In [1]:

```
# Do not edit this cell

# course: 3654
# a: Homework 7
# d: VT
```

Homework 7

Enter your Name: Catherine Squillante

Enter your PID: cat1997

I have neither given nor received unauthorized assistance on this assignment. See the course sylabus for details on the Honor Code policy. In particular, sharing lines of solution code is prohibited.

In [2]:

```
# Run this cell first.  Do NOT edit this cell.
Answer1 = Answer2 = Answer3 = Answer4 = Answer5 = None
import pandas
import numpy
import matplotlib
import matplotlib.pyplot
import sklearn.cluster
import sklearn.manifold
#%matplotlib inline
states = pandas.read_csv('State_demographics.csv')
survey = pandas.read_csv('Survey-3654-Fall2019-clean.csv')
states.shape, survey.shape
```

Out[2]:

```
((51, 52), (71, 35))
```

Problem 1. (20 points) What are 5 clusters in the States data? Extract and z-score normalize the quantitative columns of the State-demographics data. Then, compute k=5 clusters of states using the k-means algorithm.

For grading purposes, eliminate the randomness of the initial step of k-means by initializing the 5 centroids using these data points in this order: 'CA','DC','LA','MT','NH' (hint: 'init' parameter of KMeans, and n_init=1). Use the default values for other unspecified parameters.

In Answer1, return a DataFrame containing only 'State' column and 'Cluster' label column, sorted by increasing Cluster label.

In [3]:

```
# Problem 1
# Insert your work here
state = states
numerics = ['int16', 'int32', 'int64', 'float16', 'float32', 'float64']
s = state.select_dtypes(include=numerics)
s = ((s - s.mean())/s.std()).set_index(state.Abbrev)
clust = s.loc[['CA','DC','LA','MT','NH'],:]
km = sklearn.cluster.KMeans(n_clusters=5, n_init = 1, init = clust)
labels = km.fit_predict(s)
state['Cluster'] = labels
Answer1 = state.loc[:,['State','Cluster']].sort_values('Cluster')
Answer1
```

Out[3]:

	State	Cluster
9	Florida	0
4	California	0
43	Texas	0
32	New York	0
8	District Of Columbia	1
49	Wisconsin	2
28	Nevada	2
31	New Mexico	2
33	North Carolina	2
35	Ohio	2
37	Oregon	2
0	Alabama	2
24	Mississippi	2
40	South Carolina	2
42	Tennessee	2
46	Virginia	2
47	Washington	2
48	West Virginia	2
38	Pennsylvania	2
22	Michigan	2
25	Missouri	2
2	Arizona	2
18	Louisiana	2
17	Kentucky	2
3	Arkansas	2
10	Georgia	2
11	Hawaii	2
14	Indiana	2
13	Illinois	2
12	Idaho	3
1	Alaska	3
44	Utah	3
5	Colorado	3

	State	Cluster
41	South Dakota	3
36	Oklahoma	3
50	Wyoming	3
34	North Dakota	3
15	Iowa	3
16	Kansas	3
27	Nebraska	3
26	Montana	3
23	Minnesota	3
30	New Jersey	4
39	Rhode Island	4
7	Delaware	4
29	New Hampshire	4
6	Connecticut	4
45	Vermont	4
19	Maine	4
20	Maryland	4
21	Massachusetts	4

Problem 2. (20 points) How can the data be reduced to 2 dimensions? Use MDS with L2 Euclidean distance to reduce the dimensionality of the same z-scored quantitative columns to 2 dimensions (hint: MDS can compute L2 distances for you).

For grading purposes, eliminate the randomness of the initial step of MDS by initializing the 2 reduced dimensions with the data in columns: "Education.Bachelor's Degree or Higher", "Income.Per Capita Income" (hint: 'init' parameter of MDS.fit_transform). Set n_init=1, eps=0 and max_iter=1000. Use the default values for other unspecified parameters.

In Answer2, return a DataFrame containing the 'State' column and the new 'X' and 'Y' columns, sorted by increasing 'Y'.

In [4]:

```
# Problem 2
# Insert your work here
states = pandas.read_csv('State_demographics.csv')
b = states.select_dtypes(include=numerics)
b = ((b - b.mean())/b.std())
mds = sklearn.manifold.MDS(n_init=1, max_iter=1000, dissimilarity = 'euclidean', ep
s=0)
data2D = mds.fit_transform(b, init = b[['Education.Bachelor\'s Degree or Higher','I
ncome.Per Capita Income']])
data2D = pandas.DataFrame(data2D, columns=['X','Y'])
data2D['State'] = state.State
data2D = data2D[['State', 'X', 'Y']]
Answer2 = data2D.sort_values('Y')
Answer2
```

Out[4]:

	State	X	Y
31	New Mexico	-0.285137	-9.159046
24	Mississippi	0.236394	-6.536576
43	Texas	12.722746	-6.212489
48	West Virginia	-3.232024	-5.767393
44	Utah	-7.130742	-5.073875
18	Louisiana	0.986791	-4.484911
36	Oklahoma	-3.684575	-4.263667
3	Arkansas	-1.906322	-4.072289
0	Alabama	0.150749	-3.991878
2	Arizona	2.370230	-3.578496
40	South Carolina	0.331193	-3.179548
17	Kentucky	-1.545422	-3.126270
10	Georgia	4.315692	-2.824359
12	Idaho	-4.546232	-2.563115
42	Tennessee	0.147229	-2.375492
33	North Carolina	2.286076	-2.200765
9	Florida	9.611176	-2.064536
41	South Dakota	-6.045716	-1.864550
14	Indiana	-0.536119	-1.757652
22	Michigan	1.849720	-1.431629
25	Missouri	-0.824118	-1.376733
35	Ohio	2.503063	-1.339809
4	California	18.099671	-1.258182
16	Kansas	-2.677611	-1.002323
26	Montana	-5.394458	-0.711747
27	Nebraska	-3.745631	-0.674639
15	Iowa	-3.234172	-0.469964
38	Pennsylvania	3.579552	-0.370387
34	North Dakota	-7.233937	-0.350347
49	Wisconsin	-1.454282	-0.299537
50	Wyoming	-5.962055	0.383342
13	Illinois	4.836855	0.485074
37	Oregon	-0.919002	0.830576

	State	X	Y
23	Minnesota	-1.642234	0.868231
46	Virginia	2.477981	1.782247
47	Washington	0.900486	1.955401
19	Maine	-4.783378	2.085264
32	New York	9.377373	2.106633
5	Colorado	-0.047304	2.246286
45	Vermont	-4.996300	3.021205
1	Alaska	-12.043828	3.521970
7	Delaware	-2.492018	3.645289
39	Rhode Island	-1.180261	3.794525
30	New Jersey	4.908768	3.916304
21	Massachusetts	2.221147	4.182440
6	Connecticut	0.608190	4.355831
29	New Hampshire	-4.395069	4.471688
20	Maryland	3.596402	4.765091
28	Nevada	-0.459785	5.602439
8	District Of Columbia	7.083936	13.317260
11	Hawaii	-2.803688	17.045108

Problem 3. (20 points) How would you describe the categorization of the states? Put the previous results together in a visualization. Draw a scatterplot of the MDS result. Color the dots by their cluster memberships. (What type of colormap should you use?) Label each dot with its state abbreviation (hint: axes.text( )). Compute the 2D cluster centroids of the 2-dimensional X,Y data from MDS, and plot the centroids in the same plot, using the same color scheme, but make the centroids dots much larger than the state dots and give them transparency (alpha).

In Answer3, return the 2D centroids as a DataFrame with columns 'X','Y', indexed and sorted by cluster label.

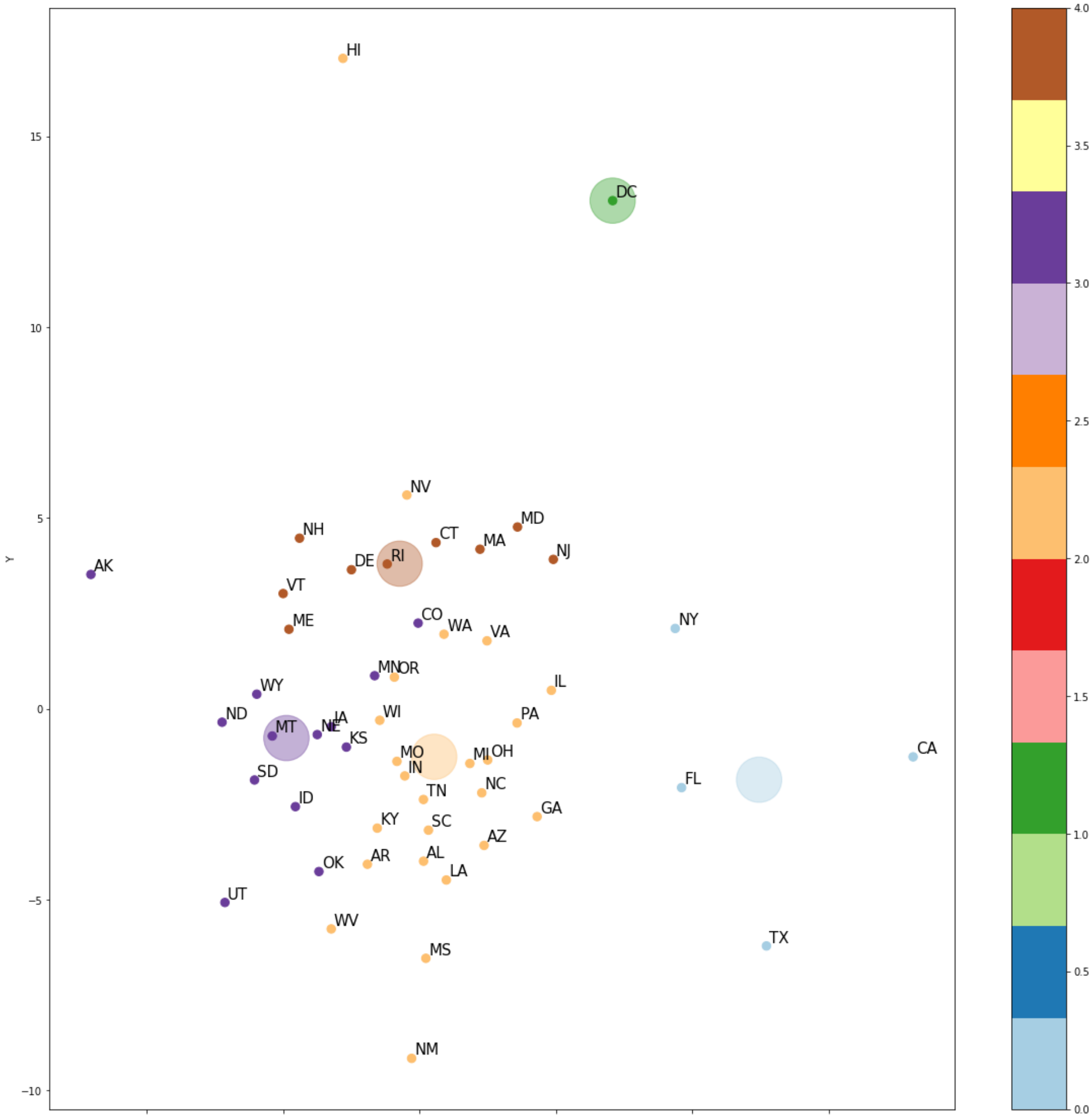
Think about: Do the clusters overlap? Are the points always nearest to their own centroid? Why might the clusters not necessarily look strictly clustered in the MDS plot? Hint: think high-dimensionally.

In [5]:

```
# Problem 3
# Insert your work here
data = pandas.DataFrame(data2D)
data['Cluster'] = labels
data['Abbrev'] = states.Abbrev
centroids = data.groupby('Cluster').mean()
data['Cluster'] = labels
ax = data.plot.scatter(x='X', y='Y', cmap = 'Paired', c = data.Cluster, s = 70)
ax.figure.set_size_inches(20, 20, forward = True)
for i in range(len(data.X)):
    x = data.X[i]
    y = data.Y[i]
    ax.axes.text(x + .1, y + .1, data.Abbrev[i], size = 15)
ax.scatter(x = centroids.X, y = centroids.Y, cmap = 'Paired', c = centroids.index,
s = 2000, alpha = .4)
Answer3 = centroids
Answer3
```

Out[5]:

	X	Y
Cluster		
0	12.452741	-1.857144
1	7.083936	13.317260
2	0.541921	-1.257163
3	-4.876038	-0.765723
4	-0.723613	3.804182


HI
DC
NV
NH
MD
CT
MA
NJ
DE
RI
AK
VT
ME
CO
WA
VA
NY
MN
OR
IL
WY
ND
WI
PA
MT
NE
IA
KS
MO
MI
OH
CA
IN
FL
SD
TN
NC
ID
GA
KY
SC
AZ
AL
AR
OK
LA
UT
WV
TX
MS
NM
Y
15
10
5
0
−5
−10
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0

Problem 4. (20 points) Is there a natural number of clusters for the States data? Conduct an "elbow" analysis by re-running k-means with all possible values of k. Display a line plot, with circle markers, of 'total within-cluster variance' (kmeans.inertia_) as a function of k. To get good results, you will want to use the default init='k-means++' parameter. For reasonable running times, use n_init=3 and max_iter=20.

In Answer4, return the results as a DataFrame with columns 'K' and 'Inertia', sorted by increasing K.

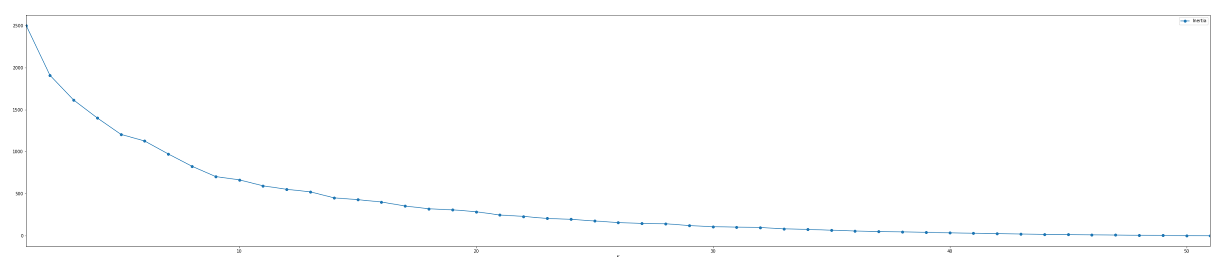
In [6]:

```
# Problem 4
# Insert your work here
state1 = states
a = state1.select_dtypes(include=numerics)
a = ((a - a.mean())/a.std())
km = sklearn.cluster.KMeans(n_init = 3, init = 'k-means++', max_iter = 20)
labels = km.fit_predict(a)
cluster_var = []
K = []
for k in range(len(a)):
        K.append(k+1)
        km = sklearn.cluster.KMeans(n_init = 3, init = 'k-means++', max_iter = 20,
n_clusters = k+1)
        labels = km.fit_predict(a)
        cluster_var.append(km.inertia_)
frame = pandas.DataFrame(data = list(zip(K, cluster_var)), columns = ['K', 'Inerti
a'])
plt1 = frame.plot(x = 'K', y = 'Inertia', marker = 'o')
plt1.figure.set_size_inches(50, 10, forward = True)
Answer4 = frame
Answer4
```

Out[6]:

	K	Inertia
0	1	2500.000000
1	2	1909.199852
2	3	1614.775368
3	4	1401.290026
4	5	1205.994448
5	6	1126.934473
6	7	972.714719
7	8	826.428413
8	9	704.030345
9	10	664.801159
10	11	594.020588
11	12	552.374320
12	13	522.081832
13	14	450.713894
14	15	429.426741
15	16	401.463530
16	17	353.765051
17	18	320.846765
18	19	308.478969
19	20	285.755517
20	21	246.704955
21	22	230.277793
22	23	205.507449
23	24	195.384940
24	25	176.024750
25	26	156.165179
26	27	147.481003
27	28	142.805372
28	29	120.894479
29	30	108.043313
30	31	102.766735
31	32	97.720440
32	33	82.387000

	K	Inertia
33	34	75.933410
34	35	65.715998
35	36	56.452642
36	37	50.200780
37	38	46.165187
38	39	41.126106
39	40	34.576571
40	41	29.506672
41	42	24.546219
42	43	20.742362
43	44	16.925678
44	45	14.085092
45	46	10.979101
46	47	8.345700
47	48	5.532980
48	49	3.466384
49	50	1.621574
50	51	0.000000



Problem 5. (20 points) Complete the following sentence: "There are two kinds of people in the world (well, in our class anyway), ..." How would you describe those two kinds of people?

Using the Survey data, eliminate the Faculty member, z-score normalize the quantitative data, and then use k-means and Parallel Coordinates visualization of centroids to find the answer. Hint: Use clustering, and find out what is most different about their centroids. Rerun your analysis several times to see what columns are most consistently most different. Since this data is more complex, use n_init=100, max_iter=100. Visually justify your claim with a Parallel Coordinates plot of the z-score centroids.

In Answer5, return a Series, indexed by the quantitative column names, containing the absolute-value difference between the z-score centroids, sorted in decreasing magnitude.

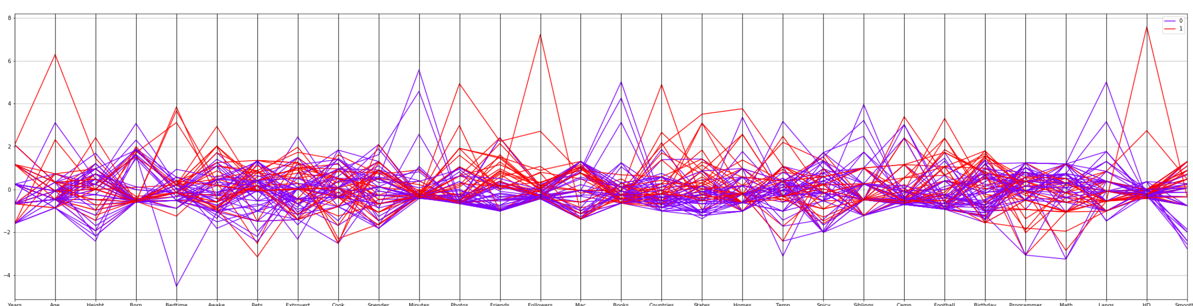
In [69]:

```
# Problem 5
# Insert your work here

Answer5
sur = survey.iloc[:-1,:]
sur = sur.select_dtypes(include=numerics)
norm = ((sur - sur.mean())/sur.std())
km = sklearn.cluster.KMeans(n_init = 100,max_iter = 100, n_clusters = 2)
for i in range(10):
    labels = km.fit_predict(norm)
    norm['Cluster'] = labels
    centroids = norm.groupby('Cluster').mean()
data = centroids.iloc[0,:] - centroids.iloc[1,:]
diff = pandas.Series(data = abs(data), index = centroids.columns).sort_values(ascen
ding = False)
plot = pandas.plotting.parallel_coordinates(norm, 'Cluster', colormap = matplotlib.
pyplot.cm.rainbow)
plot.figure.set_size_inches(40, 10, forward = True)
Answer5 = diff
Answer5
```

Out[69]:

```
States        1.080252
Friends       1.041460
Football      0.852859
Smoothie      0.769744
Photos        0.758029
Followers     0.742038
Extrovert     0.670417
Math          0.657830
Countries     0.607166
Programmer    0.555444
Bedtime       0.538872
Awake         0.522170
Langs         0.518064
Birthday      0.505303
Books         0.486903
Camp          0.453199
Years         0.415373
HD            0.365748
Temp          0.359060
Minutes       0.335067
Mac           0.328325
Born          0.304382
Age           0.274282
Siblings      0.264215
Homes         0.255923
Height        0.249924
Pets          0.247157
Cook          0.241787
Spender       0.073314
Spicy         0.043821
dtype: float64
```



In [54]:

```
# scratch space
```

In [ ]:

```

```