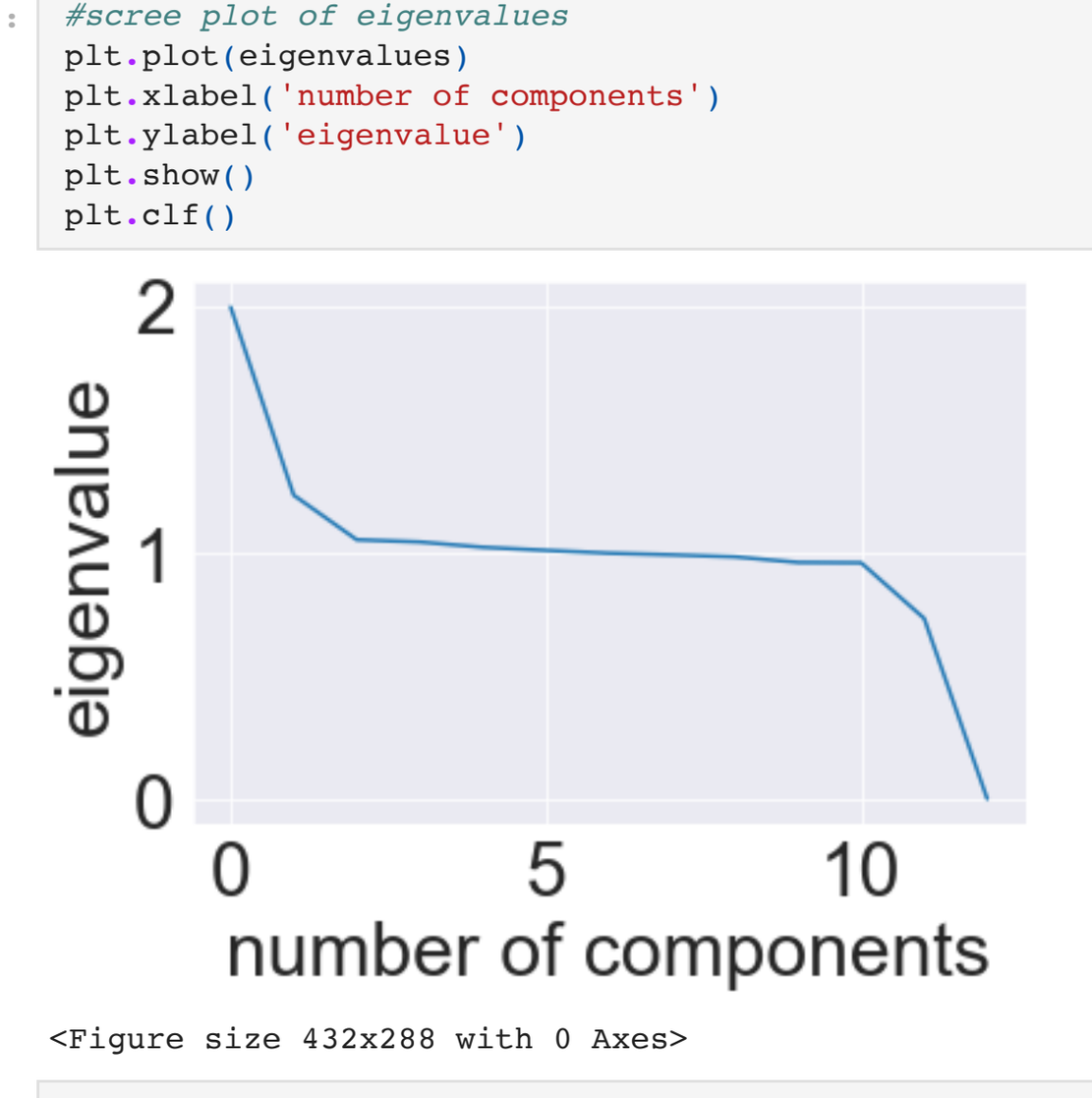
In [16]:
```python
#increase readablity of covariance matrix as heatmap in seaborn
plt.figure(figsize=(40,20))
sns.heatmap(cov_matrix,xticklabels = df_continuous.columns, yticklabels = df_continuous.columns,vmin = -1,\
            center = 0, vmax = 1, cmap='PuOr', annot=True, fmt='.1g', square = True)
plt.clf()
```

```
<Figure size 2880x1440 with 0 Axes>
```

In [17]:
```python
#develop eigen values for further visualization
eigenvalues = [np.dot(eigenvector.T, np.dot(cov_matrix, eigenvector)) for eigenvector in pca.components_]
for pc, eig in zip(pc_columns, eigenvalues):
    print(pc,eig)
```

```
PC1 1.9947054618579014
PC2 1.2340280493497051
PC3 1.0536643989384982
PC4 1.0445900393374434
PC5 1.0231857472453192
PC6 1.0106807421123258
PC7 0.9992026494391042
PC8 0.9925759143925394
PC9 0.9844326023843165
PC10 0.9617902162889184
PC11 0.9603992599773483
PC12 0.7352794109644142
PC13 0.00546550771217469
```

In [18]:
```python
#scree plot of eigenvalues
plt.plot(eigenvalues)
plt.xlabel('number of components')
plt.ylabel('eigenvalue')
plt.show()
plt.clf()
```




```
<Figure size 432x288 with 0 Axes>
```

In [ ]:

In [ ]: